Unit I

Product and Process: The Nature of Software – The changing nature of Software – The Software Process – Process models –The Waterfall Model– Incremental Process Model- Evolutionary Process Models–The Unified Process- Agile Development – Extreme Programming (XP)- Adaptive Software Development(ASD)- Scrum-Crystal-Feature Driven Development(FDD)-Agile Modeling.

NATURE OF SOFTWARE:

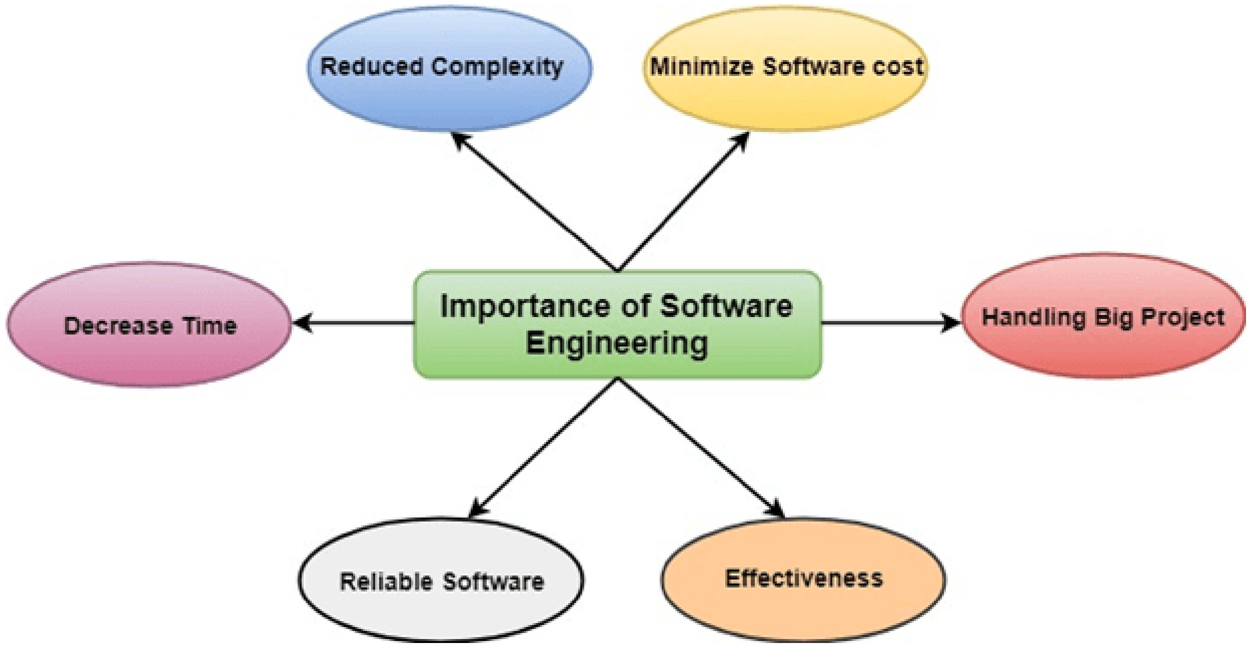
Software is:
(1) instructions (computer programs) that when executed provide desired features, function, and performance;
(2) data structures that enable the programs to adequately manipulate information, and
(3) document that describes the operation and use of the programs.
Characteristics of software

➢Software is developed or engineered, it is not manufactured in the classical sense.

➢Software does not wear out. However it deteriorates due to change.

➢Software is custom built rather than assembling existing components.

- Although the industry is moving towards component based construction, most software continues to be custom built



THE CHANGING NATURE OF SOFTWARE

• Seven Broad Categories of software are challenges for software engineers

System software-

System software is a collection of programs written to service other programs. System software is a collection of programs which are written to service other programs. Some system software processes complex but determinate, information structures.System software: such as compilers, editors, file management utilities.

Application software:
stand-alone programs for specific needs. Application software is defined as programs that solve a specific business need. This software are used to controls business needs. Ex: Transaction processing.
Embedded software –
This software resides within a system or product and it is used to implement and control features and functions from end user and system itself. This software performs limited function like keypad control for microwave oven.
Artificial intelligence software-
Artificial intelligence (AI) software makes use of nonnumeric algorithms to solve complex problems. Application within this area include robotics, pattern recognition, game playing.
Engineering and scientific software
-Engineering and scientific software have been characterized by "number crunching" algorithm. This software is used to facilitate the engineering function and task. however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications
Product-line software
Designed to provide a specific capability for use by many different customers, product line software can focus on the limited and esoteric marketplace or address the mass consumer market. focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management) .

WebApps (Web applications) It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business application

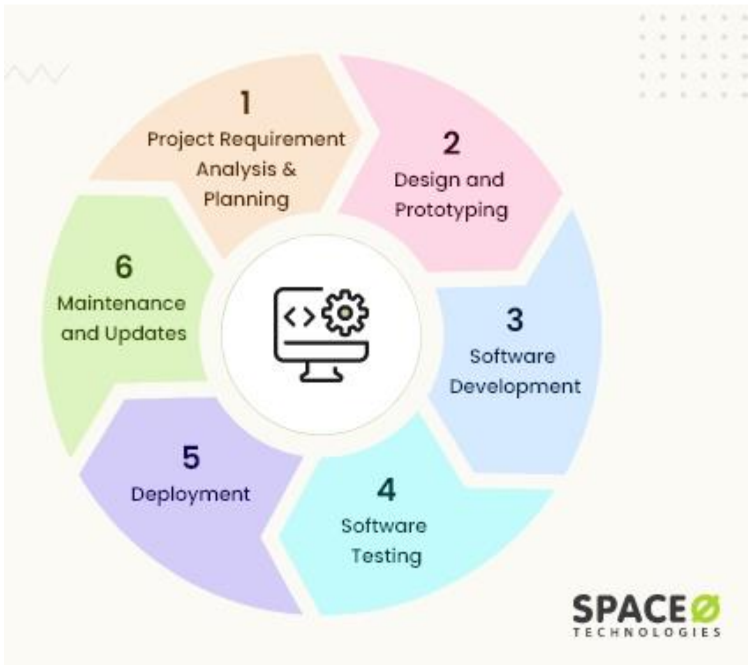
The Software Process

Software processes

Software processes in software engineering refer to the methods and techniques used to develop and maintain software. Some examples of software processes include:

- **Waterfall**: a linear, sequential approach to software development, with distinct phases such as requirements gathering, design, implementation, testing, and maintenance.
- **Agile**: a flexible, iterative approach to software development, with an emphasis on rapid prototyping and continuous delivery.
- **Scrum**: a popular Agile methodology that emphasizes teamwork, iterative development, and a flexible, adaptive approach to planning and management.
- **DevOps**: a set of practices that aims to improve collaboration and communication between development and operations teams, with an emphasis on automating the software delivery process.

Each process has its own set of advantages and disadvantages, and the choice of which one to use depends on the specific project and organization.



Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process.

Software Development : In this process, designing, programming, documenting, testing, and bug fixing is done.

Components of Software :
There are three components of the software: These are : Program, Documentation, and Operating Procedures.

1. **Program –**
   A computer program is a list of instructions that tell a computer what to do.
   Documentation –
   Source information about the product contained in design documents, detailed code comments, etc.
   Operating Procedures –
   Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.
2. **Code**: the instructions that a computer executes in order to perform a specific task or set of tasks.
3. **Data**: the information that the software uses or manipulates.
4. **User** interface: the means by which the user interacts with the software, such as buttons, menus, and text fields.
5. **Libraries**: pre-written code that can be reused by the software to perform common tasks.
6. **Documentation**: information that explains how to use and maintain the software, such as user manuals and technical guides.

7. **Test cases**: a set of inputs, execution conditions, and expected outputs that are used to test the software for correctness and reliability.
8. **Configuration files**: files that contain settings and parameters that are used to configure the software to run in a specific environment.
9. **Build and deployment** scripts: scripts or tools that are used to build, package, and deploy the software to different environments.
10. **Metadata**: information about the software, such as version numbers, authors, and copyright information.

There are four basic key process activities:

1. **Software Specifications –**
   In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
   Software Development –
   In this process, designing, programming, documenting, testing, and bug fixing is done.
   Software Validation –
   In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.
   Software Evolution –
   It is a process of developing software initially, then timely updating it for various reasons.

2. <u>The Software Process</u>

Types of Software Process Model
Software processes, methodologies and frameworks range from specific prescriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a “sponsor” or “maintenance” organization distributes an official set of documents that describe the process.
Software Process and Software Development Lifecycle Model
One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models. carried out. The most used, popular and important SDLC models are given below:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

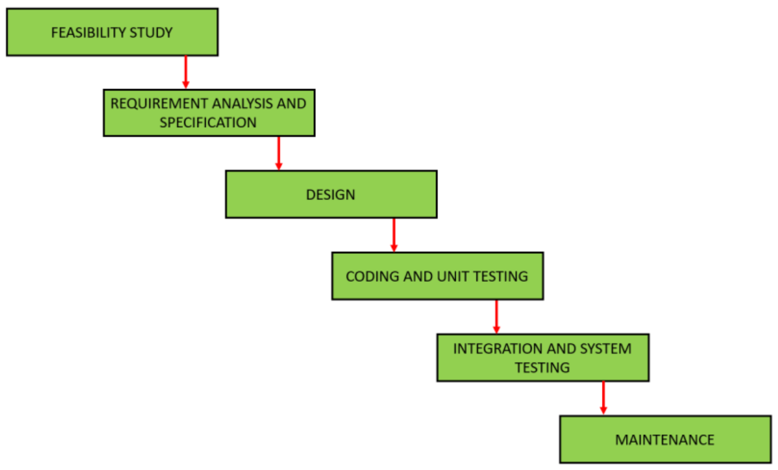
<u>Waterfall Model – Software Engineering</u>

The classical waterfall model is the basic <u>software development life cycle</u> model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. But it is very important because all the other software development life cycle models are based on the classical waterfall model.
Why Do We Use the Waterfall Model?

The waterfall model is a software development model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to project management and software development.
The waterfall model is useful in situations where the project requirements are well-defined and the project goals are clear. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome.

Features of the Waterfall Model

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


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

1. Feasibility Study

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

2. Requirements Analysis and Specification

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

- **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed.
- **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document.

3. Design

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

4. Coding and Unit Testing

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

5. Integration and System testing

Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps.

System testing consists of three different kinds of testing activities as described below.

