What is Software Engineering?

The term **software engineering** is the product of two words, **software**, and **engineering**.

The **software** is a collection of integrated programs.

Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.

Computer programs and related documentation such as requirements, design models and user manuals.

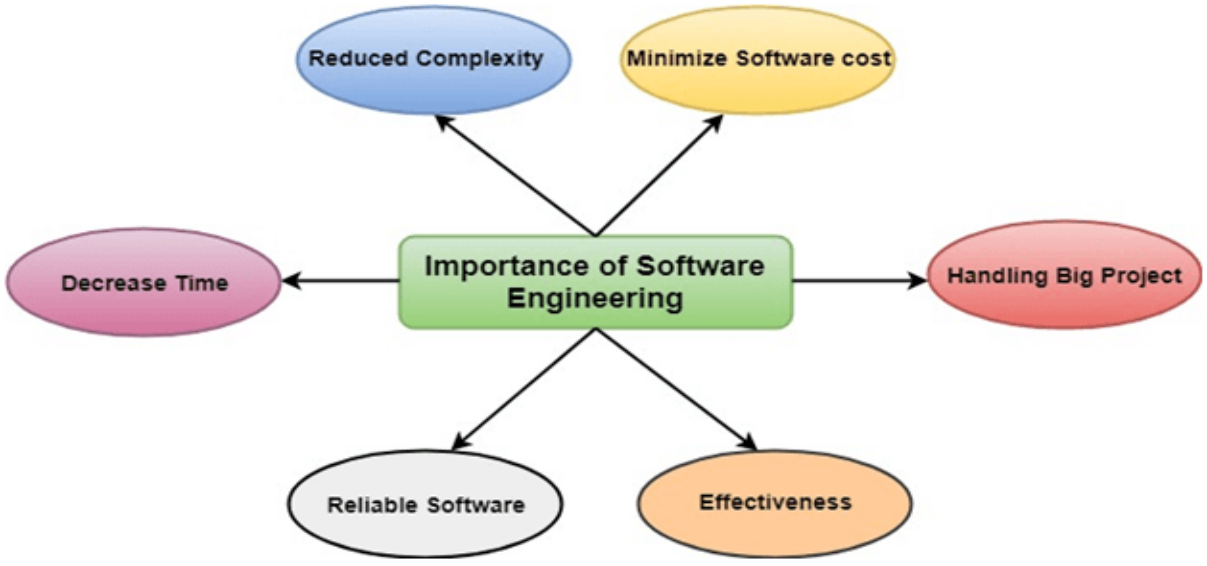
Engineering is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain**, and **improve frameworks, processes, etc**.

Why is Software Engineering required?

Software Engineering is required due to the following reasons:

- To manage Large software
- For more Scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

Importance of Software Engineering



The importance of Software engineering is as follows:

1. **Reduces complexity:** Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. Software engineering divides big problems into various small

issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.

2. **To minimize software cost:** Software needs a lot of hardwork and software engineers are highly paid experts. A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.
3. **To decrease time:** Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. This is a very time-consuming procedure, and if it is not well handled, then this can take a lot of time. So if you are making your software according to the software engineering method, then it will decrease a lot of time.
4. **Handling big projects:** Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software engineering method.
5. **Reliable software:** Software should be secure, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.
6. **Effectiveness:** Effectiveness comes if anything has made according to the standards. Software standards are the big target of companies to make it more effective. So Software becomes more effective in the act with the help of software engineering.

Software Applications

• 1. System software: such as compilers, editors, file management utilities

• 2. Application software: stand-alone programs for specific needs.

• 3. Engineering/scientific software: Characterized by “number crunching”algorithms. such

as automotive stress analysis, molecular biology, orbital dynamics etc

• 4. Embedded software resides within a product or system. (key pad control of a

microwave oven, digital function of dashboard display in a car)

• 5. Product-line software focus on a limited marketplace to address mass consumer

market. (word processing, graphics, database management)

• 6. WebApps (Web applications) network centric software. As web 2.0 emerges, more

sophisticated computing environments is supported integrated with remote database and

business applications.

• 7. AI software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

[1] What is the NEEDS OF SOFTWARE ENGINEERING

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.

Scalability - If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

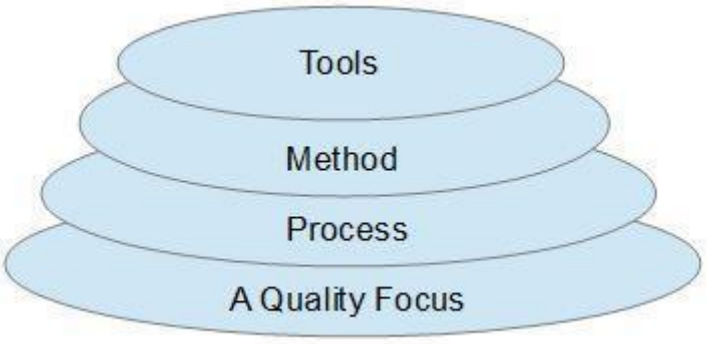
Cost - As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.

