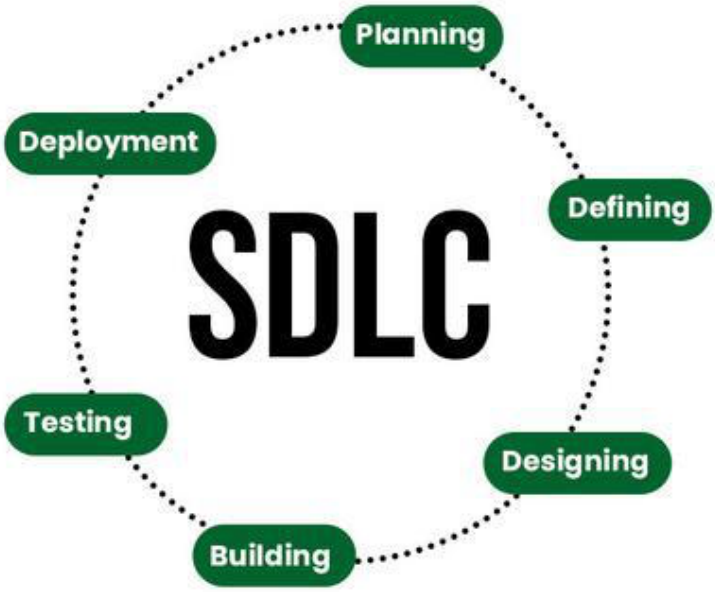
Chapter 2

SOFTWARE PROCESS MODELS AND AGILITY

2.1. Software Development Lifecycle (SDLC):

- Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software.
- SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.
- SDLC is a process followed for software building within a software organization.
- SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software.
- The life cycle defines a method for improving the quality of software and the all-around development process



.

Stages of the Software Development Life Cycle

SDLC is a collection of these six stages, and the stages of SDLC are as follows:

Stage-1: Planning and Requirement Analysis

- Planning is a crucial step in everything, just as in software Development.
- In this same stage, requirement analysis is also performed by the developers of the organization.
- This is attained from customer inputs, and sales department/market surveys.
- The information from this analysis forms the building blocks of a basic project.
- The quality of the project is a result of planning.
- Thus, in this stage, the basic project is designed with all the available information.

Stage-2: Defining Requirements

- In this stage, all the requirements for the target software are specified.
- These requirements get approval from customers, market analysts, and stakeholders.
- This is fulfilled by utilizing SRS (Software Requirement Specification).
- This is a sort of document that specifies all those things that need to be defined and created during the entire project cycle.

Stage-3: Designing Architecture

- SRS is a reference for software designers to come up with the best architecture for the software.
- Hence, with the requirements defined in SRS, multiple designs for the product architecture are present in the Design Document Specification (DDS).
- This DDS is assessed by market analysts and stakeholders.
- After evaluating all the possible factors, the most practical and logical design is chosen for development.

Stage-4: Developing Product

- At this stage, the fundamental development of the product starts.
- For this, developers use a specific programming code as per the design in the DDS.
- Hence, it is important for the coders to follow the protocols set by the association.
- Conventional programming tools like compilers, interpreters, debuggers, etc. are also put into use at this stage.
- Some popular languages like C/C++, Python, Java, etc. are put into use as per the software regulations.

Stage-5: Product Testing and Integration

- After the development of the product, testing of the software is necessary to ensure its smooth execution.
- Although, minimal testing is conducted at every stage of SDLC.
- Therefore, at this stage, all the probable flaws are tracked, fixed, and retested.
- This ensures that the product confronts the quality requirements of SRS.

Stage-6: Deployment and Maintenance of Products

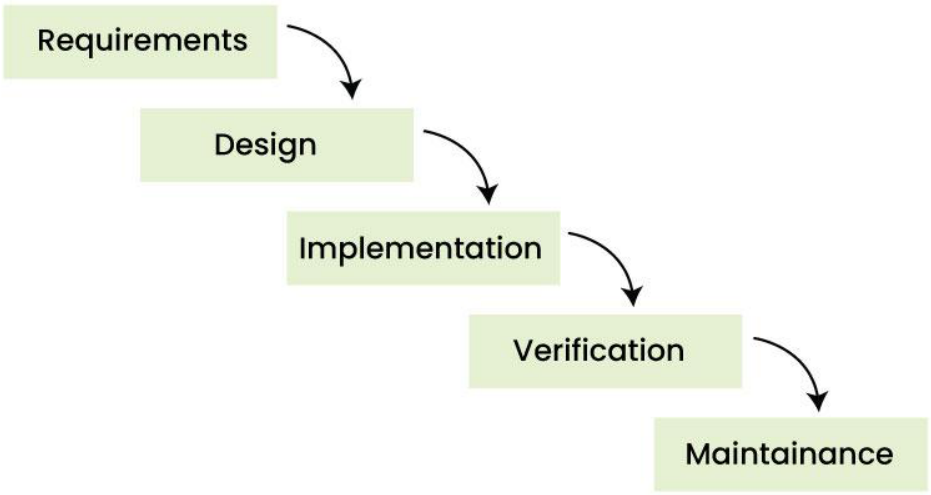
- After detailed testing, the conclusive product is released in phases as per the organization's strategy. Then it is tested in a real industrial environment.
- It is important to ensure its smooth performance.
- If it performs well, the organization sends out the product as a whole.
- After retrieving beneficial feedback, the company releases it as it is or with auxiliary improvements to make it further helpful for the customers.

2.2. Software Development Life Cycle Models:

1. Waterfall Model:

- It is a famous and good version of SDLC for software engineering.
- The waterfall model is a linear and sequential model, which means that a development phase cannot begin until the previous phase is completed.
- We cannot overlap phases in waterfall model.
- We can imagine waterfall in the following way "Once the water starts flowing over the edge of the cliff, it starts falling down the mountain and the water cannot go back up."

Phases of Waterfall model:



Waterfall Model

Requirement phase:- Requirement phase is the first phase of the waterfall model. In this phase the requirements of the system are collected and documented. This phase is very crucial because the next phases are based on this phase.

Design phase:- Design phase is based on the fact how the software will be built. The main objective of the design phase is to prepare the blueprint of the software system so that no problems are faced in the coming phases and solutions to all the requirements in the requirement phase are found.

Implementation phase:- In this phase, hardware, software and application programs are installed and the database design is implemented. Before the database design can be implemented, the software has to go through a testing, coding, and debugging process. This is the longest lasting phase in waterfall.

Verification phase:- In this phase the software is verified and it is evaluated that we have created the right product. In this phase, various types of testing are done and every area of the software is checked. It is believed that if we do not verify the software properly and there is any defect in it then no one will use it, hence verification is very important. One advantage of verification is that it reduces the risk of software failure.

Maintenance phase:- This is the last phase of waterfall. When the system is ready and users start using it, then the problems that arise have to be solved time-to-time. Taking care of the finished software and maintaining it as per time is called maintenance.

Advantages of Waterfall Model

- This model is simple and easy to understand.
- This is very useful for small projects.
- This model is easy to manage.
- The end goal is determined early.
- Each phase of this model is well explained.
- It provides a structured way to do things.

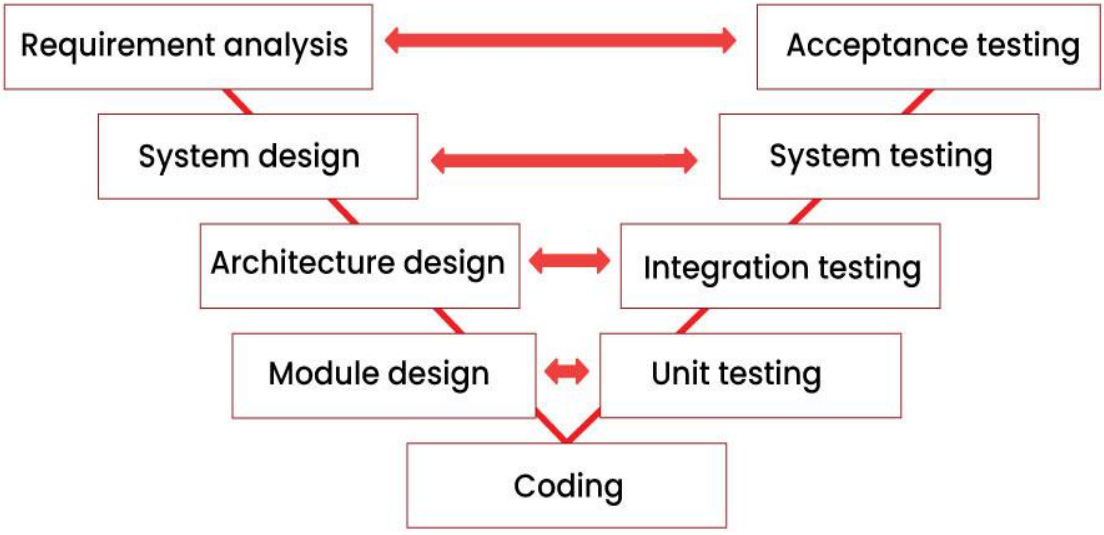
Disadvantages of Waterfall Model

- In this model, complete and accurate requirements are expected at the beginning of the development process.
- Working software is not available for very long during the development life cycle.
- We cannot go back to the previous phase due to which it is very difficult to change the requirements.
- Risk is not assessed in this, hence there is high risk and uncertainty in this model.
- In this the testing period comes very late.
- Due to its sequential nature this model is not realistic in today's world.
- This is not a good model for large and complex projects.