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

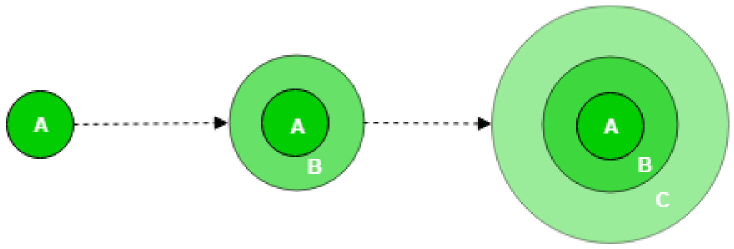
6. Maintenance

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

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

Incremental Process Model

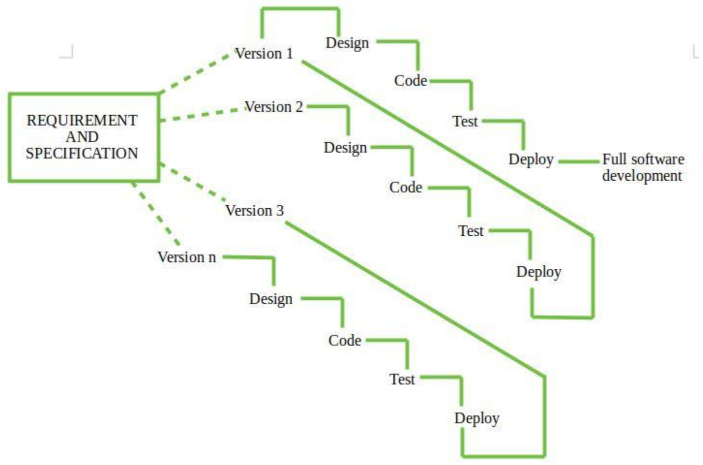
The incremental process model is also known as the Successive version model.
First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.



A, B, and C are modules of Software Products that are incrementally developed and delivered.

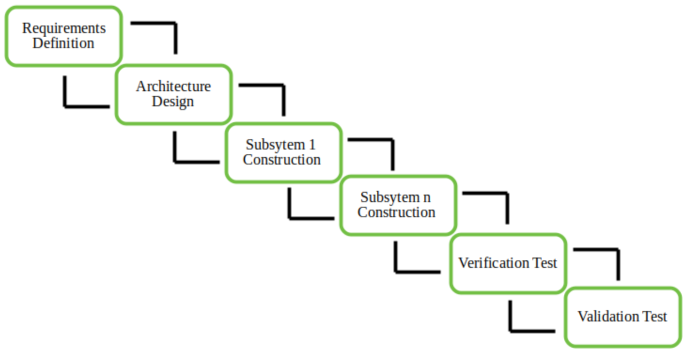
Life cycle activities:

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered.
Each incremental version is usually developed using an iterative waterfall model of development.
As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated into the next version.
Each version of the software has more additional features than the previous ones.



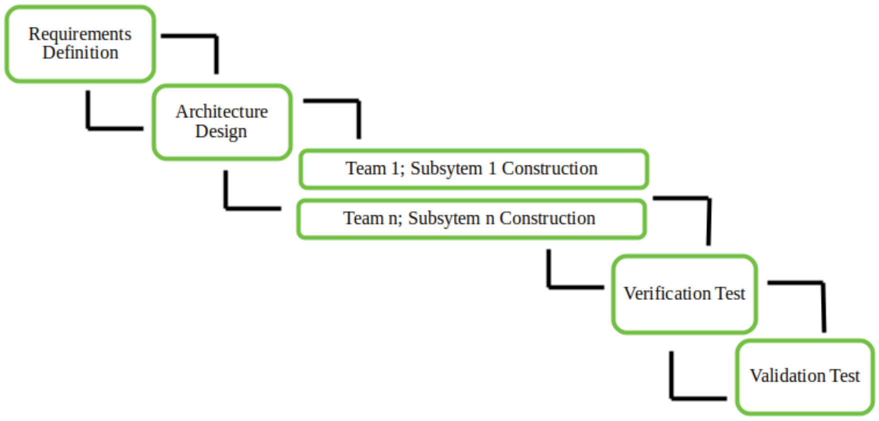
After Requirements gathering and specification, requirements are then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. After the last version (version n), it is now deployed at the client site.

Types of Incremental model:



1. Staged Delivery Model: Construction of only one part of the project at a time.

2. Parallel Development Model – Different subsystems are developed at the same time. It can decrease the calendar time needed for the development, i.e. TTM (Time to Market) if enough resources are available.



Evolutionary Process Models

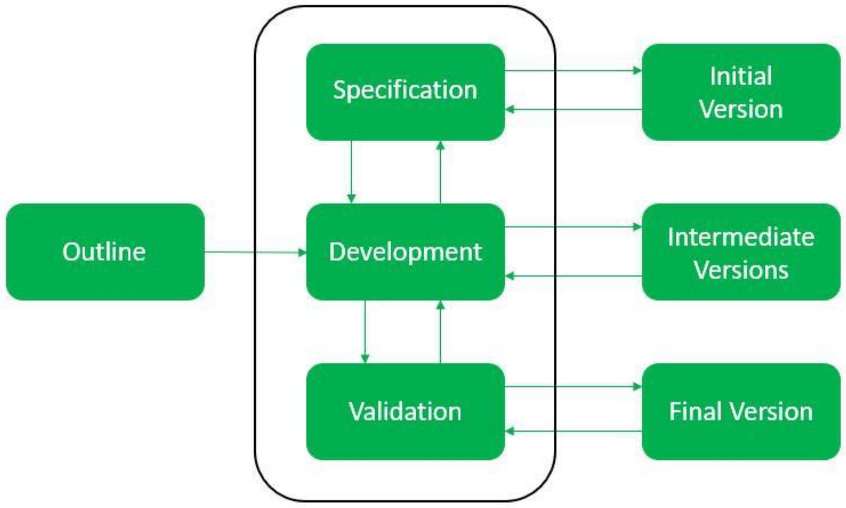
Evolutionary model is a combination of Iterative and Incremental model of software development life cycle. In this article, we are going to understand different types of evolutionary process models with the help of examples.

Software Process Model

A software process model is a structured representation of the activities of the software development process. The software process model includes various activities such as steps like planning, designing, implementation, defining tasks, setting up milestones, roles, and responsibilities, etc.

