# CCS356 OBJECT ORIENTED SOFTWARE ENGINEERING

## Unit I Software Process and Agile Development

Introduction to Software Engineering, Software Process, Perspective and Specialized Process Models –Introduction to Agility - Agile process - Extreme programming - XP Process - Case Study.

---

### Introduction to Software Engineering

### Software engineering

Software engineering is the application of engineering principles to the design, development, and maintenance of software. It is a systematic approach to the development of software that ensures that the software is reliable, efficient, and meets the needs of the users.

### Types of software engineering

- **Front-End -** specializes in software that the client, customer, or user will see and use
- **Back-End -** focuses on system development and the underlying performance of the software
- **Full Stack -** deals with both front-end and back-end work
- **QA (Quality Assurance) -** tests software through development
- **DevOps (Development Operations) -** the go-between for operations and engineers, overseeing the development, maintenance, and implementation of software
- **Security** - tests security of the software and fixes any security flaws

### Software Engineering Process

The software engineering process is a systematic approach to the development of software. It involves the following steps:

- **Requirements gathering** - This step involves gathering information about the needs of the users.
- **Design** - This step involves creating a plan for how the software will be developed.
- **Implementation** - This step involves writing the code for the software.

- **Testing** - This step involves testing the software to ensure that it meets the requirements.
- **Deployment** - This step involves making the software available to users.
- **Maintenance** - This step involves fixing bugs and adding new features to the software.

**Software Engineering Challenges**

- ***Complexity*** - Software can be complex, making it difficult to design, develop, and test.
- ***Change*** **-** Software is constantly changing, making it difficult to keep up with the latest requirements.
- ***Quality*** **-** It can be difficult to ensure that software is reliable and meets the needs of the users.
- ***Security*** **-** Software can be vulnerable to security attacks, making it important to secure the software.

**Does All Software Require Software Engineering?**

- ***Not all software requires software engineering***. Simplistic games or programs that are used by consumers may not need engineering, depending on the risks associated with them.
- Almost all companies do require software engineering because of the high-risk information that they store and security risks that they pose.
- Software engineering helps to create customized, personalized software that should look into vulnerabilities and risks before they even emerge. Even when the software engineering principles of safety aren't required, it can also help to reduce costs and improve customer experience.

---

**Software Process**

- Software Processes is ***a coherent set of activities for specifying, designing, implementing and testing software systems.***
- A software process model is an ***abstraction of the software development process***. The models specify the stages and order of a process. It is a representation of the **order of activities** of the process and the **sequence** in which they are performed.

**A model will define the following:**

- The tasks to be performed

- The input and output of each task
- The pre and post-conditions for each task
- The flow and sequence of each task

**Factors that involved in software process**

- Project requirements
- Project size
- Project complexity
- Cost of delay
- Customer involvement
- Familiarity with technology
- Project resources

**Software Development Life Cycle (SDLC)**

**SDLC**

- The software development life cycle is a process of planning, creating, testing, and deploying information systems across hardware and software. It is a structured process that enables the production of high-quality, low-cost software, in the shortest possible production time.
- ***The goal of the SDLC*** is to produce superior software that meets and exceeds all customer expectations and demands.

**The importance of SDLC**

- It provides a standardized framework that defines activities and deliverables
- It aids in project planning, estimating, and scheduling
- It makes project tracking and control easier
- It increases the speed of development
- It improves client relations
- It decreases project risks, project management expenses and the overall cost of production.