2. **V-Model**

- V-Model is an SDLC model, it is also called Verification and Validation Model.
- V-Model is widely used in the Software Development Process, and it is considered a disciplined model.
- In V-Model, the execution of each process is sequential, that is, the new phase starts only after the previous phase ends.
- It is based on the association of testing phase with each development phase that is in V-Model with each development phase, its testing phase is also associated in a V-shape in other words both software development and testing activities take place at the same time.

Phases of V-model



Requirements analysis:- This is the first phase of the development cycle, in which the requirements of the product are analyzed according to the customer's needs. In this phase, product related requirements are thoroughly collected from the customer. This is a very important phase because this phase determines the coming phases. In this phase, acceptance tests are designed for later use.

System design:- When we have the requirements of the product, after that we prepare a complete design of the system. In this phase, a complete description of the hardware and all the technical components required to develop the product .

Architectural design:- In this phase architectural specifications are designed. It contains the specification of how the software will link internally and externally with all the components. Therefore this phase is also called high level design (HLD).

Module design:- In this phase the internal design of all the modules of the system is specified. Therefore it is called low level design (LLD). It is very important that the design of all modules should be according to the system architecture. Unit tests are also designed in the module design phase.

Coding phase:- In the coding phase, coding of the design and specification done in the previous phases is done. This phase takes the most time.

Validation Phases of V-Model

Unit testing:- In the unit testing phase, the unit tests created during the module design phase are executed. Unit testing is code level testing, it only verifies the technical design. Therefore it is not able to test all the defects.

Integration testing:- In integration testing, the integration tests created in the architectural design phase are executed. Integration testing ensures that all modules are working well together.

System testing:– In system testing, the system tests created in the system design phase are executed. System tests check the complete functionality of the system. In this, more attention is given to performance testing and regression testing.

Acceptance testing:- In acceptance testing, the acceptance tests created in the requirement analysis phase are executed. This testing ensures that the system is compatible with other systems. And in this, non-functional issues like:- load time, performance etc. are tested in the user environment.

Advantages of V-Model

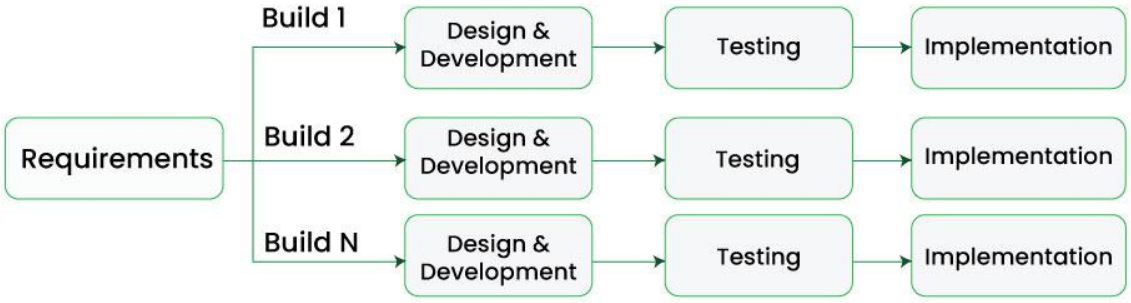
- This is a simple and easy to use model.
- Planning, testing and designing tests can be done even before coding.
- This is a very disciplined model, in which phase by phase development and testing goes on.
- Defects are detected in the initial stage itself.
- Small and medium scale developments can be easily completed using it.

Disadvantages of V-Model

- This model is not suitable for any complex projects.
- There remains both high risk and uncertainty.
- This is not a suitable model for an ongoing project.
- This model is not at all suitable for a project which is unclear and in which there are changes in the requirement.

3. **Incremental Model**

- In Incremental Model, the software development Process is divided into several increments and the same phases are followed in each increment.
- In simple language, under this model a complex project is developed in many modules or builds.
- For example, we collect the customer's requirements, now instead of making the entire software at once, we first take some requirements and based on them create a module or function of the software and deliver it to the customer. Then we take some more requirements and based on them add another module to that software.



Phases of Incremental Model

Communication: In the first phase, we talk face to face with the customer and collect his mandatory requirements. Like what functionalities does the customer want in his software, etc.

Planning: In this phase the requirements are divided into multiple modules and planning is done on their basis.

Modeling: In this phase the design of each module is prepared. After the design is ready, we take a particular module among many modules and save it in DDS (Design Document Specification). Diagrams like ERDs and DFDs are included in this document.

Construction: Here we start construction based on the design of that particular module. That is, the design of the module is implemented in coding. Once the code is written, it is tested.

Deployment: After the testing of the code is completed, if the module is working properly then it is given to the customer for use. After this, the next module is developed through the same phases and is combined with the previous module. This makes new functionality available to the customer. This will continue until complete modules are developed.

Advantages of Incremental Model

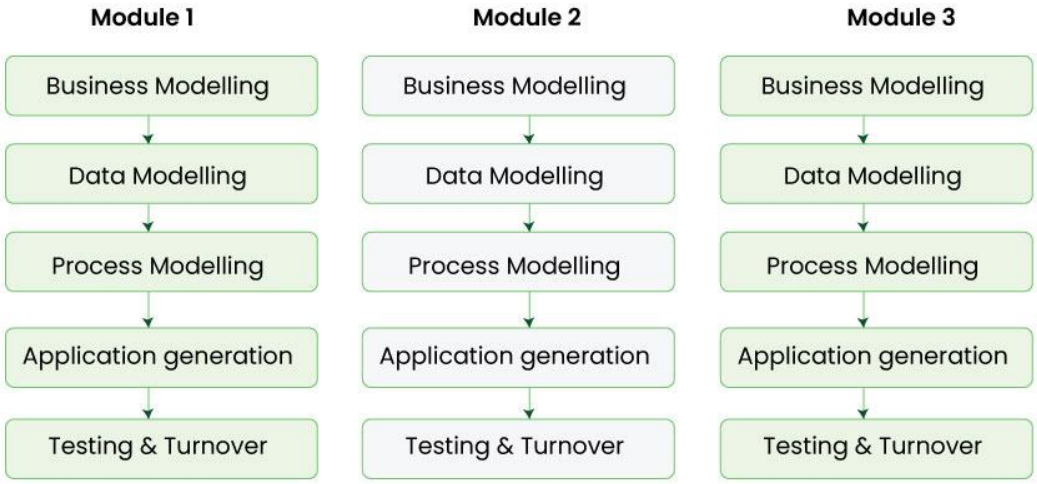
- This model is flexible and less expensive to change requirements and scope.
- The customer can respond to each module and provide feedback if any changes are needed.
- Project progress can be measured.
- It is easier to test and debug during a short iteration.
- Errors are easy to identify.

Disadvantages of Incremental Model

- Management is a continuous activity that must be handled.
- Before the project can be dismantled and built incrementally,
- The complete requirements of the software should be clear.
- This requires good planning and designing.
- The total cost of this model is higher.

4. **RAD Model**

- RAD model stands for rapid application development model.
- The methodology of RAD model is similar to that of incremental or waterfall model.
- It is used for small projects.
- If the project is large then it is divided into many small projects and these small projects are planned one by one and completed.
- In this way, by completing small projects, the large project gets ready quickly.



RAD Model

Phases RAD model

Business modeling: In this phase, the business model is designed on the basis of whatever functions the business has. If we speak in a little technical language, then we design the business model for the product on the basis of flow of information and distribution of information between different business channels. Here information flow means what type of information drives the business, where the information comes from and where it goes, who generates it, etc. This means that a complete business analysis is done in this phase.

Data modeling: Using the business model we had prepared, the data objects required for the business are defined.

Process modeling: The data objects that we defined in the data modeling phase are converted to establish the business information flow. It is necessary to achieve specific business objectives.

Application generation: In this phase we start building the software based on the output of the above three phases. For this we take the help of automation tools. However, in this phase we do not develop the actual software but make a working prototype.

Testing and turnover: Whatever prototype we have prepared or whatever components and interfaces we have, they are tested in this phase. Since prototypes are tested separately during each iteration, the overall testing time in rapid application development is reduced.