Evolutionary Process Model

The evolutionary model is based on the concept of making an initial product and then evolving the software product over time with iterative and incremental approaches with proper feedback. In this type of model, the product will go through several iterations and come up when the final product is built through multiple iterations. The development is carried out simultaneously with the feedback during the development.



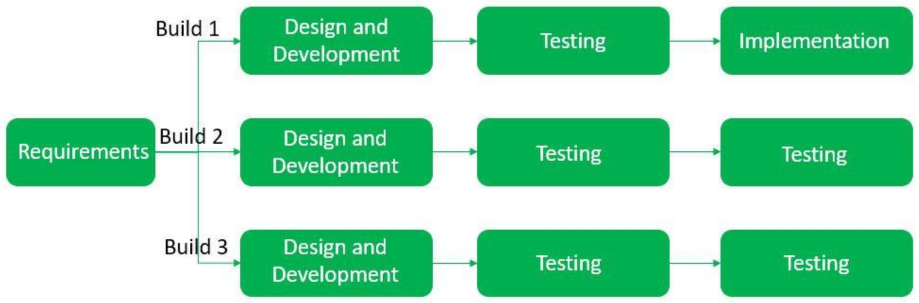
Evolutionary Model

Types of Evolutionary Process Models

1. Iterative Model
2. Incremental Model
3. Spiral Model

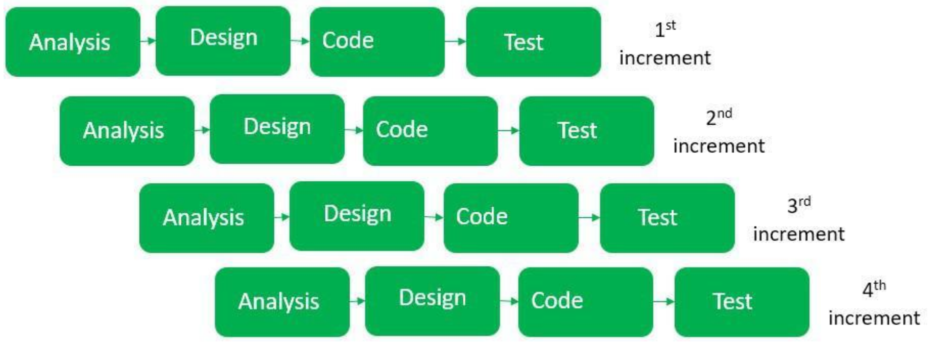
Iterative Model

In the iterative model first, we take the initial requirements then we enhance the product over multiple iterations until the final product gets ready. In every iteration, some design modifications were made and some changes in functional requirements is added. The main idea behind this approach is to build the final product through multiple iterations that result in the final product being almost the same as the user wants with fewer errors and the performance, and quality would be high.



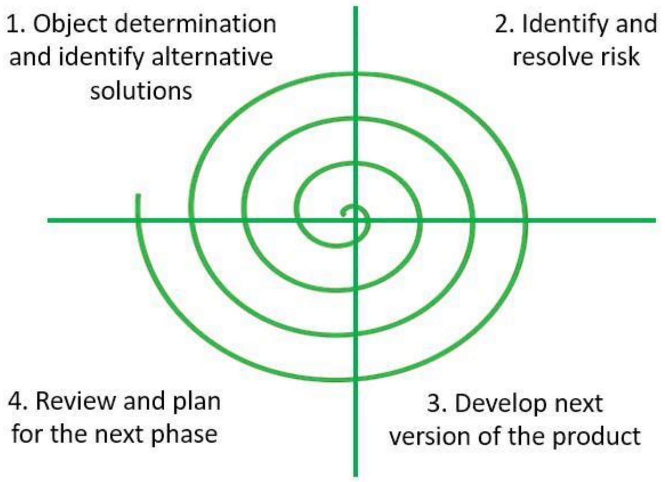
Iterative model

Incremental Model
In the incremental model, we first build the project with basic features and then evolve the project in every iteration, it is mainly used for large projects. The first step is to gather the requirements and then perform analysis, design, code, and test and this process goes the same over and over again until our final project is ready.



Incremental Model

Spiral Model
The spiral model is a combination of waterfall and iterative models and in this, we focused on risk handling along with developing the project with the incremental and iterative approach, producing the output quickly as well as it is good for big projects. The software is created through multiple iterations using a spiral approach. Later on, after successive development the final product will develop, and the customer interaction is there so the chances of error get reduced.

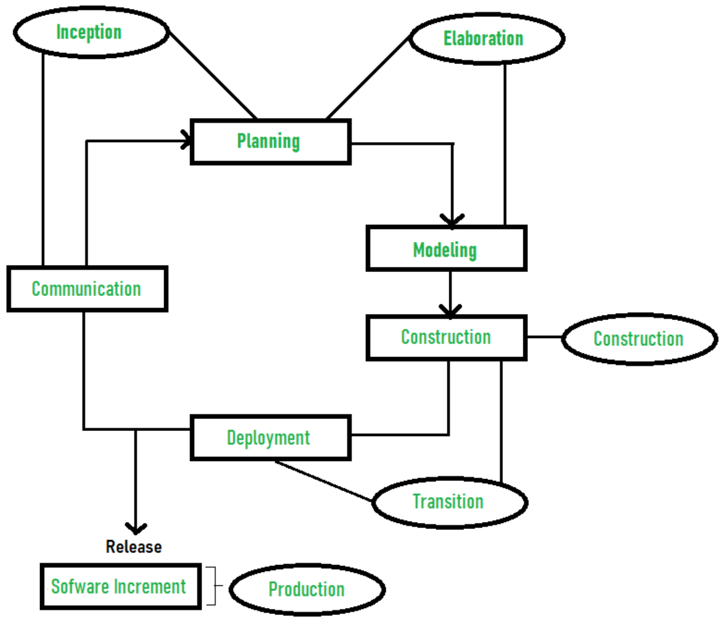

1. Object determination and identify alternative solutions
2. Identify and resolve risk
3. Develop next version of the product
4. Review and plan for the next phase

Spiral Model

The Unified Process

Rational Unified Process (RUP) is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process. RUP is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh. Some characteristics of RUP include use-case driven, Iterative (repetition of the process), and Incremental (increase in value) by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. RUP reduces unexpected development costs and prevents wastage of resources.