Dynamic Nature - The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Quality Management - Better process of software development provides better and quality software product

[10] *What is Layered Technology ?*

Software engineering is a layered technology. Which means, to develop software one will have to move from one layer to another. The layers are related and each layer demands the fulfillment of the previous layer. The figure below shows the four layers of software development :



1. A Quality Focus : An engineering approach must have a focus on quality which provides a continuous process improvement culture. This quality is achieved through use of Total Quality Management, which enables continuous process improvement custom that leads to the development of more established approaches to software engineering. The bedrock that supports software engineering is a Quality Focus.

2. Process : Process layer is the foundation that defines a framework with activities for effective delivery of software engineering technology. The process layer allows the development of software on time. It defines an outline for a set of key process areas that must be acclaimed for effective delivery of software engineering technology. The process layer establishes the context in which technical methods are applied, work products such as models, documents, data, reports, forms, etc. are produced, milestones are established, quality is ensured, and change is properly managed.

3. Methods : The Methods layer provide technical how-to's for building software. This layer covers a broad array of tasks that include requirements analysis, design, coding, testing, and maintenance phase of the software development. The Methods layer relies on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

4. Tools : The Tools layer provide automated and semi-automated support for the process and methods. Sometimes tools are integrated in such a way that other tools can use information created by one tool. This multi-usage is commonly referred to as Computer-Aided Software Engineering (CASE). CASE combines software, hardware, and software engineering database to create software engineering analogous to Computer-Aided Design (CAD) for hardware. CASE helps in application development including analysis, design, code generation, and debugging and testing.

Or

Tools: This layer contains automated or semi-automated tools that offer support for the framework and the method each software engineering project will follow.

Method: This layer contains the methods, the technical knowledge and “how-tos” in order to develop software.

Process: This layer consists of the framework that must be established for the effective delivery of software.

A Quality Focus: This layer is the fundamental layer for software engineering. As stated above it is of great importance to test the end product to see if it meets its specifications. Efficiency, usability, maintenance and reusability are some of the requirements that need to be met by new software.

Having Tools, Methods and Processes laid out from the beginning of any software engineering process makes it an easier task for both developers and project managers to check the quality of the end product and deliver a more complex software on time by staying on budget.

(11) Component Based Development

- Commercial off-the-shelf (COTS) software components, developed by vendors who offer them as products, provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software that is to be built.
- The component-based development model incorporates many of the characteristics of the spiral model.
- It is evolutionary in nature, demanding an iterative approach to the creation of software.
- However, the component-based development model constructs applications from prepackaged software components.

The component-based development model incorporates the following steps:

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

- The component-based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

[12] Characteristics of **Software:-**

- **Software is developed or engineered; it is not manufactured in the classical sense:**
    - Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.
    - In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.

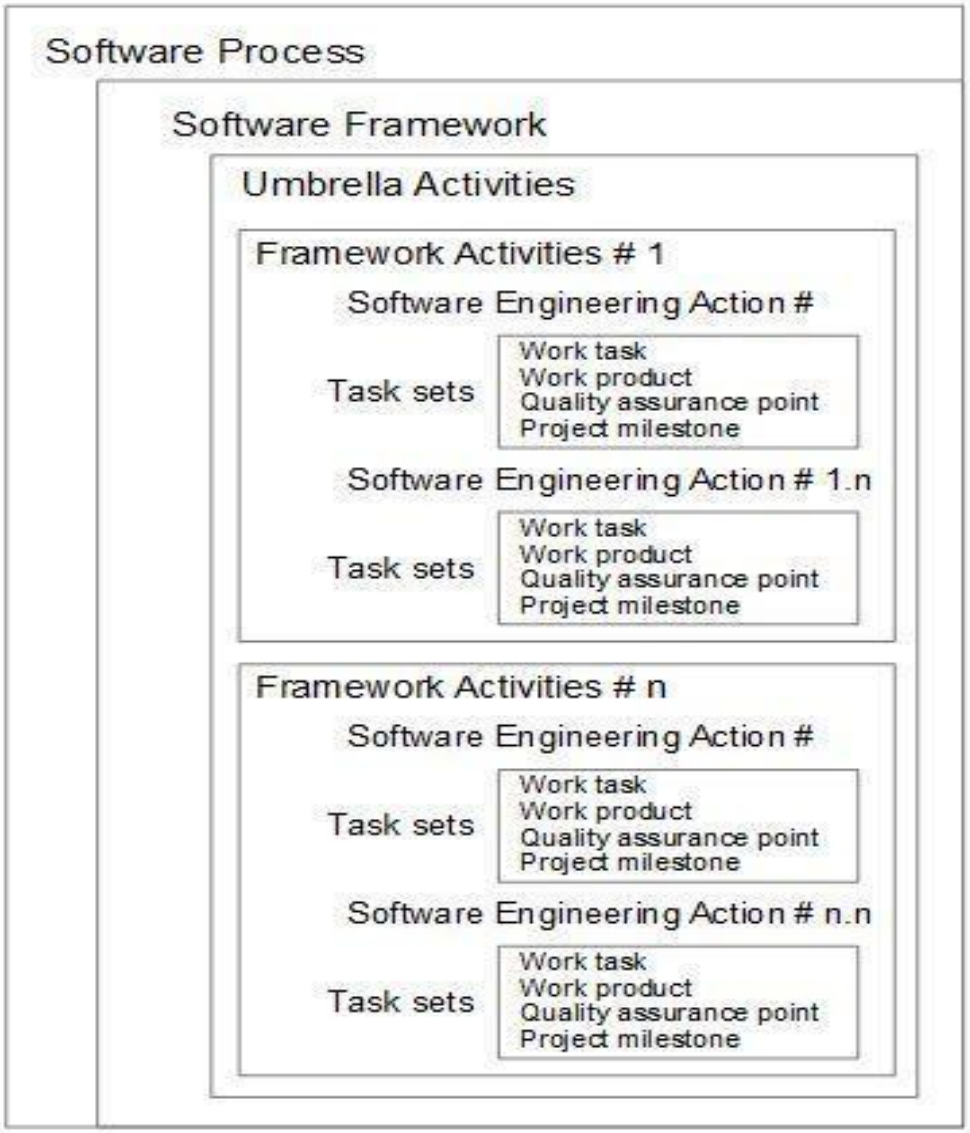
  [que]The software doesn't "wear out.":
    - Hardware components suffer from the growing effects of many other environmental factors. Stated simply, the hardware begins to wear out.
    - Software is not susceptible to the environmental maladies that cause hardware to wear out.
    - When a hardware component wears out, it is replaced by a spare part.
    - There are no software spare parts.
    - Every software failure indicates an error in design or in the process through which design was translated into