Advantage of RAD Model

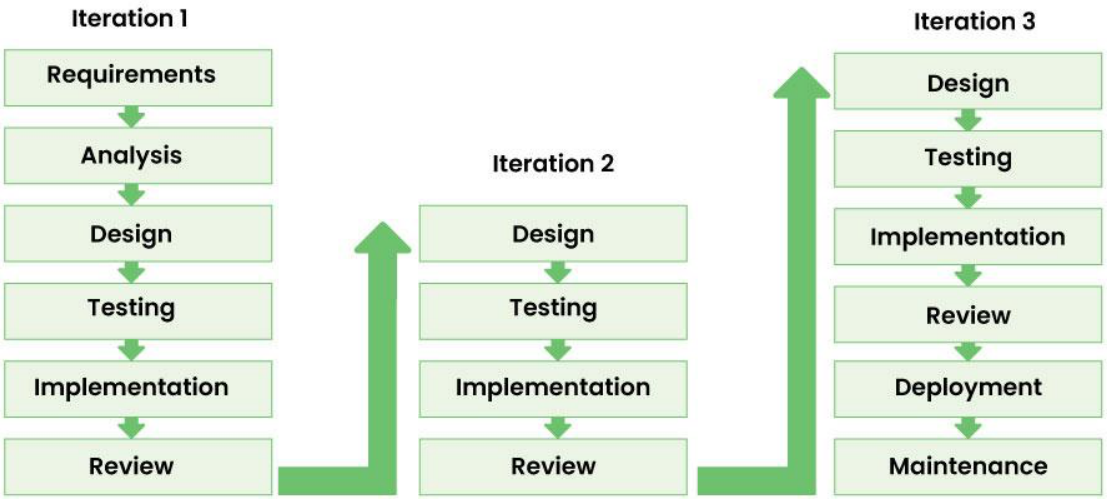
- It reduces the time taken in development.
- In this the components are reused.
- It is flexible and it is easy to make any changes in it.
- There are very few defects in it because it is a prototype by nature.
- In this, productivity can be increased in less time with less people.
- It is cost effective.
- It is suitable for small projects.

Disadvantages of RAD Model

- In this we need highly skilled developers and designers.
- It is very difficult to manage.
- It is not suitable for project that are complex and takes long time.
- In this, feedback from the client is required for the development of each phase.
- Automated code generation is very expensive.
- This model is suitable only for component based and scalable systems.

5. **Iterative Model**

- In Iterative model we start developing the software with some requirements and when it is developed, it is reviewed.
- If there are requirements for changes in it, then we develop a new version of the software based on those requirements.
- This process repeats itself many times until we get our final product.
- After the first version is developed, if there is a need to change the software, then a new version is developed with the second iteration.



Phases of iterative model

Requirement gathering & analysis: In this phase, all the software requirements of the customer are collected and it is analyzed whether those requirements can be meet or not. Besides, it is also checked whether this project will not go beyond our budget.

Design: In this phase the design of software is prepared. For this, various diagrams like Data Flow diagram, class diagram, activity diagram, state transition diagram, etc. are used.

Implementation: Now the design of software is implemented in coding through various programming languages. We also call this coding phase.

Testing: After the coding of the software is done, it is now tested so that the bugs and errors present in it can be identified. To do this, various testing techniques like performance testing, security testing, requirement testing, stress testing, etc. are done.

Deployment: Finally the software is given to the customer. After this the customer starts using that software in his work environment.

Review: After the software is deployed in its work environment, it is reviewed. If any error/bug is found or any new requirements come in front of developer, then again these phases are repeated with new iteration and a new version is developed.

Maintenance: In this phase we look at customer feedback, solve problems, fix errors, update software, etc.

Advantage of Iterative model

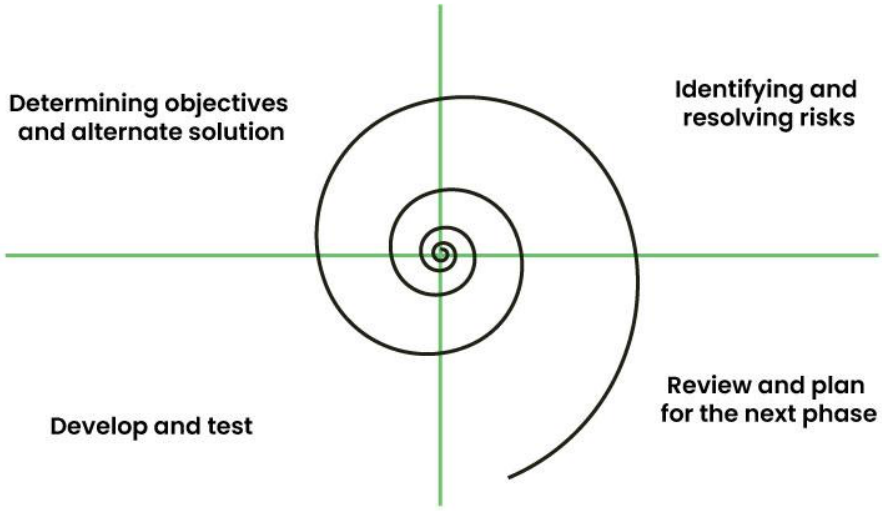
- In iterative models, bugs and errors can be identified quickly.
- Under this model, software is prepared quickly with some specifications.
- Testing and debugging the software becomes easier during each iteration.
- We get reliable feedback from users along with blueprints.
- This model is easily adaptable to constantly changing needs.
- Risks are identified and resolved during iteration.

Disadvantage of Iterative model:

- Iterative model is not suitable for small projects.
- Since the requirements are constantly changing, we have to make frequent changes in the software.
- Due to constantly changing requirements, the budget of the project also increases and it takes more time to complete it.
- In this model, it is complicated to control the entire process of software development.
- It is very difficult to tell by what date the complete software will be ready.

6. **Spiral Model**

- This model has characteristics of both iterative and waterfall models.
- This model is used in projects which are large and complex.
- This model was named spiral because if we look at its figure, it looks like a spiral, in which a long curved line starts from the center point and makes many loops around it.
- The number of loops in the spiral is not decided in advance but it depends on the size of the project and the changing requirements of the user.
- We also call each loop of the spiral a phase of the software development process.
- It was first developed by Barry Boehm in 1986.



Spiral model

These phases are as follows:-

Determining objectives and alternate solutions: In the first phase, whatever requirements the customer has related to the software are collected. On the basis of which objectives are identified and analyzed and various alternative solutions are proposed.

Identifying and resolving risks: In this phase, all the proposed solutions are assessed and the best solution is selected. Now that solution is analyzed and the risks related to it are identified. Now the identified risks are resolved through some best strategy.

Develop and test: Now the development of software is started. In this phase various features are implemented, that is, their coding is done. Then those features are verified through testing.

Review and plan for the next phase: In this phase the developed version of the software is given to the customer and he evaluates it. Gives his feedback and tells new requirements. Finally planning for the next phase (next spiral) is started.

Advantages of Spiral Model

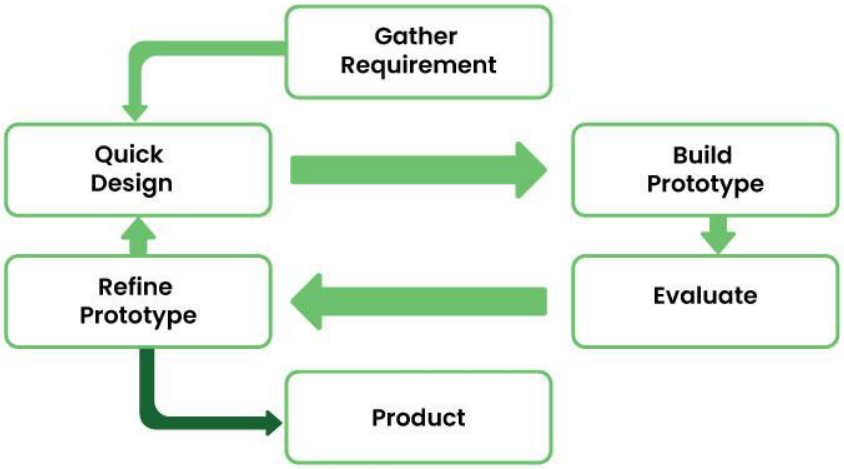
- If we have to add additional functionality or make any changes to the software, then through this model we can do so in the later stages also.
- Spiral model is suitable for large and complex projects.
- It is easy to estimate how much the project will cost.
- Risk analysis is done in each phase of this model.
- Since continuous feedback is taken from the customer during the development process, the chances of customer satisfaction increases.

Disadvantage of Spiral Model

- This is the most complex model of SDLC, due to which it is quite difficult to manage.
- This model is not suitable for small projects.
- The cost of this model is quite high.
- It requires more documentation than other models.
- Experienced experts are required to evaluate and review the project from time to time.
- Using this model, the success of the project depends greatly on the risk analysis phase.

7. **Prototype model**

- Prototype model is an activity in which prototypes of software applications are created.
- First a prototype is created and then the final product is manufactured based on that prototype.
- The prototype model was developed to overcome the shortcomings of the waterfall model.
- This model is created when we do not know the requirements well.
- The specialty of this model is that this model can be used with other models as well as alone.