Phases of RUP: There is total of five phases of the life cycle of RUP:



1. **Inception –**
   - Communication and planning are the main ones.
   - Identifies the scope of the project using a use-case model allowing managers to estimate costs and time required.
   - Customers' requirements are identified and then it becomes easy to make a plan for the project.
   - The project plan, Project goal, risks, use-case model, and Project description, are made.
   - The project is checked against the milestone criteria and if it couldn't pass these criteria then the project can be either canceled or redesigned.
2. **Elaboration –**
   - Planning and modeling are the main ones.

- A detailed evaluation and development plan is carried out and diminishes the risks.
- Revise or redefine the use-case model (approx. 80%), business case, and risks.
- Again, checked against milestone criteria and if it couldn't pass these criteria then again project can be canceled or redesigned.
- Executable architecture baseline.

3. **Construction –**
    - The project is developed and completed.
    - System or source code is created and then testing is done.
    - Coding takes place.
4. **Transition –**
    - The final project is released to the public.
    - Transit the project from development into production.
    - Update project documentation.
    - Beta testing is conducted.
    - Defects are removed from the project based on feedback from the public.
5. **Production –**
    - The final phase of the model.
    - The project is maintained and updated accordingly.

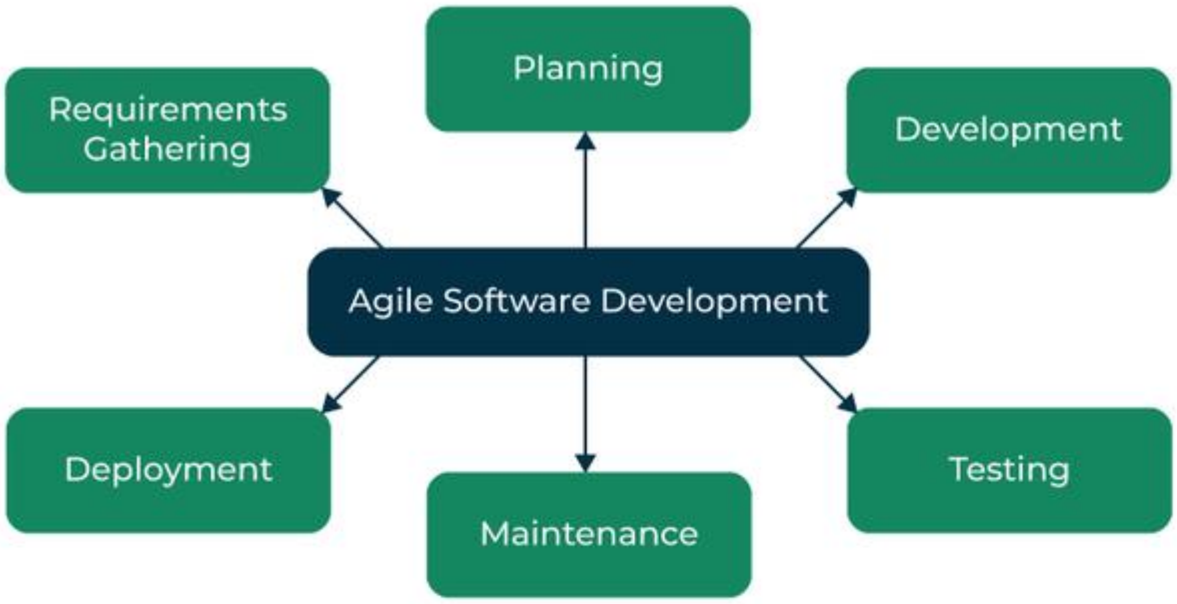
Agile Development

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change.

Principles of Agile:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. It welcomes changing requirements, even late in development.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shortest timescale.
4. Build projects around motivated individuals. Give them the environment and the support they need and trust them to get the job done.
5. Working software is the primary measure of progress.
6. Simplicity the art of maximizing the amount of work not done is essential.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
8. By the amount of work that has been finished, gauge your progress.
9. Never give up on excellence.
10. Take advantage of change to gain a competitive edge.

The Agile Software Development Process:



Agile Software Development

1. **<u>Requirements Gathering</u>:** 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.

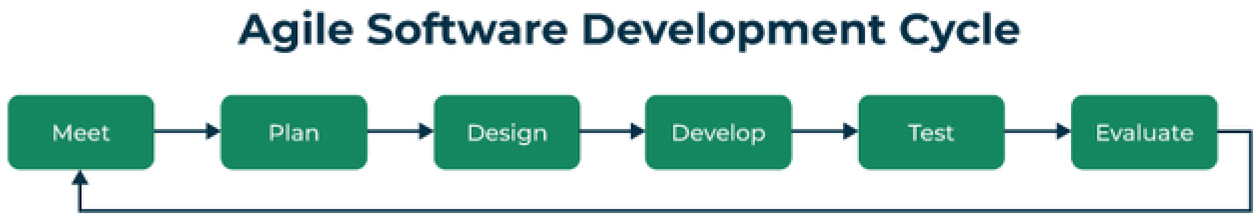
Agile Software Development is widely used by software development teams and is considered to be a flexible and adaptable approach to software development that is well-suited to changing requirements and the fast pace of software development.

