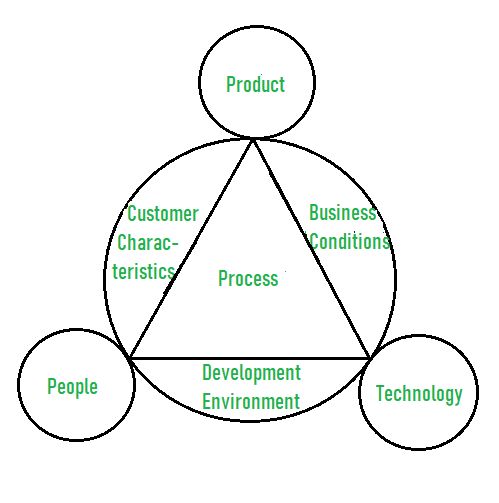
**­ Software Engineering**

**Unit 1&2**

**Difference Between PSP and TSP**

Software is the set of instructions in the form of programs to govern the computer system and process the hardware components. To produce a software product a set of activities is used. This set is called a [software process.](https://www.geeksforgeeks.org/software-processes-in-software-engineering/)



### PSP:

The personal software process is focused on individuals to improve their performance. The PSP is an individual process, and it is a bottom-up approach to software process improvement. The PSP is a prescriptive process, it is a more mature methodology with a well-defined set of tools and techniques.

### TSP:

TSP is a team-based process. It is focused on team productivity. Basically, it is a top-down approach. The TSP is an adaptive process, and process management methodology.

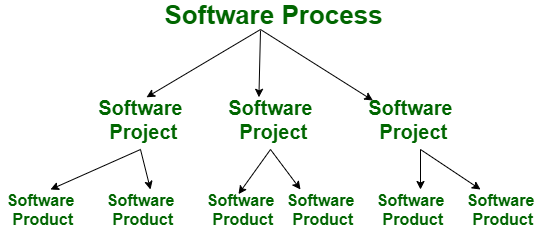
| **PSP** | **TSP** |
| --- | --- |
| PSP is a project management process that defines how to manage a project in a face-to-face environment. | TSP is a project management process that defines how to manage a project in a virtual environment. |
| PSP is more formal and structured than TSP. | TSP is less formal and structured than PSP. |
| PSP is based on the waterfall model. | TSP is based on the agile model. |
| PSP is more suited for large projects. | TSP is more suited for small projects. |
| PSP projects are typically completed in one phase. | TSP projects are typically completed in multiple phases. |
| PSP is a high-level language and it is easy to learn and use. | TSP is a low-level language and it is difficult to learn and use. |
| PSP is a structured language and it is easy to read and write. | TSP is an unstructured language and it is difficult to read and write. |
| PSP programs are written in English and they are easy to understand. | TSP programs are written in assembly language and they are difficult to understand. |
| PSP is a portable language and it can be run on any platform. | TSP is a platform-dependent language and it can be run only on specific platforms. |
| PSP is an interpreted language and it does not need to be compiled. | TSP is a compiled language and it needs to be compiled before it can be run. |
| PSP is a free language and it can be downloaded from the internet. | TSP is a commercial language and it is not available for free. |
| PSP is an open-source language and it is available to everyone. | TSP is a closed-source language and it is not available to everyone. |
| PSP is a linear process model that is completed in a sequential manner. | TSP, on the other hand, is an iterative and incremental process model that allows for feedback and changes at each stage of the process. |

# Difference between Product and Process

**Product:**   
In the context of software engineering, Product includes any software manufactured based on the customer’s request. This can be a problem solving software or computer based system. It can also be said that this is the result of a project.

**Process:**   
Process is a set of sequence steps that have to be followed to create a project. The main purpose of a process is to improve the quality of the project. The process serves as a template that can be used through the creation of its examples and is used to direct the project.

The main difference between a process and a product is that the process is a set of steps that guide the project to achieve a convenient product. while on the other hand, the product is the result of a project that is manufactured by a wide variety of people.