Prototype model

The prototype model has the following phases

Requirement gathering: The first step of prototype model is to collect the requirements, although the customer does not know much about the requirements but the major requirements are defined in detail.

Build the initial prototype: In this phase the initial prototype is built. In this some basic requirements are displayed and user interface is made available.

Review the prototype: When the construction of the prototype is completed, it is presented to the end users or customer and feedback is taken from them about this prototype. This feedback is used to further improve the system and possible changes are made to the prototype.

Revise and improve the prototype: When feedback is taken from end users and customers, the prototype is improved on the basis of feedback. If the customer is not satisfied with the prototype, a new prototype is created and this process continues until the customer gets the prototype as per his desire.

Advantages of Prototype model

- Prototype Model is suggested to create applications whose prototype is very easy and which always includes human machine interaction within it.
- When we know only the general objective of creating software, but we do not know anything in detail about input, processing and output. Then in such a situation we make it a Prototype Model.

- When a software developer is not very sure about the capability of an algorithm or its adaptability to an operating system, then in this situation, using a prototype model can be a better option.

Disadvantages of Prototype model

- When the first version of the prototype model is ready, the customer himself often wants small fixes and changes in it rather than rebuilding the system. Whereas if the system is redesigned then more quality will be maintained in it.
- Many compromises can be seen in the first version of the Prototype Model.
- Sometimes a software developer may make compromises in his implementation, just to get the prototype model up and running quickly, and after some time he may become comfortable with making such compromises and may forget that it is completely inappropriate to do so.

8. **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 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. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. 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.

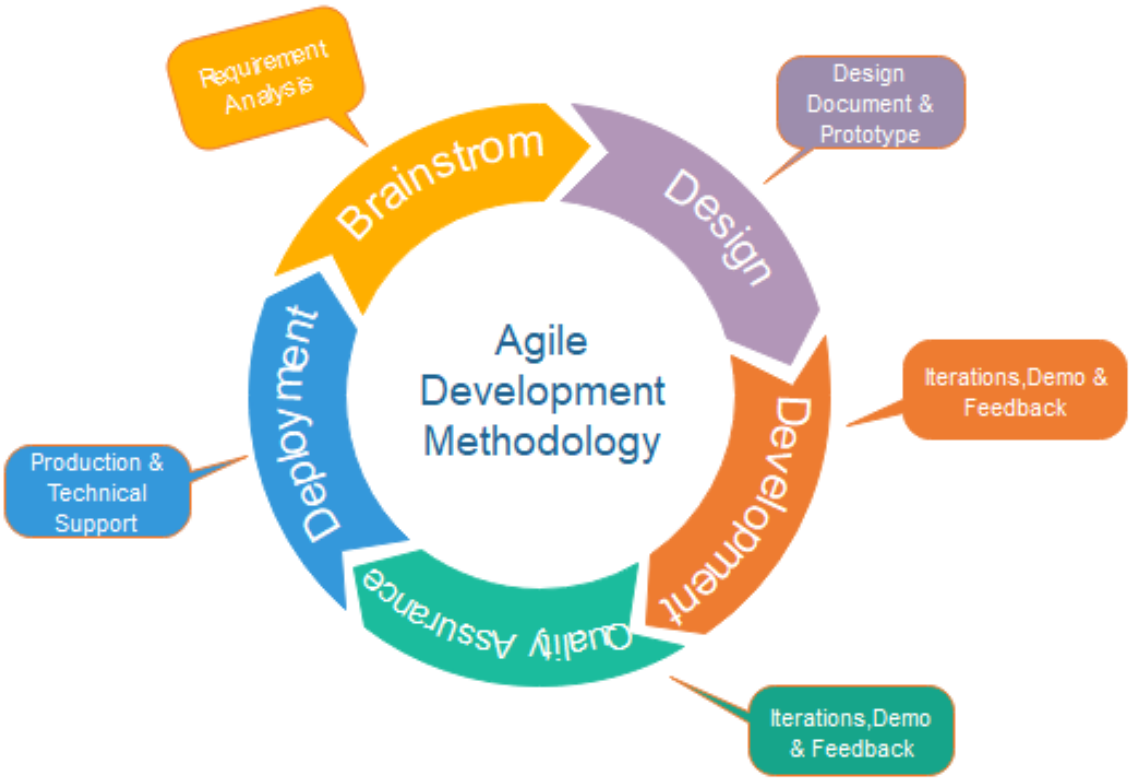


Fig. Agile Model

Phases of Agile Model:
Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

1. **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
3. **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
4. **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
5. **Deployment:** In this phase, the team issues a product for the user's work environment.

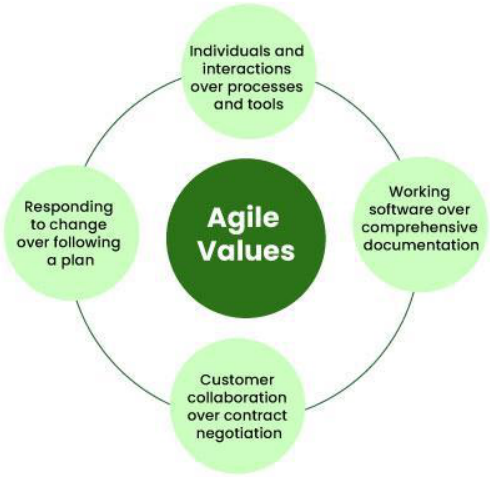
6. **Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Agile software Development Model Methodologies/Frameworks:

- **Scrum:** This framework divides work into "sprints" (usually 1-4 weeks), where teams set goals, complete work, and review progress. Key roles are the Scrum Master, Product Owner, and Development Team.
- **Kanban:** Kanban visualizes work using a board with columns representing stages in the workflow (e.g., To-Do, In Progress, Done). It's particularly suitable for projects where tasks flow continuously.
- **Lean:** Derived from Lean Manufacturing, it focuses on reducing waste, maximizing value, and optimizing processes.
- **Extreme Programming (XP):** This emphasizes engineering practices like test-driven development, pair programming, and continuous integration.

4 Core Values of Agile Software Development

The Agile Software Development Methodology Manifesto describe four core values of Agile in software development.

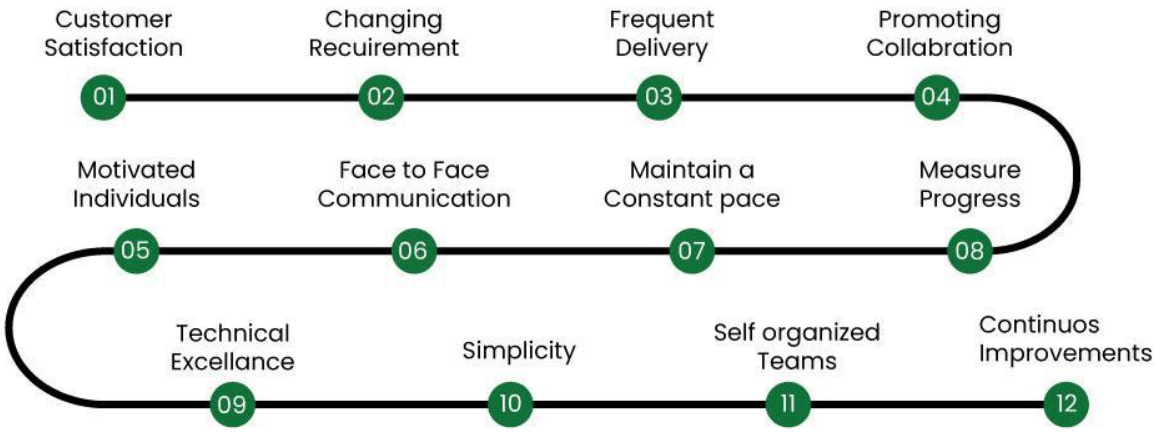


4 Values of Agile

1. Individuals and Interactions over Processes and Tools
2. Working Software over Comprehensive Documentation
3. Customer Collaboration over Contract Negotiation
4. Responding to Change over Following a Plan

12 Principles of Agile Software Development

The Agile Manifesto is based on four values and twelve principles that form the basis, for methodologies.