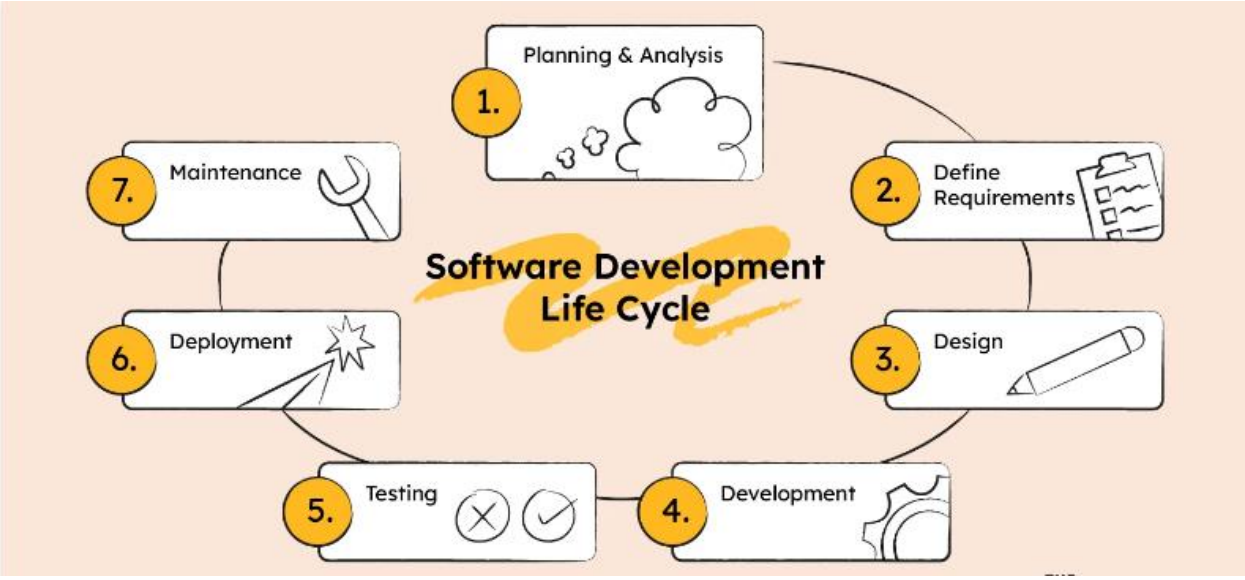
**The SDLC models are**

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model

- Iterative model
- Prototype model
- Spiral model

**Phases of the Software Development Life Cycle**

The software development life cycle ***can be an iterative process.***




**1. Planning & Analysis**

- The first phase of the SDLC is the project planning stage where you are gathering business requirements from client or stakeholders.
- This phase evaluate the feasibility of creating the product, the cost of production, the needs of the end-users, etc.
- To properly decide what to make, what not to make, and what to make first, a feature prioritization framework that takes into account the value of the software/update, the cost, the time it takes to build, and other factors.

**2. Define Requirements**

- This phase is critical for converting the information gathered during the planning and analysis phase into clear requirements for the development team.
- This process guides the development of several important documents:
  - ✓ A ***Software Requirement Specification (SRS)*** - describes what the software will do and how it will be expected to perform. It also describes the functionality the product needs to fulfill the needs of all stakeholders (business, users).

- ✓ A ***Use Case Document –*** a business document that provides a narrative of how a user will interact with the system to achieve a particular goal. It's a popular technique for gathering requirements from customers and business analysts. A well-written use case document can be used as the basis for design, development, and testing.
- ✓ A ***Requirement Traceability Matrix document -*** a document that maps and traces user requirement with test case. It also demonstrates the relationship between requirements and other artifacts. It's used to prove that requirements have been fulfilled.

**3. Design**

- ➢ In design phase, the plan and vision are elaborated into a software design document (SDD) that includes the system design, programming language, templates, platform to use, and application security measures.
- ➢ This is also where a flowchart can show how the software responds to user actions.
- ➢ The design phase will include the development of a prototype model. Creating a pre-production version of the product can give the team the to visualize the product.

**4. Development**

- ➢ The actual development phase is where the development team members divide the project into software modules and turn the software requirement into code that makes the product.
- ➢ This SDLC phase can take quite a lot of time and specialized development tools. It's important to have a set timeline and can keep track of the progress in this stage.
- ➢ In some cases, the development stage can also merge with the testing stage where certain tests are run to ensure there are no critical bugs.

**5. Testing**

- ➢ Before the software product release, it's important to have quality assurance, team perform validation testing to make sure it is functioning properly and does what it's meant to do.
- ➢ The testing process can find any major user experience issues and security issues.
- ➢ The types of testing to do in this phase:
  - ✓ **Performance testing -** Assesses the software's speed and scalability under different conditions
  - ✓ **Functional testing -** Verifies that the software meets the requirements
  - ✓ **Security testing -** Identifies potential vulnerabilities and weaknesses

- ✓ **Unit-testing -** Tests individual units or components of the software
- ✓ **Usability testing -** Evaluates the software's user interface and overall user experience
- ✓ **Acceptance testing -** This is the final testing stage to test if the software product delivers on what it promises

**6. Deployment**

- During deployment phase, the final product is delivered to the intended user.
- This process can automate and schedule the deployment depending on the type.

**7. Maintenance**

- The maintenance phase is the final stage of the SDLC, where maintenance is only a stage for further improvement.
- In the maintenance stage, users may find bugs and errors that were missed in the earlier testing phase. These bugs need to be fixed for better user experience and retention.
- In some cases, these can lead to going back to the first step of the software development life cycle.