Let’s see the difference between Product and Process:-

| **S. No.** | **Product** | **Process** |
| --- | --- | --- |
| 1. | Product is the final production of the project. | While the process is a set of sequence steps that have to be followed to create a project. |
| 2. | A product focuses on the final result. | Whereas the process is focused on completing each step being developed. |
| 3. | In the case of products, the firm guidelines are followed. | In contrast, the process consistently follows guidelines. |
| 4. | A product tends to be short-term. | Whereas the process tends to be long-term. |
| 5. | The main goal of the product is to complete the work successfully. | While the purpose of the process is to make the quality of the project better. |
| 6. | Product is created based on the needs and expectations of the customers. | A process serves as a model for producing various goods in a similar way. |
| 7. | A product layout is a style of layout design in which the materials required to make the product are placed in a single line depending on the order of operations. | When resources with similar processes or functions are grouped together, it is referred to as a process layout. |
| 8. | Product patents are thought to offer a greater level of protection than process patents. | A process patent provides the inventor only limited protection. |

# Difference between Project and Product

**1. Project :**

Also known as a software project comprises the steps involved in making a product before it is actually available to the market. The project can be handled by people which are as less as one person to the involvement of a lot of people (over 100). These are usually assigned by an enterprise and are undertaken to form a new product that has not already been made.

**2. Product :**

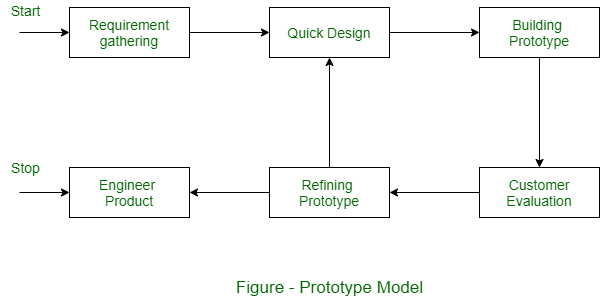
The study of products is a part of Software engineering. The software is built by developers on requests from the customer. After the customer is satisfied with the development process, he launches the software by manufacturing it. This can be a problem-solving software or computer based system. This is the result of a project. The software project, when completed, is called a product after it is available to the market for usage.

**Difference between Project and Product :**

| **S.NO.** | **Project** | **Product** |
| --- | --- | --- |
| 1. | It comprises the steps involved in making a software before it is actually available to the market. | It is the manufacture of the project for users. |
| 2. | The main goal of a project is to form a new product that has not already been made. | The main goal of the product is to complete the work successfully (solve a specific problem). |
| 3. | Project is undertaken to form a new software. | Product is the final production of the project. |
| 4. | It focuses on increasing the performance of the software that is being built. | A product focuses on the final result and the efficiency with which it can solve the given problem. |
| 5. | A project is done only once to get a new software. | A product can be made again and again for the purpose of distribution among users. |
| 6. | It is more risky as here, a software is being made for the first time. | It is relatively less risky as the software has already been made and tested. The only risk in most cases would be of wear and tear. |
| 7. | It is handled by the project managers. | It is handled by the product managers. |
| 8. | It exists before the software is made. | It exists after the completion of the software development phases. |
| 9. | It has some cost barriers as the enterprise sets a fixed budget for the project. | It has no cost barrier as it is continued to be developed only as long as the distributing enterprise is profitable. |
| 10. | Project is based on Predictive planning (up-front). | The product is based on adaptive planning (iterative). |

# Prototype model

[Prototyping Model](https://www.geeksforgeeks.org/software-engineering-prototyping-model/) is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.



System is partially implemented before or during the analysis phase thereby giving the customers an opportunity to see the product early in the life cycle.

**Advantages of using Prototype Model :**

1. This model is flexible in design.
2. It is easy to detect errors.
3. We can find missing functionality easily.
4. There is scope of refinement, it means new requirements can be easily accommodated.
5. It can be reused by the developer for more complicated projects in the future.
6. It ensures a greater level of customer satisfaction and comfort.
7. It is ideal for online system.
8. It helps developers and users both understand the system better.
9. Integration requirements are very well understood and deployment channels are decided at a very early stage.
10. It can actively involve users in the development phase.