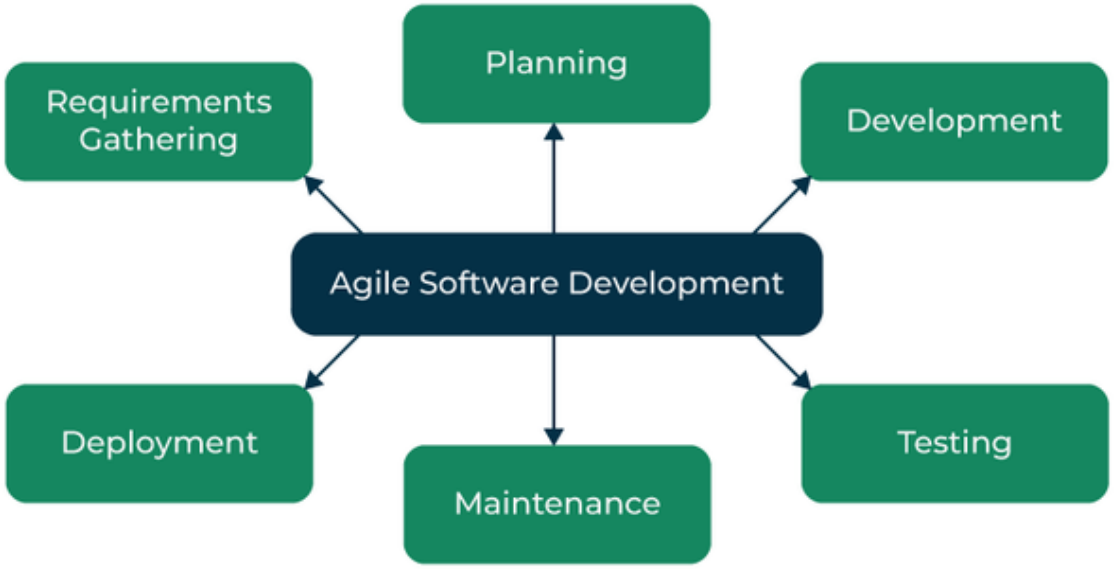


12 Principles of Agile Methodology

These principles include:

1. Ensuring customer satisfaction through the early delivery of software.
2. Being open to changing requirements in the stages of the development.
3. Frequently delivering working software with a main focus on preference for timeframes.
4. Promoting collaboration between business stakeholders and developers as an element.
5. Structuring the projects around individuals. Providing them with the necessary environment and support.
6. Prioritizing face to face communication whenever needed.
7. Considering working software as the measure of the progress.
8. Fostering development by allowing teams to maintain a pace indefinitely.
9. Placing attention on excellence and good design practices.
10. Recognizing the simplicity as crucial factor aiming to maximize productivity by minimizing the work.
11. Encouraging self organizing teams as the approach to design and build systems.
12. Regularly reflecting on how to enhance effectiveness and to make adjustments accordingly.

The Agile Software Development Process



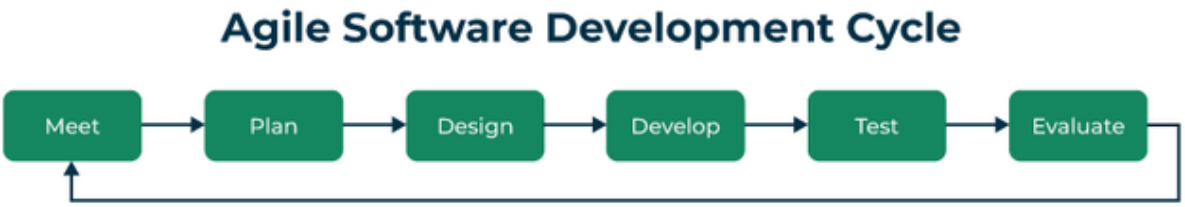
1. Requirement Gathering: The customer's requirements for the software are gathered and prioritized.
2. **Planning:** The development team creates a plan for delivering the software, including the features that will be delivered in each iteration.
3. **Development:** The development team works to build the software, using frequent and rapid iterations.
4. **Testing:** The software is thoroughly tested to ensure that it meets the customer's requirements and is of high quality.
5. **Deployment:** The software is deployed and put into use.
6. **Maintenance:** The software is maintained to ensure that it continues to meet the customer's needs and expectations.

Agile Software Development is widely used by software development teams and is considered to be a flexible and adaptable approach to software development that is well-suited to changing requirements and the fast pace of software development.
Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Agile Software development cycle

Let's see a brief overview of how development occurs in Agile philosophy.

1. concept
2. inception
3. iteration/construction
4. release
5. production
6. retirement



- **Step** 1: In the first step, concept, and business opportunities in each possible project are identified and the amount of time and work needed to complete the project is estimated. Based on their technical and financial viability, projects can then be prioritized and determined which ones are worthwhile pursuing.
- **Step** 2: In the second phase, known as inception, the customer is consulted regarding the initial requirements, team members are selected, and funding is secured. Additionally, a schedule outlining each team's responsibilities and the precise time at which each sprint's work is expected to be finished should be developed.
- **Step** 3: Teams begin building functional software in the third step, iteration/construction, based on requirements and ongoing feedback. Iterations, also known as single development cycles, are the foundation of the Agile software development cycle.

Advantages Agile Software Development

- Deployment of software is quicker and thus helps in increasing the trust of the customer.
- Can better adapt to rapidly changing requirements and respond faster.
- Helps in getting immediate feedback which can be used to improve the software in the next increment.
- People – Not Process. People and interactions are given a higher priority than processes and tools.
- Continuous attention to technical excellence and good design.
- **Increased collaboration and communication:** Agile Software Development Methodologyemphasize collaboration and communication among team members, stakeholders, and customers. This leads to improved understanding, better alignment, and increased buy-in from everyone involved.
- **Flexibility and adaptability:** Agile methodologies are designed to be flexible and adaptable, making it easier to respond to changes in requirements, priorities, or market conditions. This allows teams to quickly adjust their approach and stay focused on delivering value.
- **Improved quality and reliability:** Agile methodologies place a strong emphasis on testing, quality assurance, and continuous improvement. This helps to ensure that software is delivered with high quality and reliability, reducing the risk of defects or issues that can impact the user experience.
- **Enhanced customer satisfaction:** Agile methodologies prioritize customer satisfaction and focus on delivering value to the customer. By involving customers throughout the development process, teams can ensure that the software meets their needs and expectations.
- **Increased team morale and motivation:** Agile methodologies promote a collaborative, supportive, and positive work environment. This can lead to increased team morale, motivation, and engagement, which can in turn lead to better productivity, higher quality work, and improved outcomes.

Disadvantages Agile Software Development

- In the case of large software projects, it is difficult to assess the effort required at the initial stages of the software development life cycle.
- Agile Development is more code-focused and produces less documentation.
- Agile development is heavily dependent on the inputs of the customer. If the customer has ambiguity in his vision of the outcome, it is highly likely that the project to get off track.
- Face-to-face communication is harder in large-scale organizations.
- Only senior programmers are capable of making the kind of decisions required during the development process. Hence, it's a difficult situation for new programmers to adapt to the environment.
- **Lack of predictability:** Agile Development relies heavily on customer feedback and continuous iteration, which can make it difficult to predict project outcomes, timelines, and budgets.
- Limited scope control: Agile Development is designed to be flexible and adaptable, which means that scope changes can be easily accommodated. However, this can also lead to scope creep and a lack of control over the project scope.
- **Lack of emphasis on testing:** Agile Development places a greater emphasis on delivering working code quickly, which can lead to a lack of focus on testing and quality assurance. This can result in bugs and other issues that may go undetected until later stages of the project.
- **Risk of team burnout:** Agile Development can be intense and fast-paced, with frequent sprints and deadlines. This can put a lot of pressure on team members and lead to burnout, especially if the team is not given adequate time for rest and recovery.
- **Lack of structure and governance:** Agile Development is often less formal and structured than other development methodologies, which can lead to a lack of governance and oversight. This can result in inconsistent processes and practices, which can impact project quality and outcomes.

Practices of Agile Software Development

- **Scrum:** Scrum is a framework for agile software development that involves iterative cycles called sprints, daily stand-up meetings, and a product backlog that is prioritized by the customer.
- **Kanban:** Kanban is a visual system that helps teams manage their work and improve their processes. It involves using a board with columns to represent different stages of the development process, and cards or sticky notes to represent work items.
- **Continuous Integration:** Continuous Integration is the practice of frequently merging code changes into a shared repository, which helps to identify and resolve conflicts early in the development process.
- **Test-Driven Development:** Test-Driven Development (TDD) is a development practice that involves writing automated tests before writing the code. This helps to ensure that the code meets the requirements and reduces the likelihood of defects.
- P**air Programming:** Pair programming involves two developers working together on the same code. This helps to improve code quality, share knowledge, and reduce the likelihood of defects.