---

**Perspective and Specialized Process Models**

**Perspective Process Models**

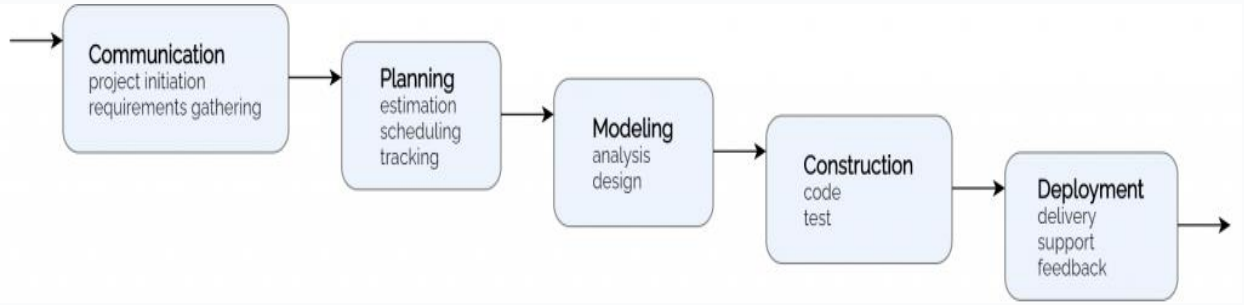
- A prescriptive process model strives for structure and order in software development. Activities and tasks occur sequentially for progress.
- It prescribes a set of process elements such as framework activities, software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project.
- Each process model prescribes a process flow (or a work flow), in which the process elements are interrelated to one another.

**Types of Perspective Process Models**

- Waterfall model
- Incremental Process Model
  - Incremental model
  - RAD Model
- Evolutionary Model
  - Prototype model
  - Spiral model
  - Concurrent Model

## 1. The Waterfall Model

- The Waterfall model is also known as '**Linear sequential model**' or '**Classic life cycle model**'.
- It is used in small projects where requirements are well defined and known before starting the project.
- Activities are carried out in a linear and systematic fashion.
- In this model, feedback is taken after each phase to ensure that the project is on the right path.




**5 Phases of Waterfall Model**

- ***Communication*** **-** The process starts with **communication**, where requirements are gathered from the customer and recorded.
- ***Planning*** **-** Then goes to the **planning** stage where the cost and time constraints are estimated, a schedule is outlined and project tracking variables are defined.
- ***Modelling*** **-** where a design based on the requirements and keeping the project constraints in mind is created.
- ***Construction*** **-** After this, code is generated and the actual building of the product is started in the **construction** phase. **Testing** (unit testing, integration testing) is done after code completion in this phase.
- ***Deployment*** **–** It is the last stage where the product is delivered, customer feedback is received and, support and maintenance for the product are provided.

**Advantages**

- ✓ A simple model to use and implement.
- ✓ Easily understandable workflow.
- ✓ Easy to manage since requirements are known prior to the start of the project.
- ✓ Can be applied to projects where quality is preferred over cost.