**Disadvantages of using Prototype Model :**

1. This model is costly.
2. It has poor documentation because of continuously changing customer requirements.
3. There may be too much variation in requirements.
4. Customers sometimes demand the actual product to be delivered soon after seeing an early prototype.
5. There may be sub-optimal solutions because of developers in a hurry to build prototypes.
6. Customers may not be satisfied or interested in the product after seeing the initial prototype.
7. There is certainty in determining the number of iterations.
8. There may be incomplete or inadequate problem analysis.
9. There may increase the complexity of the system.

# Spiral Model

**Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

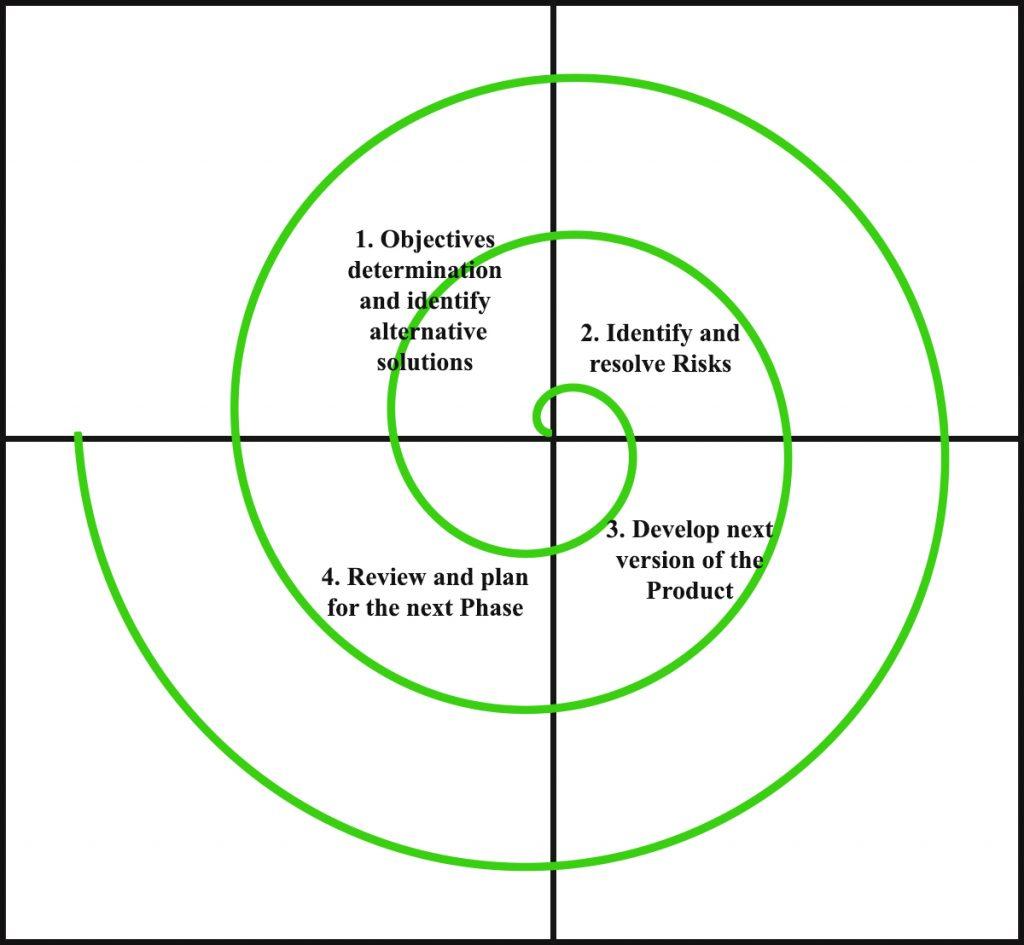
The Spiral Model is a software development life cycle (SDLC) model that provides a systematic and iterative approach to software development. It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:

1. Planning: The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.
2. Risk Analysis: In the risk analysis phase, the risks associated with the project are identified and evaluated.
3. Engineering: In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.
4. Evaluation: In the evaluation phase, the software is evaluated to determine if it meets the customer’s requirements and if it is of high quality.
5. Planning: The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.
6. The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development. It is also well-suited to projects with significant uncertainty or high levels of risk.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

1. **The below diagram shows the different phases of the Spiral Model: –**



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **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.
2. **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.
3. **Develop 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.
4. **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.