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**



Agile software development cycle

- **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.

Extreme Programming

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements.

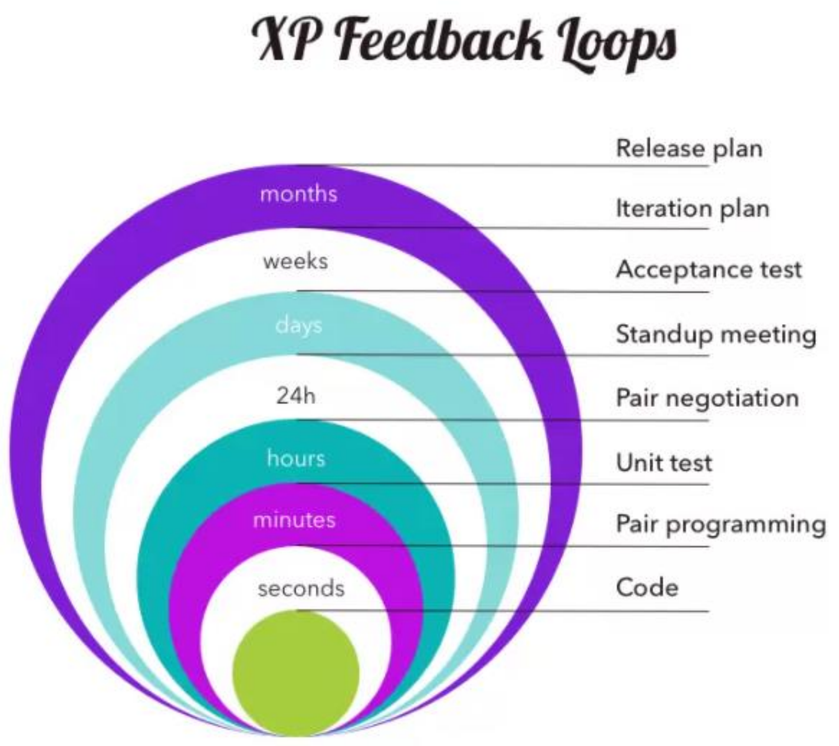
Good practices need to be practiced in extreme programming: Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **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:** It 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: XP is based on the frequent iteration through which the developers implement User Stories. It can be considered similar to a prototype. 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.

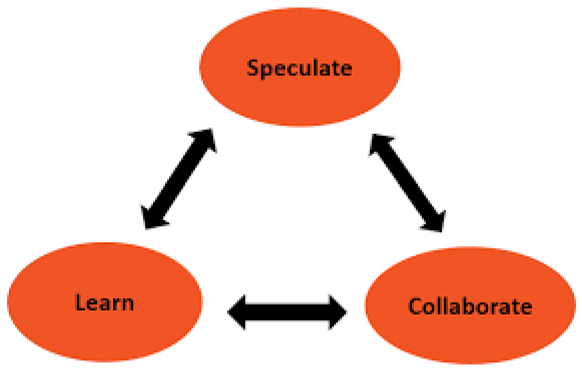


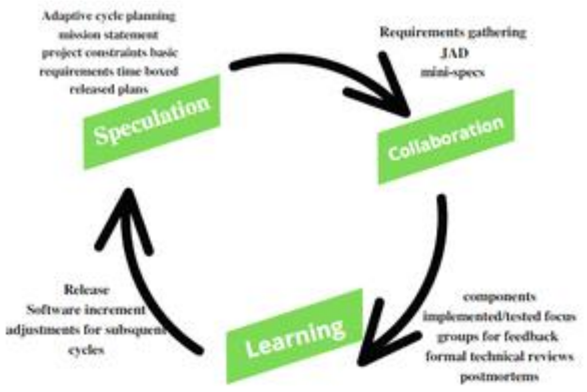
Applications of Extreme Programming (XP): Some of the projects that are suitable to develop using the XP model are given below:

- **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** bureaucracy-basedXP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority.

Adaptive Software Development(ASD)

Adaptive Software Development is a method to build complex software and system. ASD focuses on human collaboration and self-organization. ASD **"life cycle"** incorporates three phases namely:
1. Speculation
2. Collaboration
3. Learning
These are explained as following below.





Adaptive Software Development

1. Speculation:
During this phase project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs, customer mission statement, etc, to define set of release cycles that the project wants.
2. Collaboration:
It is the difficult part of ASD as it needs the workers to be motivated. It collaborates communication and teamwork but emphasizes individualism as individual creativity plays a major role in creative thinking. People working together must trust each others to

- Criticize without animosity,
- Assist without resentment,
- Work as hard as possible,
- Possession of skill set,
- Communicate problems to find effective solution.

3. Learning:
The workers may have a overestimate of their own understanding of the technology which may not lead to the desired result. Learning helps the workers to increase their level of understanding over the project.
Learning process is of 3 ways:

1. Focus groups
2. Technical reviews
3. Project postmortem

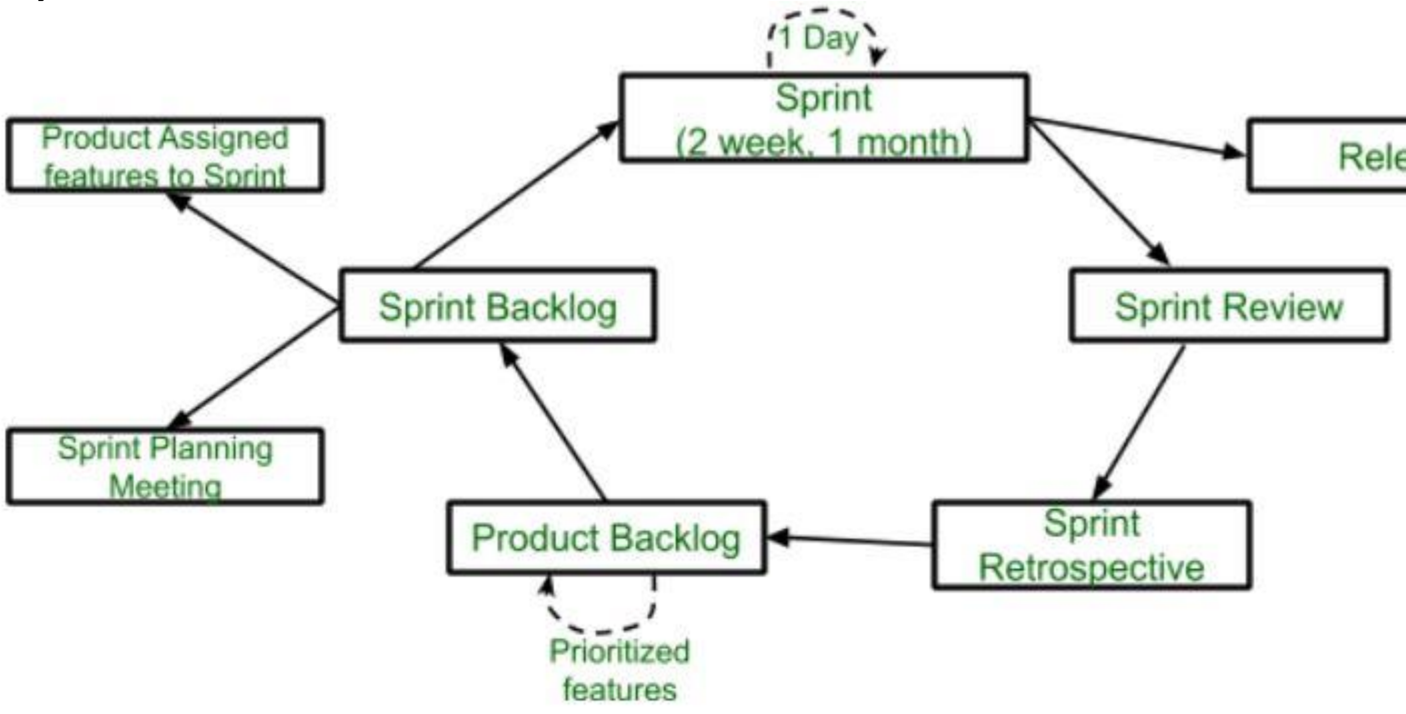
ASD's overall emphasis on the dynamics of self-organizing teams, interpersonal collaboration, and individual and team learning yield software project teams that have a much higher likelihood of success.

<u>Scrum</u>

Scrum is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process. Silent features of Scrum are:**

- Scrum is light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework help 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 using 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.

Crystal

Crystal methods in Agile Development/Framework: The crystal method is an agile framework that is considered a lightweight or agile methodology that focuses on individuals and their interactions. The methods are color-coded to significant risk to human life. It is mainly for short-term projects by a team of developers working out of a

single workspace. Among a few Agile Software Development Life Cycle (SDLC) models crystal is considered as one of the Agile SDLC models.
Two core beliefs of the Crystal method:

- Find your own way and methods to optimize workflow.
- Make use of unique methods to make the project unique and dynamic.

Let's know about the history of the Crystal Method: The crystal method was developed by an American scientist named Alistair Cockburn who worked at IBM. He decided not to focus on step-by-step developmental strategies, but to develop team collaboration and communication. Some of the traits of Cockburn's Crystal method were:

- Human-powered i.e. the project should be flexible and people involved in preferred work.
- Adaptive i.e. approaches don't have any fixed tools but can be changed anytime to meet the team's specific needs.
- Ultra-light i.e. this methodology doesn't require much documentation.

Properties of Crystal Agile Framework:

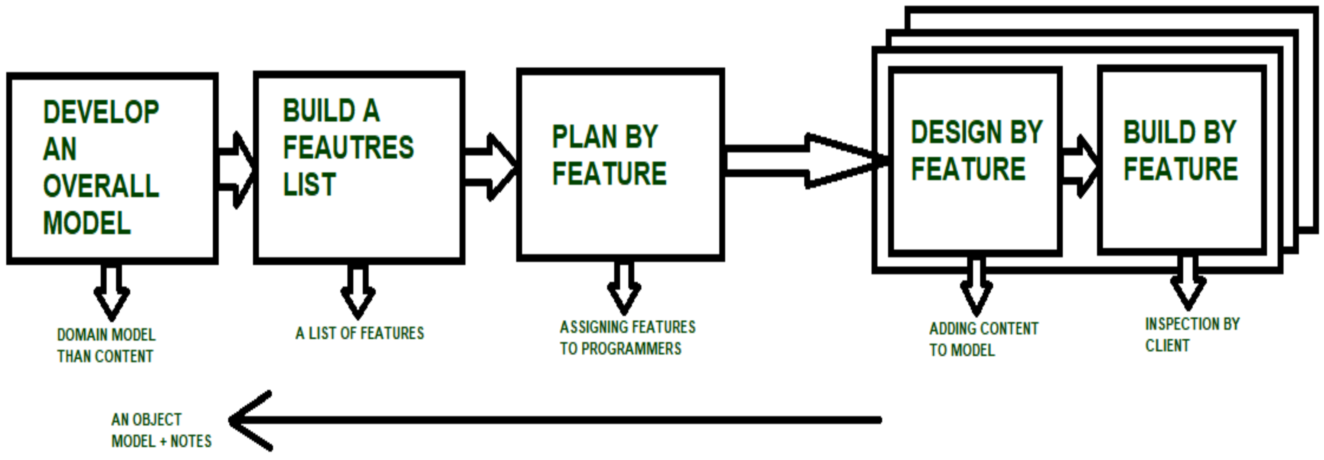
1. **Frequent Delivery-** It allows you regularly deliver the products and test code to real users. Without this, you might build a product that nobody needs.
2. **Reflective Improvement-** No matter how good you have done or how bad you have done. Since there are always areas where the product can be improved, so the teams can implement to improve their future practices.
3. **Osmotic Communication-** Alistair stated that having the teams in the same physical phase is very much important as it allows information to flow in between members of a team as in osmosis.
4. **Personal Safety-** There are no bad suggestions in a crystal team, team members should feel safe to discuss ideas openly without any fear.
5. **Focus-** Each member of the team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and works towards the same goal.
6. **Easy access to expert users-** It enhances team communication with users and gets regular feedback from real users.
7. **Technical tooling-** It contains very specific technical tools which to be used by the software development team during testing, management, and configuration. These tools make it enable the team to identify any error within less time.
8. **Continuous learning –** The framework emphasizes on continuous learning, enabling team members to acquire new skills and knowledge, and apply them in their work.
9. **Teamwork –** The framework stresses on the importance of teamwork, promoting collaboration, and mutual support among team members.
10. **Timeboxing –** The framework adopts timeboxing to manage project deadlines, ensuring that the team delivers within set timelines.
11. **Incremental development –** The framework promotes incremental development, enabling the team to deliver working software frequently, and adapt to changes as they arise.
12. **Automated testing –** The framework emphasizes on automated testing, enabling the team to detect and fix bugs early, reducing the cost of fixing errors at later stages.
13. **Customer involvement –** The framework emphasizes on involving customers in the development process, promoting customer satisfaction, and delivering products that meet their needs.

14. **Leadership** – The framework encourages leadership, enabling team members to take ownership of their work and make decisions that contribute to the success of the project.

<u>Feature Driven Development(FDD)</u>

*FDD **stands for** Feature-Driven Development**. It is an agile iterative and incremental model that focuses on progressing the features of the developing software. FDD Lifecycle***

- Build overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature



Characteristics of FDD

- **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.
- **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.
- **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.
- **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

Advantages of FDD

- Reporting at all levels leads to easier progress tracking.
- FDD provides continuous success for larger size of teams and projects.
- Reduction in risks is observed as whole model and design is build in smaller segments.
- FDD provides greater accuracy in cost estimation of the project due to feature segmentation.

Agile Modeling.

The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project. Also, anything that is a waste of time and effort is avoided.

The Agile Model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves.

Agile SDLC Models/Methods

- **Models:** Crystal Agile methodology places a strong emphasis on fostering effective communication and collaboration among team members, as well as taking into account the human elements that are crucial for a successful development process. This methodology is particularly beneficial for projects with a high degree of uncertainty, where requirements tend to change frequently.
- **Atern:** This methodology is tailored for projects with moderate to high uncertainty where requirements are prone to change frequently. Its clear-cut roles and responsibilities focus on delivering working software in short time frames. Governance practices set it apart and make it an effective approach for teams and projects.
- **Feature-driven development:** This approach is implemented by utilizing a series of techniques, like creating feature lists, conducting model evaluations, and implementing a design-by-feature method, to meet its goal. This methodology is particularly effective in ensuring that the end product is delivered on time and that it aligns with the requirements of the customer.
- **Scrum:** This methodology serves as a framework for tackling complex projects and ensuring their successful completion. It is led by a Scrum Master, who oversees the process, and a Product Owner, who establishes the priorities. The Development Team, accountable for delivering the software, is another key player.
- **Extreme programming (XP):** It uses specific practices like pair programming, continuous integration, and test-driven development to achieve these goals. Extreme programming is ideal for projects that have high levels of uncertainty and require frequent changes, as it allows for quick adaptation to new requirements and feedback.
- **Lean Development:** It is rooted in the principles of lean manufacturing and aims to streamline the process by identifying and removing unnecessary steps and activities. This is achieved through practices such as continuous improvement, visual management, and value stream mapping, which helps in identifying areas of improvement and implementing changes accordingly.
- **Unified Process:** Unified Process is a methodology that can be tailored to the specific needs of any given project. It combines elements of both waterfall and Agile methodologies, allowing for an iterative and incremental approach to development. This means that the UP is characterized by a series of iterations, each of which results in a working product increment, allowing for continuous improvement and the delivery of value to the customer.

All Agile methodologies discussed above share the same core values and principles, but they may differ in their implementation and specific practices. Agile development requires a high degree of collaboration and communication among team members, as well as a willingness to adapt to changing requirements and feedback from customers.

In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed. The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and can be completed within a couple of weeks only. At a time one iteration is planned, developed, and deployed to the customers. Long-term plans are not made.

Steps in the Agile Model

The agile model is a combination of iterative and incremental process models. The steps involve in agile SDLC models are:

- Requirement gathering
- Design the Requirements
- Construction / Iteration
- Testing / Quality Assurance
- Deployment

1. Requirement Gathering:- In this step, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.

2. Design the Requirements:- In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.

3. Construction / Iteration:- In this step, development team members start working on their project, which aims to deploy a working product.

4. Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:

5. Unit Testing:- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.

- **Integration Testing:-** Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.
- **System Testing:-** Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. Deployment:- In this step, the development team will deploy the working project to end users.

6. Feedback:- This is the last step of the **Agile Model.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

The time required to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. However, the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The Agile model's central principle is delivering an increment to the customer after each Time-box.