machine-executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. However, the implication is clear—the software doesn't wear out. But it does deteriorate.

- **The software continues to be custom-built:**
    - A software part should be planned and carried out with the goal that it tends to be reused in various projects.
    - Current reusable segments encapsulate the two information and the preparation that is applied to the information, empowering the programmer to make new applications from reusable parts.
    - In the hardware world, component reuse is a natural part of the engineering process.

(2) Process & Generic Process Model

- A software process is defined as a collection of work activities, actions, and tasks that are performed when some work product is to be created.
- Each of these activities, actions, and tasks reside within a framework or model that defines their relationship with the process and with one another.


Software Process
Software Framework
Umbrella Activities
Framework Activities # 1
Software Engineering Action #
Task sets
Work task
Work product
Quality assurance point
Project milestone
Software Engineering Action # 1.n
Task sets
Work task
Work product
Quality assurance point
Project milestone
Framework Activities # n
Software Engineering Action #
Task sets
Work task
Work product
Quality assurance point
Project milestone
Software Engineering Action # n.n
Task sets
Work task
Work product
Quality assurance point
Project milestone

(3) Umbrella Activities

- Software engineering process framework activities are complemented by a number of umbrella activities.
- In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk.

Typical umbrella activities include:

Software project tracking and control—allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.

Risk management—assesses risks that may affect the outcome of the project or the quality of the product.

Software quality assurance—defines and conducts the activities required to ensure software quality.

Technical reviews—assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

Measurement—defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs; can be used in conjunction with all other framework and umbrella activities.

Software configuration management—manages the effects of change throughout the software process.

Reusability management—defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.

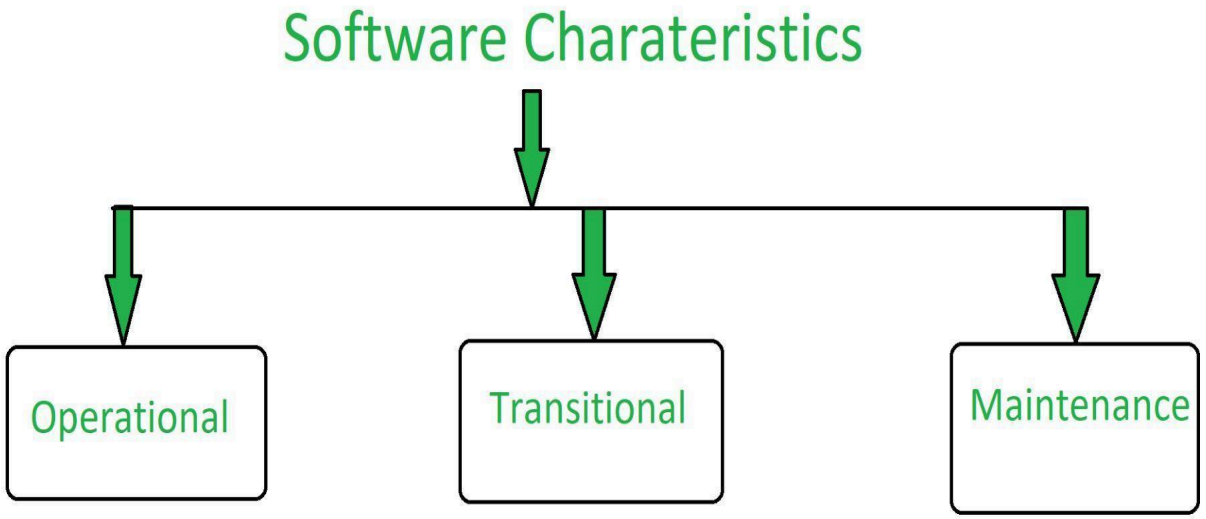
Work product preparation and production—encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

[15] Process Flow

- Linear process flow executes each of the five activities in sequence.
- An iterative process flow repeats one or more of the activities before proceeding to the next.
- An evolutionary process flow executes the activities in a circular manner. Each circuit leads to a more complete version of the software.
- A parallel process flow executes one or more activities in parallel with other activities (modeling for one aspect of the software in parallel with construction of another aspect of the software.