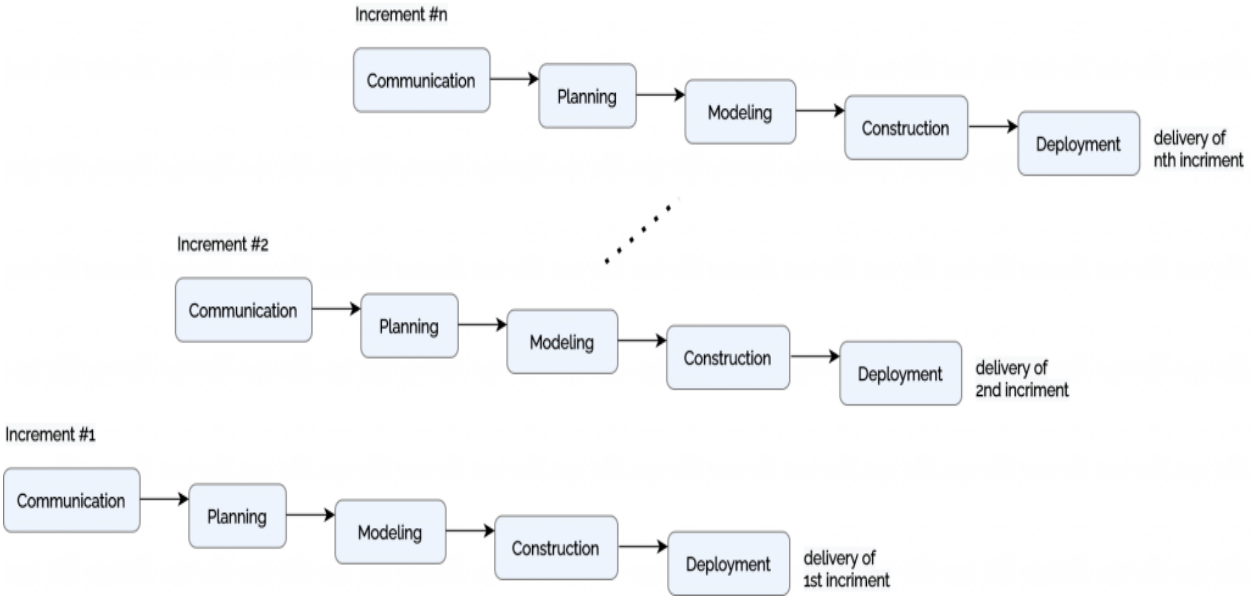
**Disadvantages**

- ✓ It may be difficult for the customer to provide all the specific requirements beforehand.

- ✓ Cannot be used for complex and object-oriented projects.
- ✓ Testing and customer evaluation are done at the last stages and hence the risk is high.
- ✓ Iteration of activities is not promoted which is unavoidable for certain projects.
- ✓ May lead to "blocking states" in which some project team members must wait for other members of the team to complete dependent tasks.

## 2. Incremental Process Model

- ➢ The Incremental process model is also known as '**Successive version model**'.
- ➢ In Incremental process model, a series of releases, called increments, are built and delivered to the customer.
- ➢ First, a simple working system (core product), that addresses basic requirements, is delivered. Customer feedback is recorded after each incremental delivery.
- ➢ Many increments are delivered, by adding more functions, until the required system is released. This model is used when a user demands a model of product with limited functionality quickly.




### Advantages

- ✓ Flexible to change requirements.
- ✓ Changes can be done throughout the development stages.
- ✓ Errors are reduced since the product is tested by the customer in each phase.
- ✓ Working software available at the early stage of the process.
- ✓ Easy to test because of small iterations and the initial cost is lower.

**Disadvantages**

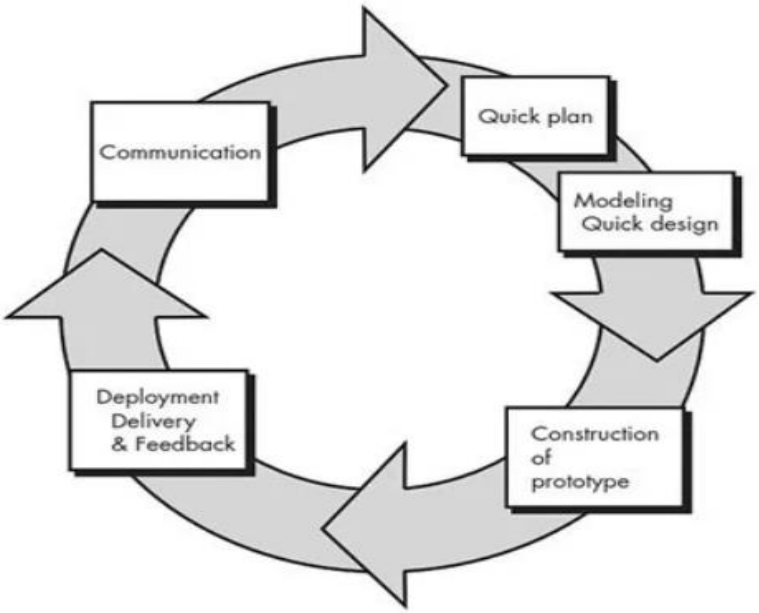
- ✓ Requires good planning and design.
- ✓ Modules and interfaces should be well defined.
- ✓ The total cost is high.
- ✓ Demands a complete planning strategy before commencement.
- ✓ Refining requirements in each iteration may affect system architecture.
- ✓ Breaking the problem into increments needs skillful management supervising.

## 3. Evolutionary Process Models

- **Evolutionary process models** are opted when the requirements may tend to change and also when the complete sophisticated product delivery cannot be done before a given deadline, but the delivery of a limited version of it is possible.
- In the incremental model, complete requirements are specified beforehand and these requirements are refined over time for each increment. The evolutionary model permits requirements, plans and estimates to evolve over time.