**Risk Handling in Spiral Model:**A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

**Advantages of Spiral Model**:

Below are some advantages of the Spiral Model.

1. **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
2. **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
3. **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
4. **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.
5. Iterative and Incremental Approach: The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.
6. Emphasis on Risk Management: The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.
7. Improved Communication: The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.
8. Improved Quality: The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability

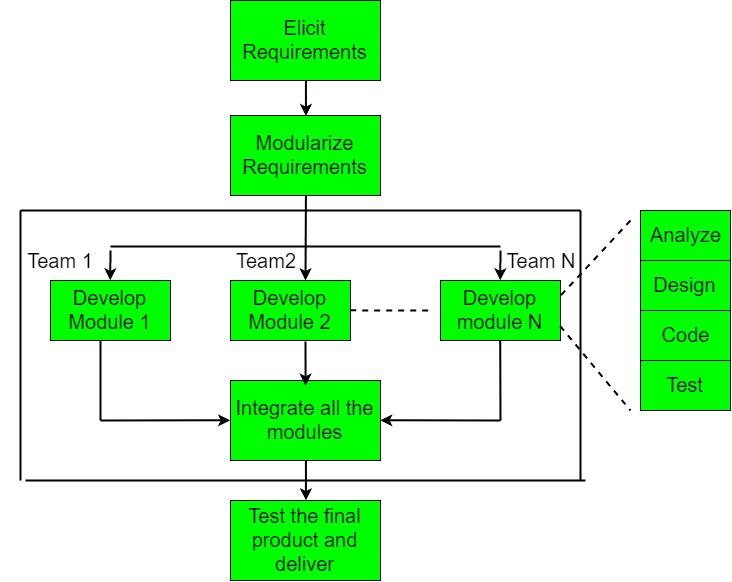
**Disadvantages of Spiral Model**:

Below are some main disadvantages of the spiral model.

1. **Complex:** The Spiral Model is much more complex than other SDLC models.
2. **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
3. **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
4. **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.
5. Complexity: The Spiral Model can be complex, as it involves multiple iterations of the software development process.
6. Time-Consuming: The Spiral Model can be time-consuming, as it requires multiple evaluations and reviews.
7. Resource Intensive: The Spiral Model can be resource-intensive, as it requires a significant investment in planning, risk analysis, and evaluations.

# Rapid application development model (RAD)

The Rapid Application Development Model was first proposed by IBM in the 1980s. The critical feature of this model is the use of powerful development tools and techniques. A software project can be implemented using this model if the project can be broken down into small modules where in each module can be assigned independently to separate teams. These modules can finally be combined to form the final product. Development of each module involves the various basic steps as in the waterfall model i.e analyzing, designing, coding, and then testing, etc. as shown in the figure. Another striking feature of this model is a short time span i.e the time frame for delivery(time-box) is generally 60-90 days.



The use of powerful developer tools such as JAVA, C++, Visual BASIC, XML, etc. is also an integral part of the projects. This model consists of 4 basic phases:

1. **Requirements Planning –** It involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it, and then processing it to form a final refined model.
2. **User Description –** This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.
3. **Construction –** In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform process and data models into the final working product. All the required modifications and enhancements are too done in this phase.
4. **Cutover –** All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

The process involves building a rapid prototype, delivering it to the customer, and taking feedback. After validation by the customer, the SRS document is developed and the design is finalized.

### Advantages:

* The use of reusable components helps to reduce the cycle time of the project.
* Feedback from the customer is available at the initial stages.
* Reduced costs as fewer developers are required.
* The use of powerful development tools results in better quality products in comparatively shorter time spans.
* The progress and development of the project can be measured through the various stages.
* It is easier to accommodate changing requirements due to the short iteration time spans.

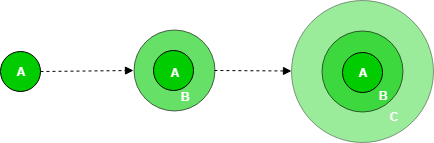
### Disadvantages:

* The use of powerful and efficient tools requires highly skilled professionals.
* The absence of reusable components can lead to the failure of the project.
* The team leader must work closely with the developers and customers to close the project in time.
* The systems which cannot be modularized suitably cannot use this model.
* Customer involvement is required throughout the life cycle.
* It is not meant for small-scale projects as in such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.