[2]What are the characteristics of a good software? Describe in detail.

Software is treated as a good software by the means of different factors. A software product is concluded as a good software by what it offers and how well it can be used. The factors that decide the software properties are divided into three categories: Operational, Transitional, and Maintenance. These are explained as following below.



1. Operational:
In operational categories, the factors that decide the software performance in operations. It can be measured on:

- Budget
- Usability
- Efficiency
- Correctness
- Functionality
- Dependability
- Security
- Safety

2. Transitional:
When the software is moved from one platform to another, the factors deciding the software quality:

- Portability
- Interoperability
- Reusability
- Adaptability

3. Maintenance:
In this categories all factors are included that describes about how well a

software has the capabilities to maintain itself in the ever changing environment:

- Modularity
- Maintainability
- Flexibility
- Scalability

(4) Prescriptive Process Models

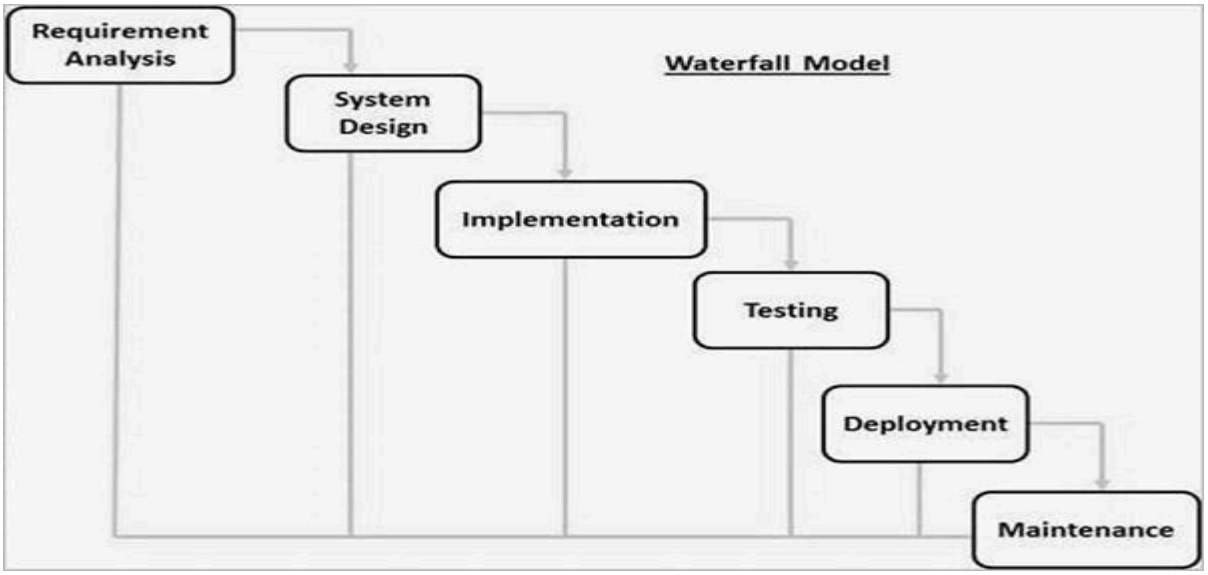
- Prescriptive process models were originally proposed to bring order to the chaos of software development.
- Traditional models have brought a certain amount of useful structure to software engineering work and have provided a reasonably effective road map for software teams.
- Following are prescriptive process models:
  - Waterfall model
  - Incremental Model
    - RAD Model
  - Evolutionary Process Model
    - Prototype Model
    - Spiral Model
    - Concurrent Process Model

[1] waterfall model

This model is named “waterfall model” because its diagrammatic representation resembles a cascade of waterfalls

The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

This model is easy to understand and reinforces the notion of “define before design” and “design before code”.



When to use SDLC Waterfall Model?