**Prototyping**

- When the requirements are unclear and are likely to change or when the developer is doubtful about working of an algorithm, a solution is to build a prototype and find out what is actually needed.
- Hence, in this model, one or more prototypes are made with unrefined currently known requirements before the actual product is made.



- A **quick design** is what occurs in a prototype model. The client evaluates the prototype and gives feedback and other requirements which are incorporated in the next prototype. This is repeated until the prototype becomes a complete product that is acceptable to the client.

***Advantages***

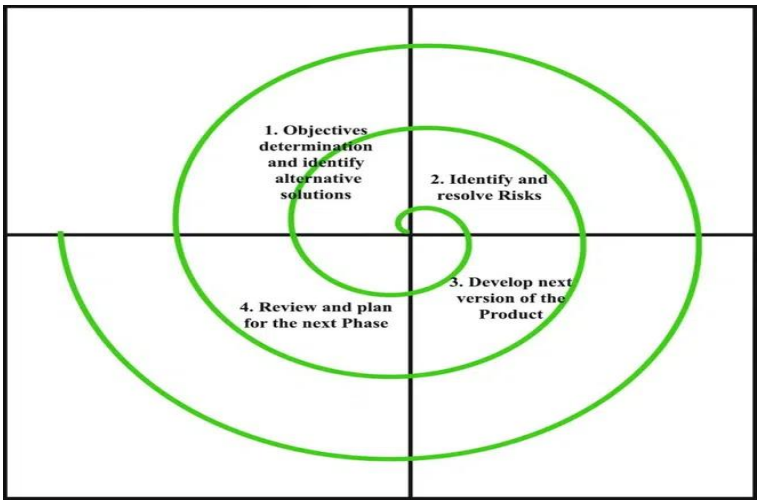
- ✓ Active involvement of the user.
- ✓ Errors are detected earlier.
- ✓ Feedback after each prototype helps in understanding the system better.
- ✓ Does not need to know detailed processes, input and output from the beginning.

***Disadvantages***

- ✓ Multiple prototypes can slow down the process.
- ✓ Frequent changes can increase complexity.
- ✓ Unsatisfied client leads to multiple throwaways.
- ✓ The customer may not be interested or satisfied after evaluating the initial prototype.

**Spiral Model**

- In spiral model, the software is developed through a series of increments. It looks like a spiral with loops where each loop is a **phase**. Each phase is split into four **sectors/quadrant**.




The functions of these four quadrants

- ✓ **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and

analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

- ✓ **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
- ✓ **Develop the next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
- ✓ **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started.

***Advantages***

- ✓ Reduces risk.
- ✓ Recommended for complex projects.
- ✓ Changes can be incorporated at a later stage.
- ✓ Strong documentation helps in better management.

***Disadvantages***

- ✓ Costly and not recommended for small projects.
- ✓ Demands risk assessment expertise.
- ✓ Looping is a complex process.
- ✓ Heavy documentation.

---

**Specialized Process Models**

- ➢ The specification models are use when only collection on specialized technique methods are expected for developing the specific software.
- ➢ These models tend to be applied when a specialized or narrowly defined software engineering approach is chosen.

**3 Types of Specialized Process Models**

- ✓ Component Based Development Model
- ✓ Formal Methods Models
- ✓ Aspect Oriented Software Development

**Components Based Development**

- The component-based development model ***leads to software reuse, and reusability*** provides software engineers with a number of measurable benefits. The software engineering team can ***achieve a reduction in development cycle time as well as a reduction in project cost*** if components are reused.
- The components-based development model makes use of various characteristic of spiral model. This model is evolutionary in nature. It means the necessary changes can be made in the software during each iteration of software development cycle.

It incorporates the following steps (implemented using an evolutionary approach):

- ✓ Available component-based products are researched and evaluated for the application domain in question.
- ✓ Component integration issues are considered.
- ✓ A software architecture is designed to accommodate the components.
- ✓ Components are integrated into the architecture.
- ✓ Comprehensive testing is conducted to ensure proper functionality.

**Formal Methods Models**

- The formal methods encompass a set of activities that leads to formal mathematical specification of computer software. It enables to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.
- They provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms. Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily.

***The advantage of using formal methods model*** is the it overcomes many problems that encounter in traditional software process models.

- ✓ Ambiguity, incompleteness and inconsistency are those problems that can be overcome if we use formal methods model.
- ✓ The formal methods offer the defect free software.