# 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.



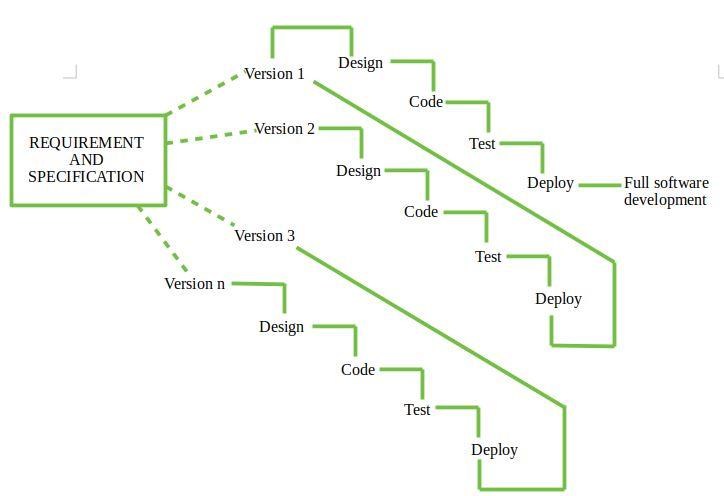
A, B, and C are modules of Software Products that are incrementally developed and delivered.

**Lifecycle activities:**

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered. At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the need of the customer. The Development Team first undertakes to develop core features (these do not need services from other features) of the system.

Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.

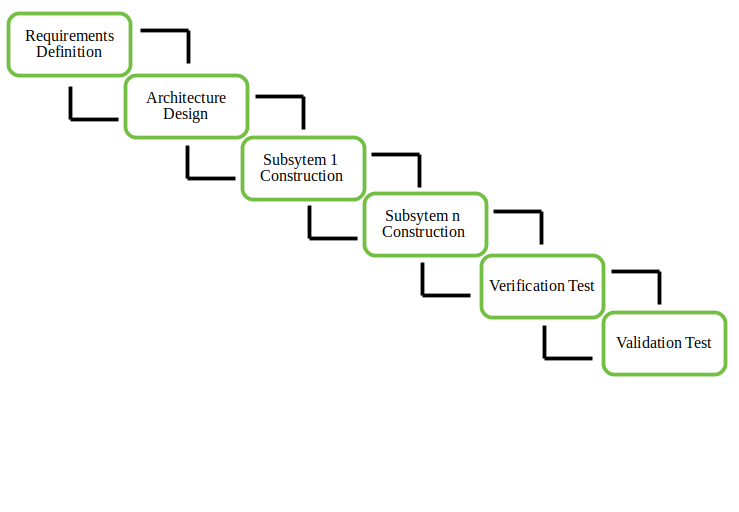
As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated into the next version. Each version of the software has more additional features than the previous ones.



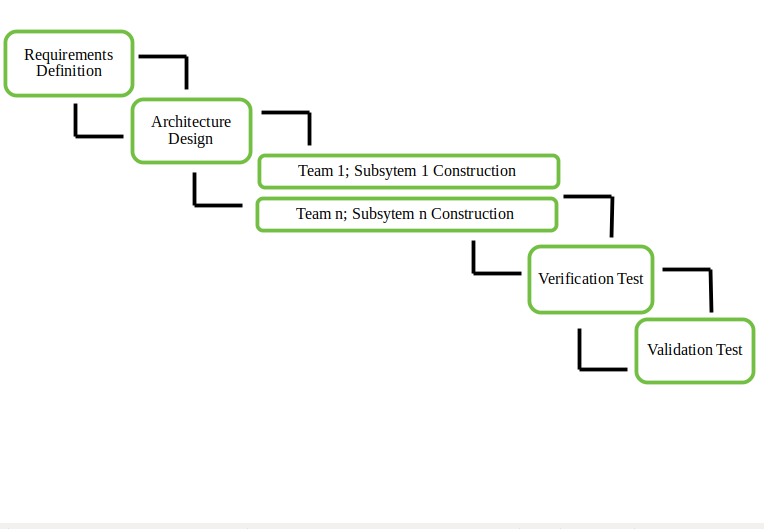
After Requirements gathering and specification, requirements are then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. After the last version (version n), it is now deployed at the client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



**Advantages-**

1. Prepares the software fast.
2. Clients have a clear idea of the project.
3. Changes are easy to implement.
4. Provides risk handling support, because of its iterations.