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare

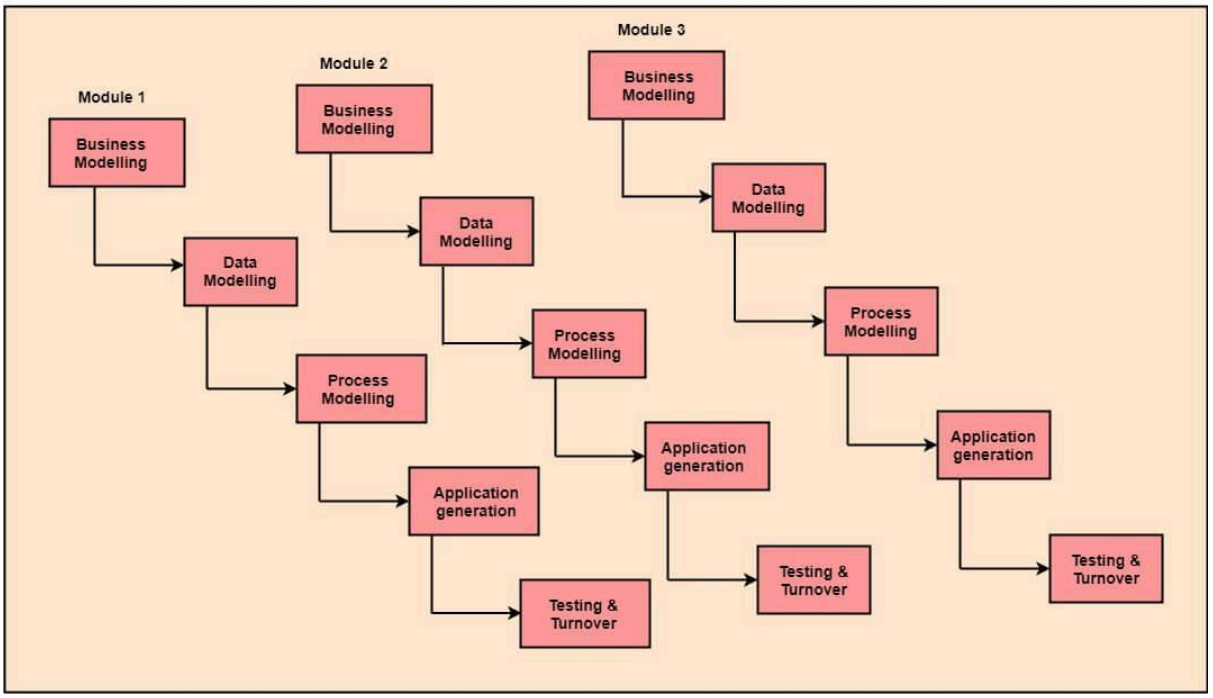
[2] RAD (Rapid Application Development) Model

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication

Fig: RAD Model



When to use RAD Model?

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Advantage of RAD Model

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

[3] Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

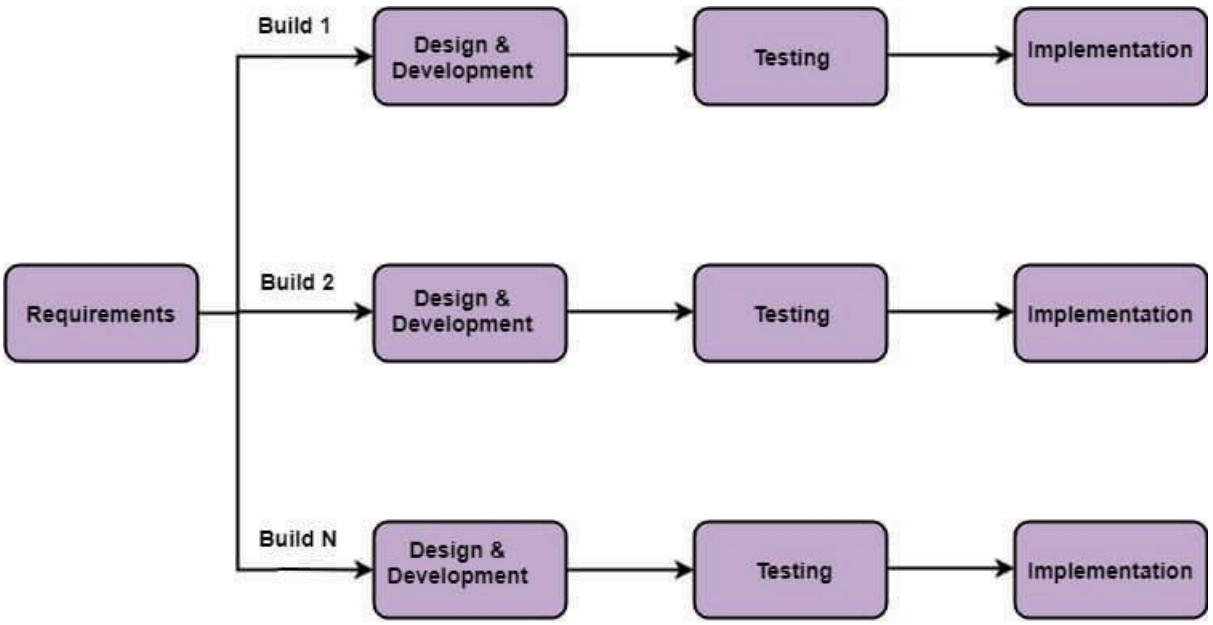


Fig: Incremental Model

When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Disadvantage of Incremental Model