But ***some disadvantages of this models are***

- ✓ The development of formal models is currently quite time consuming and expensive.
- ✓ Because few software developers have the necessary background to apply formal methods, extensive training is required.
- ✓ It is difficult to use the models as a communication mechanism for technically unsophisticated customers and correct errors.

**Aspect Oriented Software Development (AOSD)**

- It focuses on the identification, specification and representation of cross-cutting concerns and into separate functional units as well as their automated into a working system.
- Aspect requirements define these cross-cutting concerns that have impact on the software architecture.
- Aspect Oriented Software Development is often referred as ***Aspect oriented programming (AOP).***
- It is a relatively new software engineering paradigm and is not matured enough. But is likely that it will adopt the characteristics of both the spiral and concurrent process models

- ***Cross-cutting concern*** is a term in software development referring to functionality that affects multiple components of a system, spanning across different layers or modules. These concerns often cannot be properly modularized within a single system component, making them challenging to manage and maintain.
- Examples of cross-cutting concerns include security, logging, and error handling.

**Advantages**

- ✓ Provides better modularization support of software design, reducing software design, development and maintenance costs
- ✓ Because concerns are encapsulated into different modules, localization of crosscutting concerns is better promoted and handled.
- ✓ Promotes reusability of code
- ✓ Smaller code size, due to tackling cross cutting concerns

---

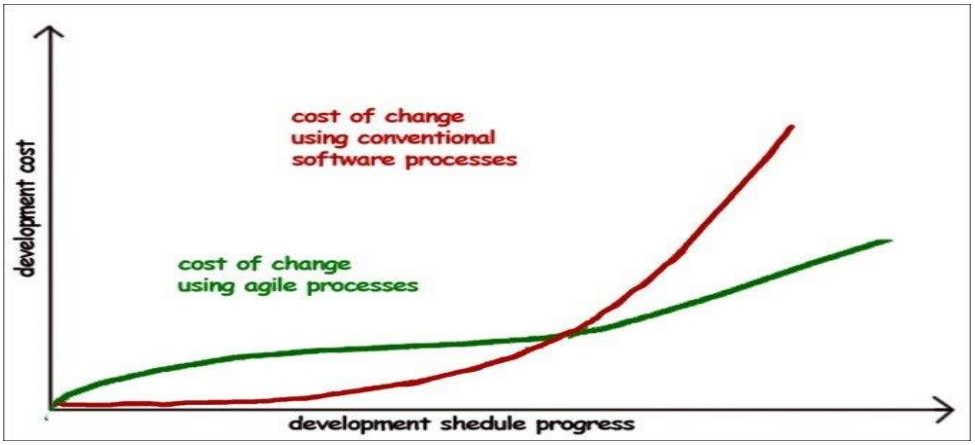
**Introduction to Agility**

**Agility**

- Agility means characteristics *of being dynamic, content specific, aggressively change embracing and growth oriented.* Agility is the *ability to respond quickly to changing needs.*
- It *emphasizes rapid delivery of operational software* and *de-emphasizes the importance of intermediate work products.*
- It adopts the customer as a part of the development team. Agility results in rapid, incremental delivery of software.
- The agile methods were developed to overcome the weakness of conventional software engineering.

**Agility and the Cost of Change**

- An agile process ***reduces the cost of change*** because ***software is released in increments and change can be better controlled within an increment.***
- Agility argue that a well-designed agile process "flattens" the cost of change curve shown in following figure, allowing a software team to accommodate changes late in a software project without dramatic cost and time impact.

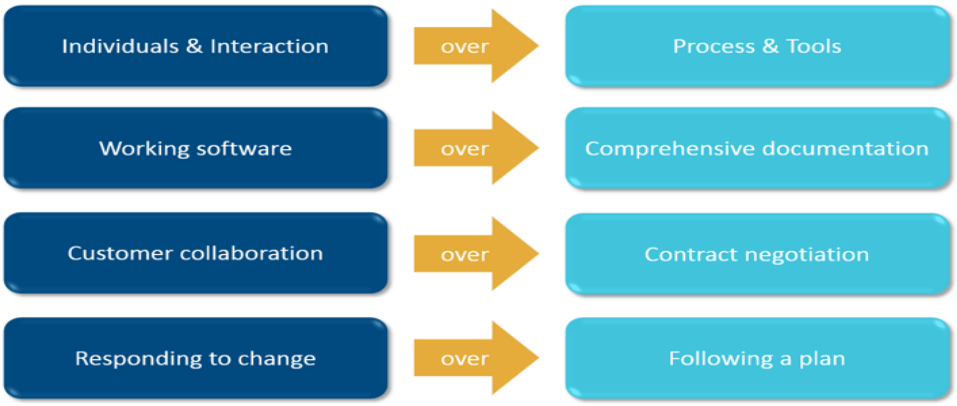



- When incremental delivery is coupled with other agile practices such as continuous unit testing and pair programming, the cost of making a change is attenuated.