Advantages of Agile Software Development over traditional software development approaches

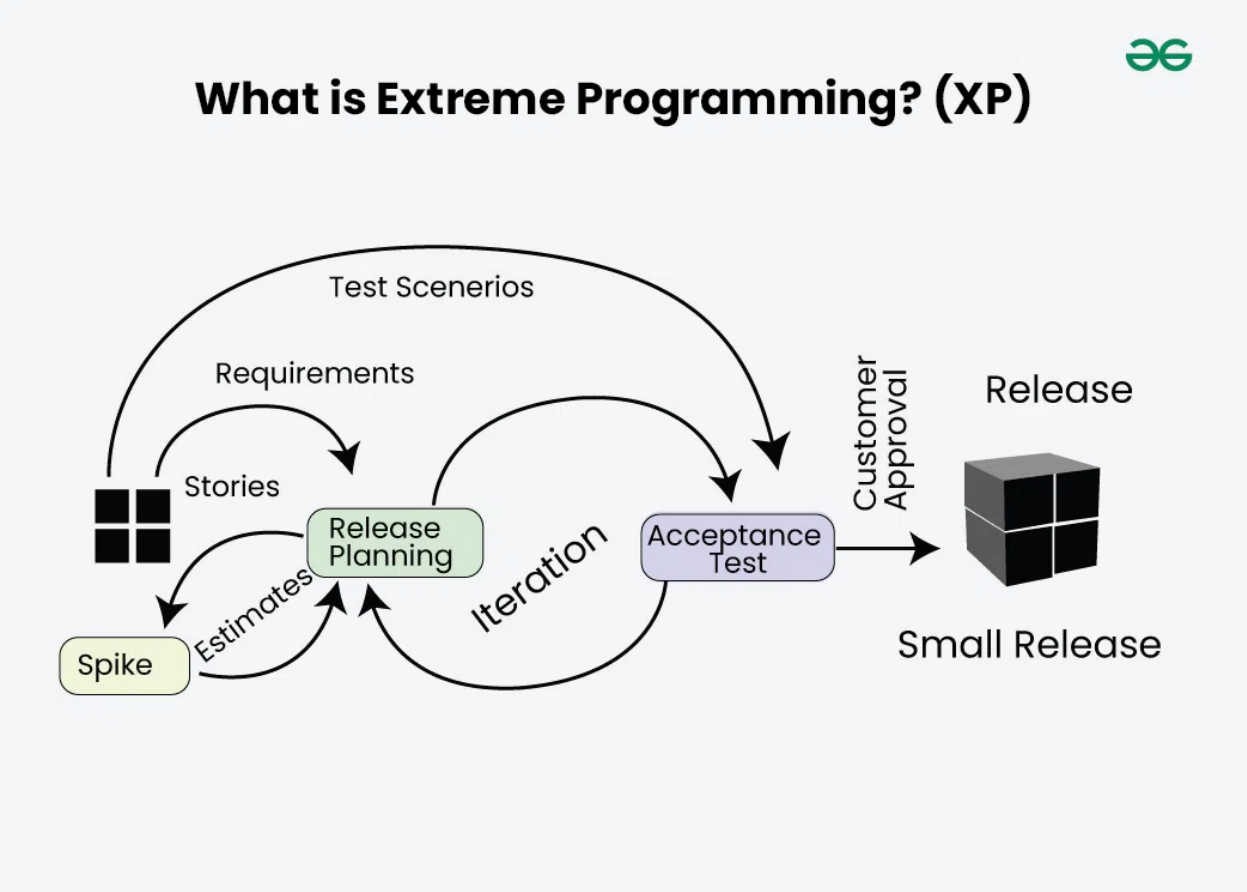
1. **Increased customer satisfaction:** Agile development involves close collaboration with the customer, which helps to ensure that the software meets their needs and expectations.
2. **Faster time-to-market:** Agile development emphasizes the delivery of working software in short iterations, which helps to get the software to market faster.
3. **Reduced risk:** Agile development involves frequent testing and feedback, which helps to identify and resolve issues early in the development process.
4. **Improved team collaboration:** Agile development emphasizes collaboration and communication between team members, which helps to improve productivity and morale.
5. **Adaptability to change:** Agile Development is designed to be flexible and adaptable, which means that changes to the project scope, requirements, and timeline can be accommodated easily. This can help the team to respond quickly to changing business needs and market demands.
6. **Better quality software:** Agile Development emphasizes continuous testing and feedback, which helps to identify and resolve issues early in the development process. This can lead to higher-quality software that is more reliable and less prone to errors.
7. **Increased transparency:** Agile Development involves frequent communication and collaboration between the team and the customer, which helps to improve transparency and visibility into the project status and progress. This can help to build trust and confidence with the customer and other stakeholders.
8. **Higher productivity:** Agile Development emphasizes teamwork and collaboration, which helps to improve productivity and reduce waste. This can lead to faster delivery of working software with fewer defects and rework.
9. **Improved project control:** Agile Development emphasizes continuous monitoring and measurement of project metrics, which helps to improve project control and decision-making. This can help the team to stay on track and make data-driven decisions throughout the development process.

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

2.3. Extreme Programming [XP]:

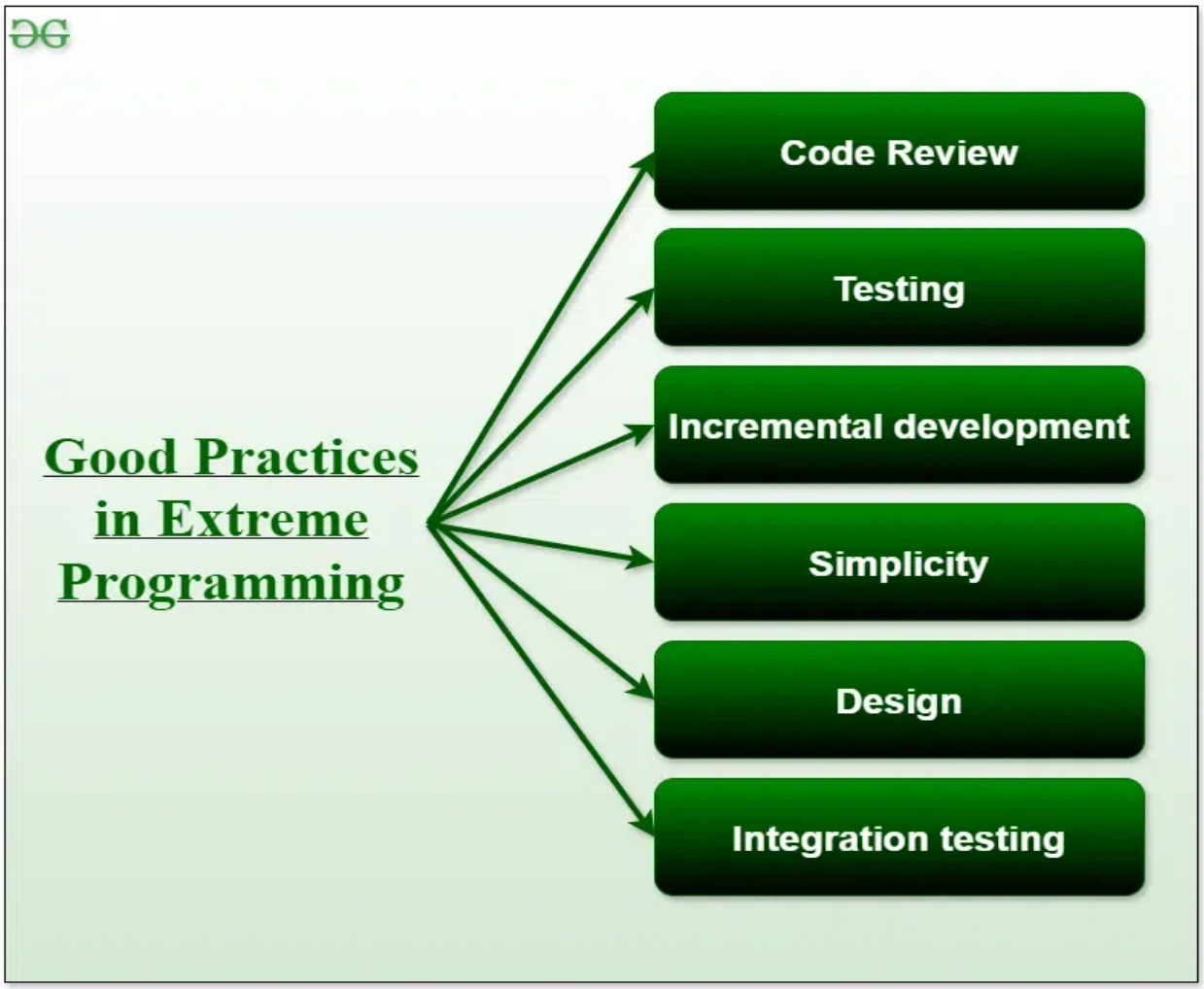
- Extreme Programming (XP) is an agile software developemt methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.
- XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with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



Extreme Programming (XP)

- Agile development approaches evolved in the 1990s as a reaction to documentation and bureaucracy-based processes, particularly the waterfall approach.

Good Practices in Extreme Programming



Extreme Programming Good Practices

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good-quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** Integration Testing helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic Principles of Extreme programming

Some of the basic activities that are followed during software development by using the XP model are given below:

- **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
- **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.
- **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
- **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.
- **Pair Programming:** XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.
- **Continuous Integration:** In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.
- **Refactoring:** XP encourages refactoring, which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized, and easy to understand.
- **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.
- **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.
- **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer's needs are understood and met, and also facilitates communication and feedback.