**Disadvantages-**

1. A good team and proper planned execution are required.
2. Because of its continuous iterations the cost increases.

# Agile Methodology Advantages and Disadvantages

[Agile Software Development Methodology](https://www.geeksforgeeks.org/software-engineering-agile-software-development/) is a process of software development (such as other software development methodologies – waterfall model, V-model, iterative model, etc.), however, the Agile development model is also a type of incremental model. Software develops in incremental, rapid cycles. In English, Agile means ‘the ability to move quickly and easily’ and respond to change rapidly – this is an important aspect of Agile software development.

**Advantages of Agile Methodology :**

1. In Agile methodology the delivery of software is unremitting.
2. The customers are satisfied because after every Sprint working feature of the software is delivered to them.
3. Customers can have a look of the working feature which fulfilled their expectations.
4. If the customers has any feedback or any change in the feature then it can be accommodated in the current release of the product.
5. In Agile methodology the daily interactions are required between the business people and the developers.
6. In this methodology attention is paid to the good design of the product.
7. Changes in the requirements are accepted even in the later stages of the development.
8. An Agile/Scrum approach can improve organizational synergy by breaking down organizational barriers and developing a spirit of trust and partnership around organizational goals.

**Disadvantages of the Agile Methodology :**

1. In Agile methodology the documentation is less.
2. Sometimes in Agile methodology the requirement is not very clear hence it’s difficult to predict the expected result.
3. In few of the projects at the starting of the software development life cycle it’s difficult to estimate the actual effort required.
4. Because of the ever-evolving features, there is always a risk of the ever-lasting project.
5. For complex projects, the resource requirement and effort are difficult to estimate.

**COCOMO Model**

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models. The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

* **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
* **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of the constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below. Boehm’s definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.
3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.
   1. Basic COCOMO Model
   2. Intermediate COCOMO Model
   3. Detailed COCOMO Model
4. **Basic Model :**

**E=a(KLOC)^b**

**Time=c(Effort)^d**

**Person required=Effort/Time**

1. The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a,b,c, and d for the Basic Model for the different categories of the system:

| **Software Projects** | **a** | **b** | **c** | **d** |
| --- | --- | --- | --- | --- |
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semi-Detached | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

1. The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in months. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, and expertise is taken into account, henceforth the estimate is rough.
2. **Intermediate Model –** The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system’s effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation. Classification of Cost Drivers and their attributes: **(i) Product attributes –**
   * Required software reliability extent
   * Size of the application database
   * The complexity of the product
   * Run-time performance constraints
   * Memory constraints
   * The volatility of the virtual machine environment
   * Required turnabout time
   * Analyst capability
   * Software engineering capability
   * Applications experience
   * Virtual machine experience
   * Programming language experience
   * Use of software tools
   * Application of software engineering methods
   * Required development schedule
3. **Detailed Model –** Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver’s impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort. The Six phases of detailed COCOMO are:
   1. Planning and requirements
   2. System design
   3. Detailed design
   4. Module code and test
   5. Integration and test
   6. Cost Constructive model

**Advantages of the COCOMO model:**

1. Provides a systematic way to estimate the cost and effort of a software project.
2. Can be used to estimate the cost and effort of a software project at different stages of the development process.
3. Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.
4. Can be used to evaluate the feasibility of a software project by estimating the cost and effort required to complete it.

**Disadvantages of the COCOMO model:**

1. Assumes that the size of the software is the main factor that determines the cost and effort of a software project, which may not always be the case.
2. Does not take into account the specific characteristics of the development team, which can have a significant impact on the cost and effort of a software project.
3. Does not provide a precise estimate of the cost and effort of a software project, as it is based on assumptions and averages.