**The Key Values and Principles of the Agile Manifesto**

The Agile Manifesto, also called the Manifesto for Agile Software Development, is a formal declaration of four key values and 12 principles to guide an iterative and people-centric approach to software development.

**4 key values of Agile**




- ***Individuals and interactions*** over ***processes and tools.***

  It is people who drive the development process and respond to business needs on the fly, so they take precedence over processes and tools.

- ***Working software*** over ***comprehensive documentation.***

  It deemphasizes the documentation of the development process, which historically took a huge amount of time and often bogged down the team.
- ***Customer collaboration*** over ***contract negotiation.***

  The customer becomes an important collaborator throughout the development process, ensuring their input is incorporated, and the result meets their needs along the way.
- ***Responding to change*** over ***following a plan.***

  Agile embraces change, focusing on releasing a minimum viable product that can be evaluated and adjusted from iteration to iteration.

**12 Principles of Agile Manifesto (***are based on the 4 core values)*

- ✓ Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- ✓ Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- ✓ Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- ✓ Business people and developers must work together daily throughout the project.
- ✓ Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- ✓ The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- ✓ Working software is the primary measure of progress.
- ✓ Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- ✓ Continuous attention to technical excellence and good design enhances agility.
- ✓ Simplicity — the art of maximizing the amount of work not done — is essential.
- ✓ The best architectures, requirements, and designs emerge from self-organizing teams.
- ✓ At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

**The Agile Process**

- The Agile process is an iterative approach to software development. The Processes which are adaptable of changes in requirements, which have incrementality and work on unpredictability.

- These processes are based on three assumptions which all do refer to the unpredictability in different stages of software process development such unpredictability at time requirements, at analysis and design or at time construction. So these processes are adaptable at all stages on SDLC.
- Instead of delivering a final working product at the end of the development lifecycle, teams work in small timeframes. ***These timeframes, called sprints***, are usually one to three weeks.
- Throughout sprints, teams collaborate and provide feedback on the work in progress. Through frequent communication, teams adapt to changing business and user needs. Communication also leads to the delivery of high-quality software products.

**Agile Process Methodologies/Models**

- ✓ Extreme Programming (XP)
- ✓ Adaptive Software development (ASD)
- ✓ Dynamic software Development Method (DSDM)
- ✓ Scrum
- ✓ Crystal
- ✓ Feature Driven development (FDD)
- ✓ Agile Modeling (AM)

---

**Extreme Programming (XP)**

Extreme programming is a software development methodology that's part of what's collectively known as agile methodologies. XP is built upon values, principles, and practices, and its goal is to allow small to mid-sized teams to produce high-quality software and adapt to evolving and changing requirements.

Extreme Programming provides specific core practices where −

- ✓ Each practice is simple and self-complete.
- ✓ Combination of practices produces more complex and emergent behavior.