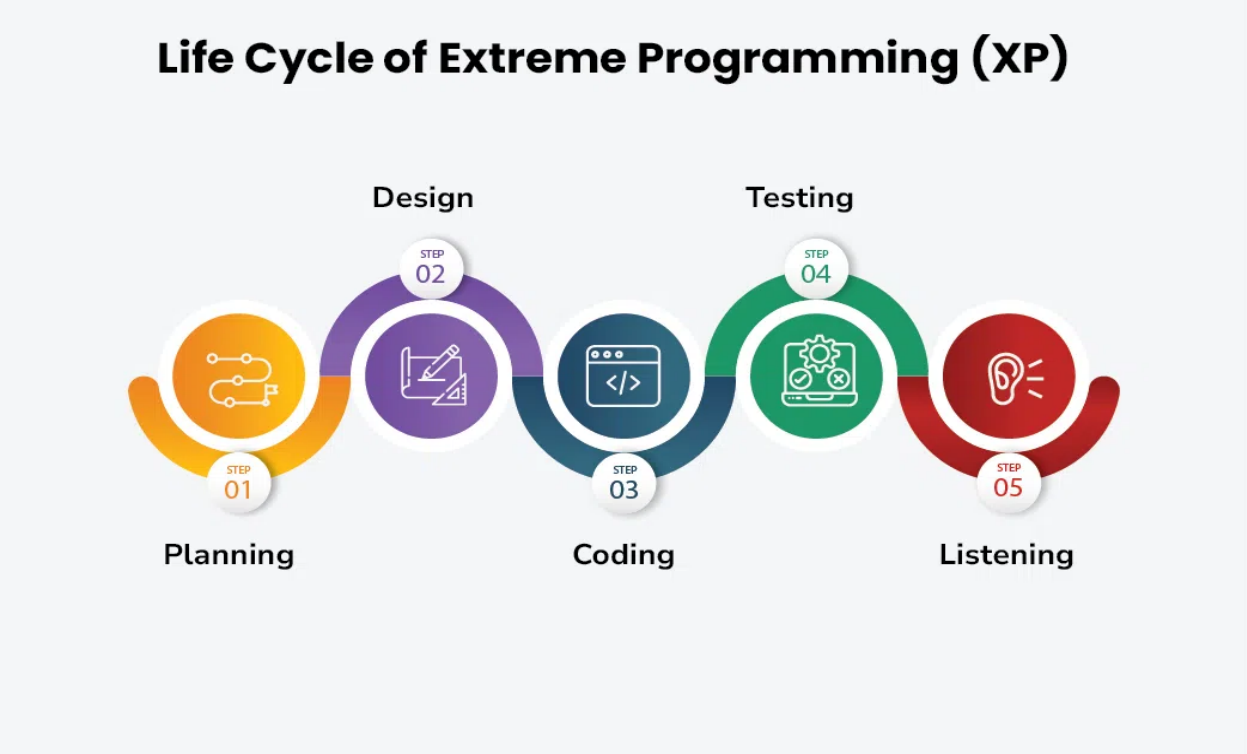
Applications of Extreme Programming (XP)

Some of the projects that are suitable to develop using the XP model are given below:

- **Small projects:** The XP model is very useful in small projects consisting of small teams as face-to-face meeting is easier to achieve.
- **Projects involving new technology or Research projects:** This type of project faces changing requirements rapidly and technical problems. So XP model is used to complete this type of project.
- **Web development projects:** The XP model is well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
- **Collaborative projects:** The XP model is useful for collaborative projects that require close collaboration between the development team and the customer.
- **Projects with tight deadlines:** The XP model can be used in projects that have a tight deadline, as it emphasizes simplicity and iterative development.
- **Projects with rapidly changing requirements: The** XP model is designed to handle rapidly changing requirements, making it suitable for projects where requirements may change frequently.
- **Projects where quality is a high priority:** The XP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority.

Life Cycle of Extreme Programming (XP)

The Extreme Programming Life Cycle consist of five phases:

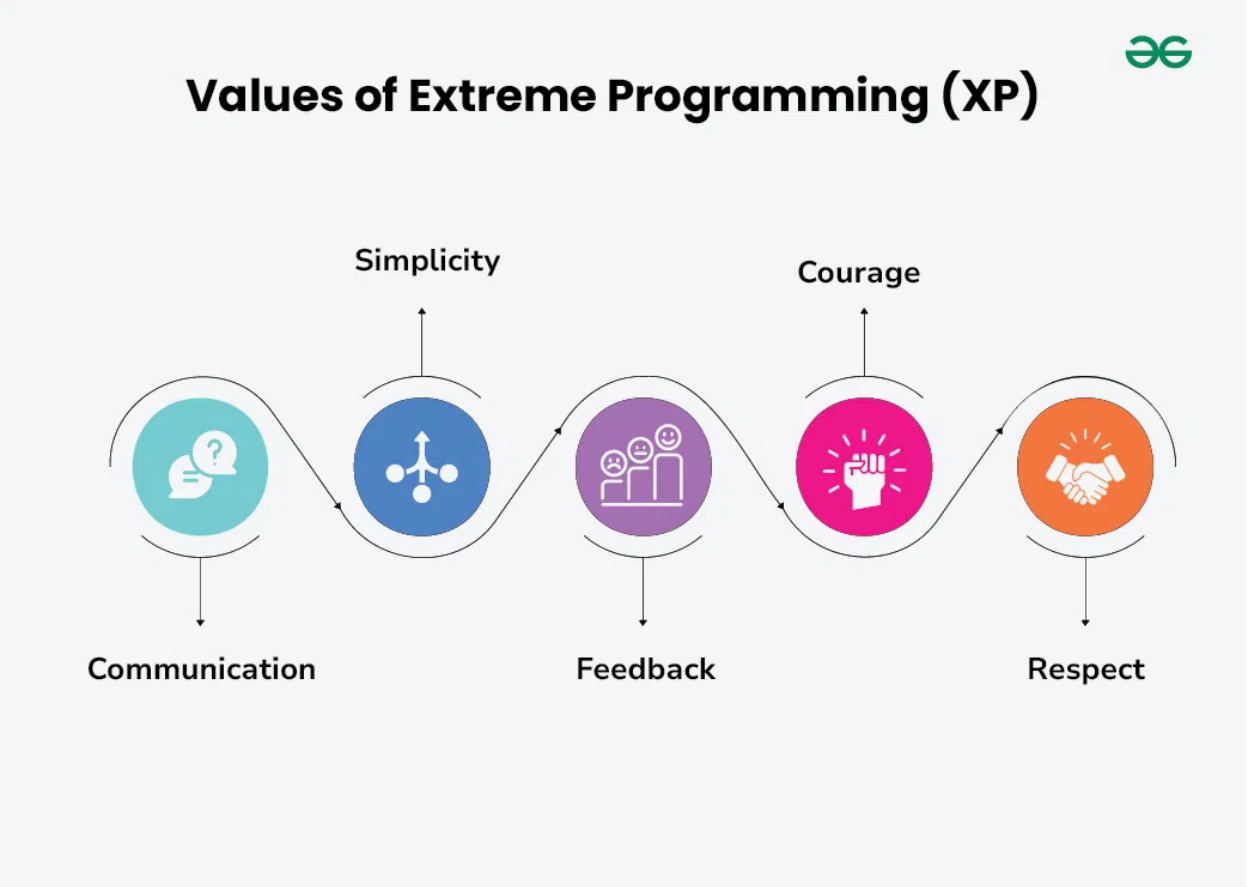


Life Cycle of Extreme Programming (XP)

1. **Planning:** The first stage of Extreme Programming is planning. During this phase, clients define their needs in concise descriptions known as user stories. The team calculates the effort required for each story and schedules releases according to priority and effort.
2. **Design:** The team creates only the essential design needed for current user stories, using a common analogy or story to help everyone understand the overall system architecture and keep the design straightforward and clear.
3. **Coding:** Extreme Programming (XP) promotes pair programming i.e**.** wo developers work together at one workstation, enhancing code quality and knowledge sharing. They write tests before coding to ensure functionality from the start (TDD), and frequently integrate their code into a shared repository with automated tests to catch issues early.
4. **Testing:** Extreme Programming (XP) gives more importance to testing that consist of both unit tests and acceptance test**.** Unit tests, which are automated, check if specific features work correctly. Acceptance tests, conducted by customers, ensure that the overall system meets initial requirements. This continuous testing ensures the software's quality and alignment with customer needs.
5. **Listening:** In the listening phase regular feedback from customers to ensure the product meets their needs and to adapt to any changes.

Values of Extreme Programming (XP)

There are five core values of Extreme Programming (XP)



Values of Extreme Programming (XP)

1. **Communication:** The essence of communication is for information and ideas to be exchanged amongst development team members so that everyone has an understanding of the system requirements and goals. Extreme Programming (XP) supports this by allowing open and frequent communication between members of a team.
2. **Simplicity:** Keeping things as simple as possible helps reduce complexity and makes it easier to understand and maintain the code.
3. **Feedback:** Feedback loops which are constant are among testing as well as customer involvements which helps in detecting problems earlier during development.
4. **Courage:** Team members are encouraged to take risks, speak up about problems, and adapt to change without fear of repercussions.
5. **Respect**: Every member's input or opinion is appreciated which promotes a collective way of working among people who are supportive within a certain group.