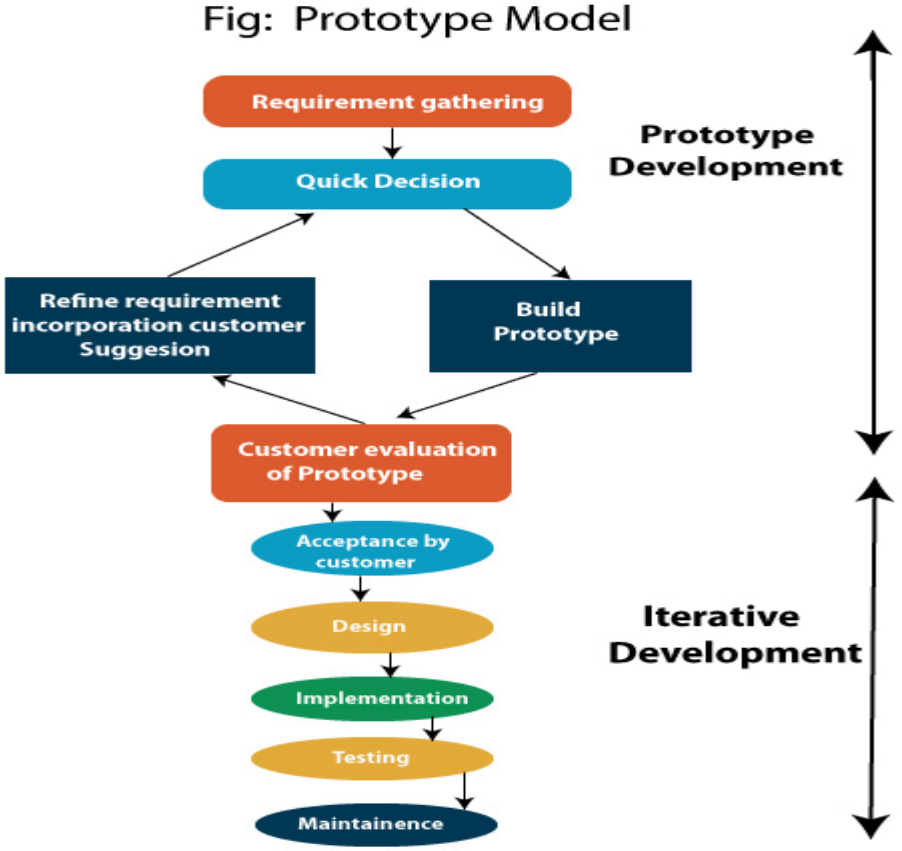
- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

[4] Prototype Model

Prototyping Model is **a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved**. It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail.

The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models)

Fig: Prototype Model



advantage of Prototype Model

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted
3. Regular visible process aids management
4. Support early product marketing
5. Reduce Maintenance cost.
6. Errors can be detected much earlier as the system is made side by side.

Disadvantage of Prototype Model

1. An unstable/badly implemented prototype often becomes the final product.
2. Require extensive customer collaboration
    - Costs customer money
    - Needs committed customer
    - Difficult to finish if customer withdraw
    - May be too customer specific, no broad market
3. Difficult to know how long the project will last.

4. Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
5. Prototyping tools are expensive.
6. Special tools & techniques are required to build a prototype.
7. It is a time-consuming process.

[5] Spiral Model

The spiral model is **similar to the incremental development for a system, with more emphasis placed on risk analysis**.

The spiral model has four phases: Planning, Design, Construct and Evaluation.

A software project repeatedly passes through these phases in iterations (called Spirals in this model).