**The process and roles of extreme programming**

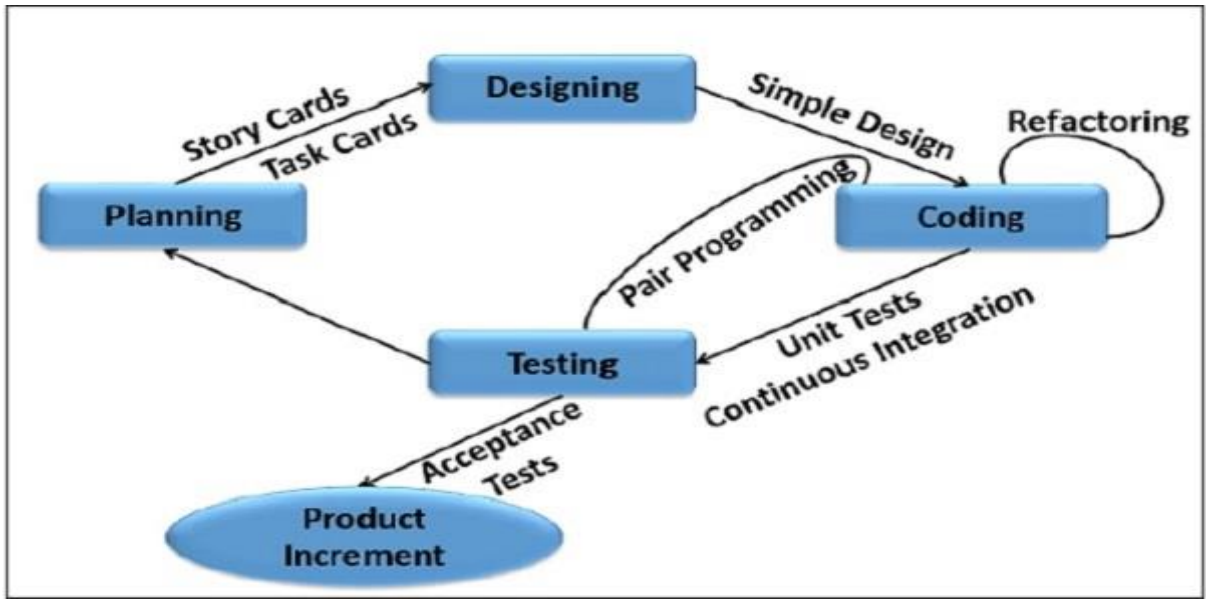
The XP framework normally involves 5 phases or stages of the development process that iterate continuously:

- ✓ **Planning,** the first stage, is when the customer meets the development team and presents the requirements in the form of user stories to describe the desired result. The team then estimates the stories and creates a release plan broken down into iterations

needed to cover the required functionality part after part. If one or more of the stories can't be estimated, so-called *spikes* can be introduced which means that further research is needed.

- ✓ **Designing** is actually a part of the planning process, but can be set apart to emphasize its importance. It's related to one of the main XP values that we'll discuss below -- simplicity. A good design brings logic and structure to the system and allows to avoid unnecessary complexities and redundancies.
- ✓ **Coding** is the phase during which the actual code is created by implementing specific XP practices such as coding standards, pair programming, continuous integration, and collective code ownership (the entire list is described below).
- ✓ **Testing** is the core of extreme programming. It is the regular activity that involves both unit tests (automated testing to determine if the developed feature works properly) and acceptance tests (customer testing to verify that the overall system is created according to the initial requirements).
- ✓ **Listening** is all about constant communication and feedback. The customers and project managers are involved to describe the business logic and value that is expected.

## Extreme Programming in a Nutshell




Extreme Programming involves −

- ✓ Collecting the user/client requirements and stored based on priority (High or Low Level) which called as ***Story card/Task Card or Index Card***. Based on that priority level, it will separated in ***Planning Phase*** itself.

- ✓ ***In Designing Phase,*** the high end design will be preferred based on planning phase.
- ✓ ***In Coding Phase,*** starting with a simple design to code the features at hand and redesigning when required.
    - There will ***Programming in pairs (called pair programming),*** with two programmers at one screen. While one of them will write the code, the other constantly reviews and provides inputs.
    - And next is ***Refactoring,*** mainly based on upgrades and improves the internal structure of the code without affecting the external behaviour.
- ✓ ***In Testing Phase***, which includes ***the unit testing*** were each modules are tested individually and eliminates defects early, thus reducing the costs. It also perform ***integration testing***, where the modules will be combined and tested,
- ✓ Putting a minimal working system into the production quickly and upgrading it whenever required.
- ✓ Keeping the customer involved all the time and obtaining constant feedback.

**Values of extreme programming**

XP has simple rules that are based on 5 values to guide the teamwork:

- ✓ **Communication -** Everyone on a team works jointly at every stage of the project.
- ✓ **Simplicity -** Developers strive to write simple code bringing more value to a product, as it saves time and effort.
- ✓ **Feedback -** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
- ✓ **Respect -** Every person assigned to a project contributes to a common goal.
- ✓ **Courage -** Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.

**Principles of extreme programming**

- ✓ **Rapid feedback -** Team members understand the given feedback and react to it right away.
- ✓ **Assumed simplicity -** Developers need to focus on the job that is important at the moment and follow YAGNI (You Ain't Gonna Need It) and DRY (Don't Repeat Yourself) principles.
- ✓ **Incremental changes -** Small changes made to a product step by step work better than big ones made at once.
- ✓ **Embracing change -** If a client thinks a product needs to be changed, programmers should support this decision and plan how to implement new requirements.
- ✓ **Quality work -** A team that works well, makes a valuable product and feels proud of it.