Advantages of Extreme Programming (XP)

- **Slipped schedules:** Timely delivery is ensured through slipping timetables and doable development cycles.
- **Misunderstanding the business and/or domain** − Constant contact and explanations are ensured by including the client on the team.
- **Cancelled projects:** Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.
- **Staff turnover:** Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.
- **Costs incurred in changes:** Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.
- **Business changes:** Changes are accepted at any moment since they are seen to be inevitable.
- **Production and post-delivery defects:** the unit tests to find and repair bugs as soon as possible.

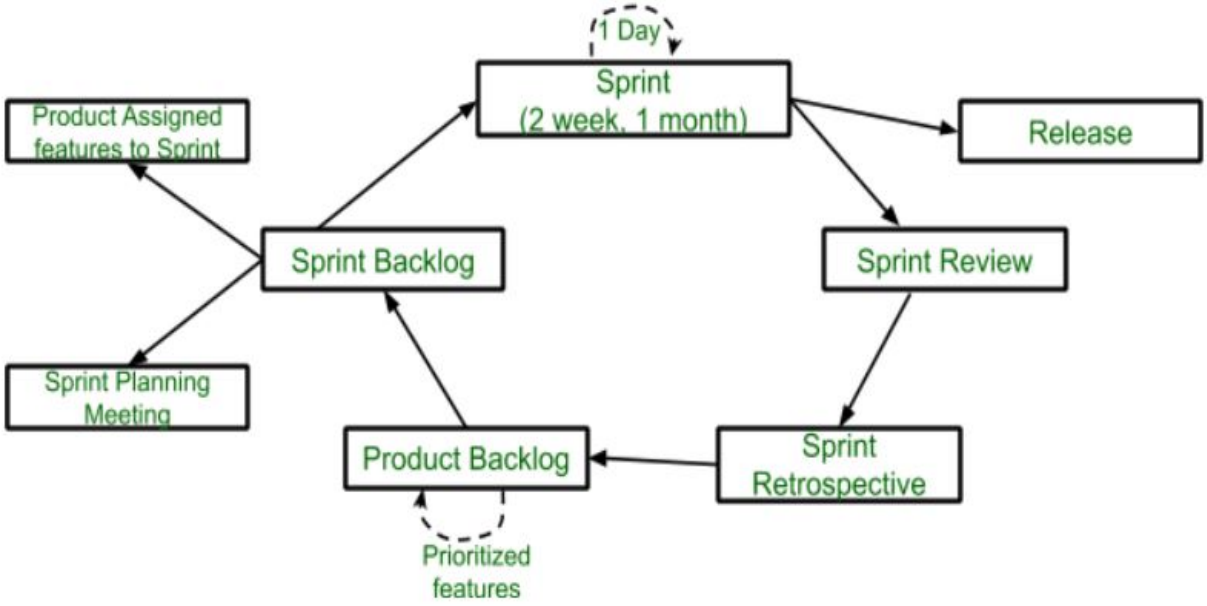
2.4. Scrum:

- Scrum is a management framework that teams use to self-organize tasks and work towards a common goal.
- It is a framework within which people can address complex adaptive problems while the productivity and creativity of delivering products are at the highest possible value.
- Scrum allows us to develop products of the highest value while making sure that we maintain creativity and productivity.
- The iterative and incremental approach used in scrum allows the teams to adapt to the changing requirements.

Silent features of Scrum

- Scrum is a light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework helps the team to work together

Lifecycle of Scrum



- **Sprint:** A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint. **Release:** When the product is completed, it goes to the Release stage.
- **Sprint Review:** If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage.
- **Sprint Retrospective**: In this stage quality or status of the product is checked.
- **Product Backlog:** According to the prioritize features the product is organized.
- **Sprint Backlog:** Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

Advantage of Scrum framework

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It's like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of Scrum framework

- Scrum framework do not allow changes into their sprint.
- Scrum framework is not fully described model. If you want to adopt it you need to fill in the framework with your own details like XP, DSDM, Kanban.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

2.5. Agile Project Management and Scaling Agile Methods:

1. Agile Project Management:

- Agile project management is an iterative and flexible approach to managing projects, initially popularized in software development but now used across various industries.
- Agile emphasizes delivering value quickly, adapting to changes, and continuously improving through collaboration among cross-functional teams.
- Instead of rigid plans, Agile relies on iterative cycles (often called "sprints") and incremental deliveries, allowing teams to adapt to changing requirements and feedback rapidly.

Core Principles of Agile Project Management

The Agile approach is built on principles outlined in the Agile Manifesto, which emphasizes:

1. **Individuals and interactions** over processes and tools.
2. **Working software** (or product) over comprehensive documentation.
3. **Customer collaboration** over contract negotiation.
4. **Responding to change** over following a fixed plan.

Benefits of Agile Project Management:

- **Faster time to market:** Agile enables rapid delivery of value.
- **Improved customer satisfaction:** Agile fosters collaboration and responsiveness to customer needs.
- **Increased flexibility:** Agile allows for adapting to changing requirements.
- **Higher quality:** Agile emphasizes continuous improvement and testing.
- **Enhanced team morale:** Agile empowers teams and promotes ownership.

Challenges of Agile:

- **Cultural shift:** Adopting Agile requires a change in mindset and organizational culture.
- **Skill development:** Teams need training and coaching to effectively implement Agile practices.
- **Measuring success:** Traditional metrics may not be suitable for Agile projects.
- **Managing dependencies:** In large-scale projects, coordinating dependencies can be complex.

2. Scaling Agile Methods:

- Scaling Agile becomes crucial when applying Agile practices across large organizations or projects involving multiple teams.
- Traditional Agile frameworks work best with small teams, but large enterprises need coordination and alignment. Some popular scaling frameworks include:

1. **SAFe (Scaled Agile Framework):** SAFe provides a structured approach to scaling Agile across large organizations, with multiple layers like Team, Program, and Portfolio. It introduces Program Increments (PIs) to align multiple teams.
2. **LeSS (Large Scale Scrum):** LeSS scales Scrum for multiple teams by using one Product Backlog across multiple teams, each working in parallel while maintaining a single Scrum Master and a shared Product Owner role.
3. **Spotify Model:** Developed by Spotify, this model organizes people into "squads" (similar to Scrum teams) and groups them into "tribes" with aligned goals. This model emphasizes autonomy and alignment, allowing teams to self-organize.
4. **Disciplined Agile Delivery (DAD):** DAD provides a decision-making framework that helps teams choose between various Agile and Lean techniques based on the project needs. It emphasizes a holistic view, considering areas like architecture and governance.

2.6. Pros and Cons of Process Models and their Applicability:

- Process models are essential tools in project management, offering structured approaches to software development and system design.
- However, their effectiveness depends on their appropriate application and understanding of their inherent strengths and weaknesses.

Common Process Models

- **Waterfall Model:** A linear, sequential approach, well-suited for projects with well-defined requirements.
- **Agile Model:** An iterative and incremental approach, ideal for projects with evolving requirements.
- **Spiral Model:** A risk-driven approach, balancing iterative development with systematic risk management.
- **V-Model:** A software development lifecycle model that emphasizes verification and validation at each stage.

Pros and Cons of Process Models

Pros:

- **Structure and Organization:** Models provide a clear framework for project planning, execution, and monitoring.
- **Risk Management:** They help identify and mitigate potential risks early in the project lifecycle.
- **Quality Assurance:** Models promote adherence to quality standards and best practices.
- **Team Collaboration:** They facilitate effective communication and coordination among team members.
- **Customer Satisfaction:** By following a structured approach, project teams can deliver products that meet customer expectations.

Cons:

- **Rigidity:** Some models, like the Waterfall model, can be inflexible and resistant to change.
- **Overemphasis on Documentation:** Excessive documentation can be time-consuming and may not always add value.
- **Limited Adaptability:** Traditional models may struggle to accommodate evolving requirements and unforeseen challenges.
- **Complexity:** Some models can be overly complex and difficult to understand and implement.

Applicability of Process Models:

The choice of a process model depends on various factors, including:

- **Project Size and Complexity:** Larger, more complex projects may benefit from a structured approach like Waterfall or V-Model.
- **Requirement Certainty:** Projects with well-defined requirements are better suited for linear models like Waterfall.
- **Customer Involvement:** Agile models are ideal for projects with high customer involvement and frequent feedback.
- **Risk Tolerance:** Risk-averse projects may benefit from a risk-driven approach like the Spiral model.
- **Team Experience and Maturity:** Experienced teams may be able to adapt to more flexible models like Agile.

Hybrid Approach

- In practice, a hybrid approach that combines elements of different models can often be the most effective.
- This allows teams to tailor their processes to specific project needs, balancing structure and flexibility.