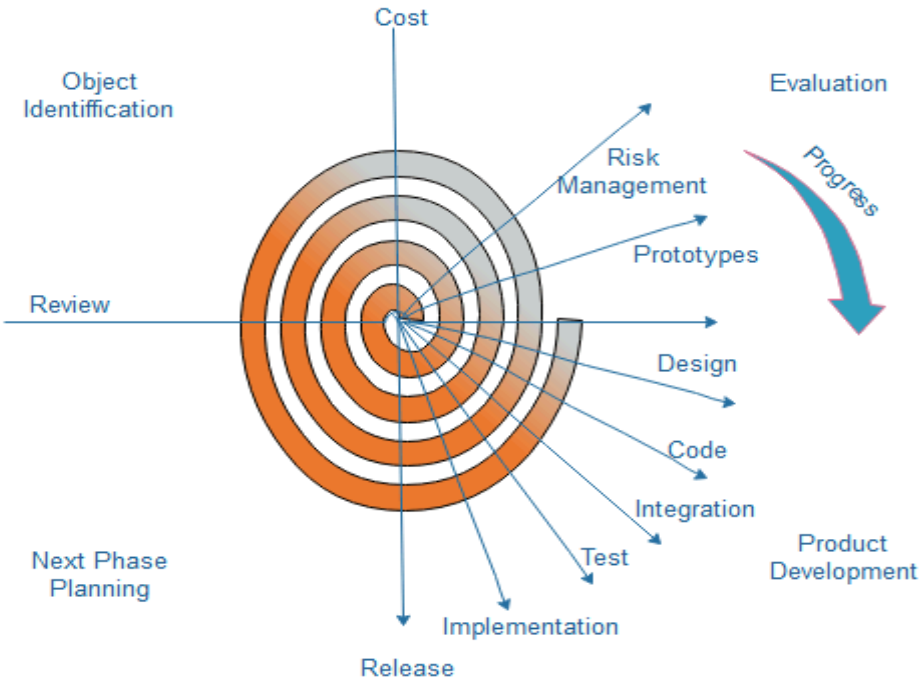


Fig. Spiral Model

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex

- When changes may require at any time
- Large and high budget projects

Advantages

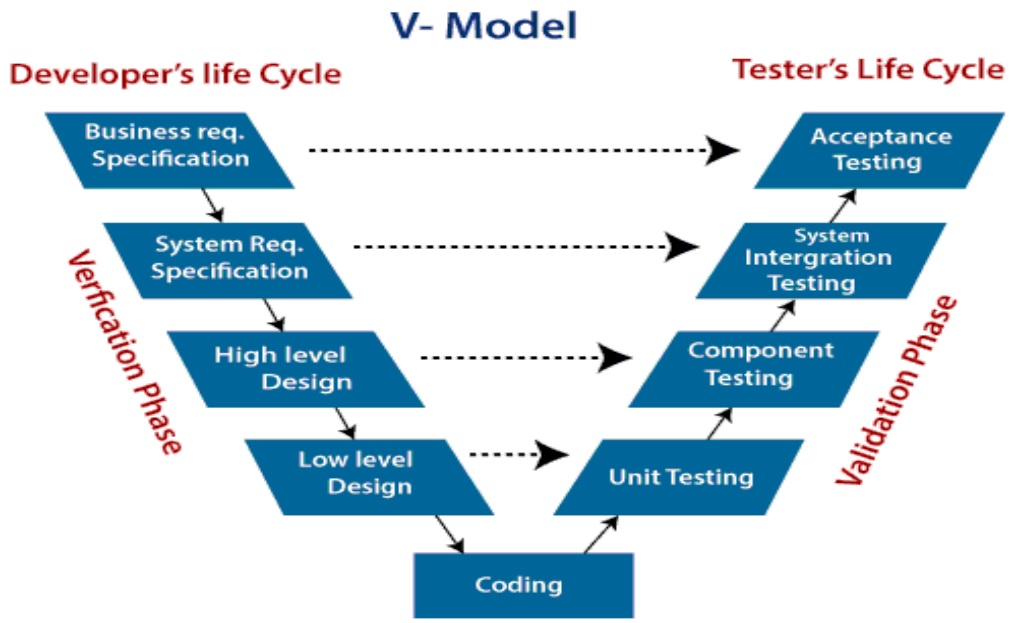
- High amount of risk analysis
- Useful for large and mission-critical projects.

Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.

[6] V-Model

V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.



Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Iterative Model

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

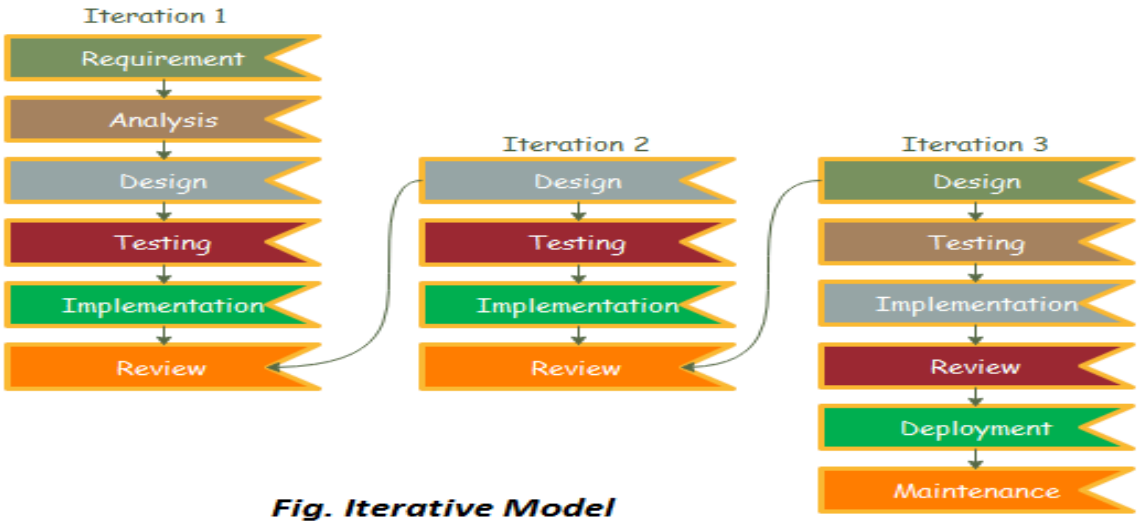


Fig. Iterative Model

The various phases of Iterative model are as follows:

1. Requirement gathering & analysis: In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

2. Design: In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

3. Implementation: In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

4. Testing: After completing the coding phase, software testing starts using different test methods. There are many test methods, but the most common are white box, black box, and grey box test methods.

5. Deployment: After completing all the phases, software is deployed to its work environment.

6. Review: In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product. And if there are any error found then the process starts again from the requirement gathering.

7. Maintenance: In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantage(Pros) of Iterative Model:

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

Disadvantage(Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements

5.

(12) Agile Process Model

- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds.
- Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

Advantages:

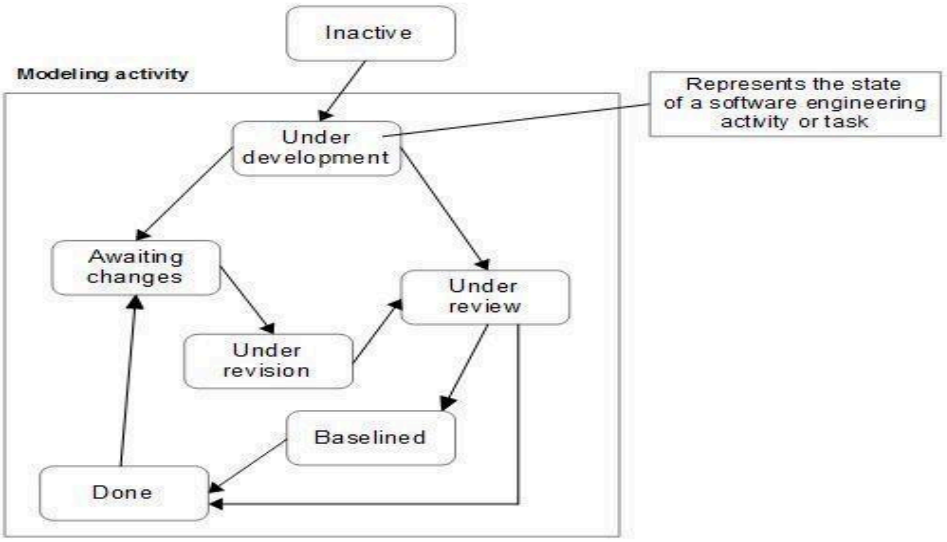
- Customer satisfaction by rapid, continuous delivery of useful software.
- Customers, developers and testers constantly interact with each other.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages:

- In case of some software, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process.

[9] Concurrent Process model

It is an evolutionary process model in software engineering. The term concurrent mean **“done at the same time”**. ... The concurrent process model shows the current state of activities, tasks and their associated states that remain in different phases.



Advantages of Concurrent process model :

1. This model is applicable to all types of software development processes.
2. It is easy for understanding and use.
3. It gives immediate feedback from testing.
4. It provides an accurate picture of the current state of a project.

Disadvantages of Concurrent process model :

1. It needs better communication between the team members. This may not be achieved all the time.
2. It requires to remember the status of the different activities.

[13]Agile Model

The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

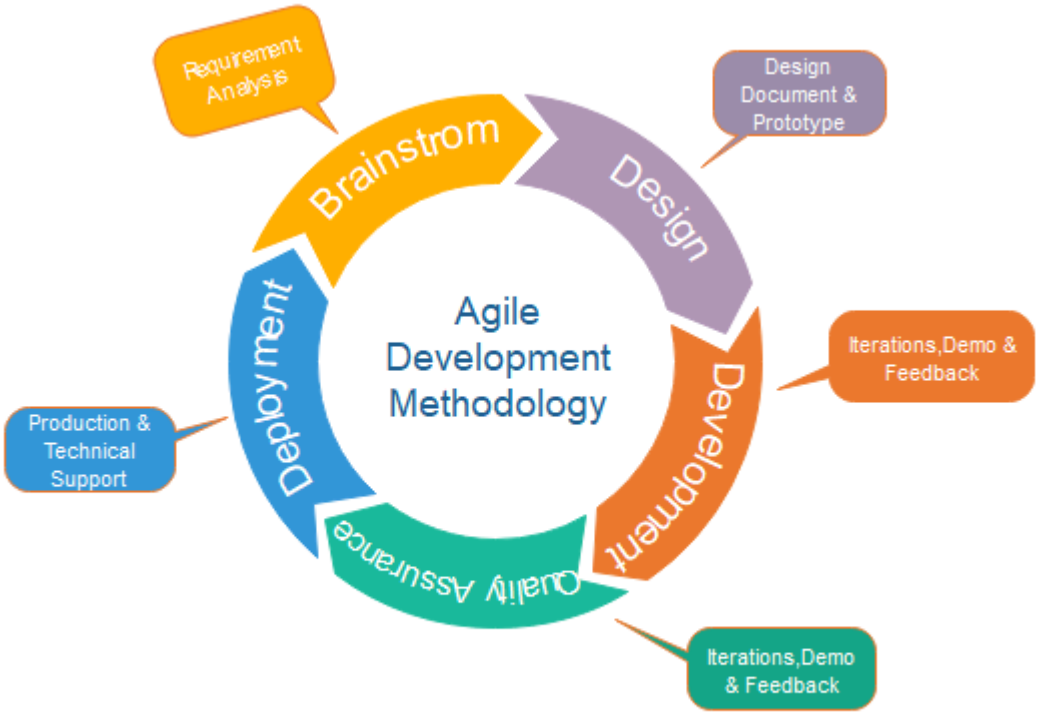


Fig. Agile Model

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

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

1. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

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

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- eXtreme Programming(XP)

Scrum

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

- **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
- **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

eXtreme Programming(XP)

This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.

Crystal:

There are three concepts of this method-

1. Chartering: Multi activities are involved in this phase such as making a development team, performing feasibility analysis, developing plans, etc.
2. Cyclic delivery: under this, two more cycles consist, these are:
    - Team updates the release plan.
    - Integrated product delivers to the users.
2. Wrap up: According to the user environment, this phase performs deployment, post-deployment.

Dynamic Software Development Method(DSDM):

DSDM is a rapid application development strategy for software development and gives an agile project distribution structure. The essential features of DSDM are that users must be actively connected, and teams have been given the right to make decisions. The techniques used in DSDM are:

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

The DSDM project contains seven stages:

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

Feature Driven Development(FDD):

This method focuses on "Designing and Building" features. In contrast to other smart methods, FDD describes the small steps of the work that should be obtained separately per function.

Lean Software Development:

Lean software development methodology follows the principle "just in time production." The lean method indicates the increasing speed of software development and reducing costs. Lean development can be summarized in seven phases.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

When to use the Agile Model?

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

- Linear process flow executes each of the five activities in process flow repeats one or more of the activities before proceeding to therocess flow executes the activities in a circular manner. Each circuit leaactivities in parallel with other activities (modeling for one aspect of the software in parallel with construction of another aspect of