UNIT-1 Software Engineering Models

Software – Software is a collections of programs and programs is a collection of instructions software is a general term for – the various programs and data stored on computer storage devices to operate the computer and related devices. Software controls the operation of computer hardware. It has three principle functions –

1) It manages the computer resources.
2) It serves as an interface between the organization and its stored information.
3) It helps to solve problem by using computer hardware.

Types of Software – There are three types of software –

1) **System Software** – System software or operating system is the software used by the computer to translate inputs from various sources into a language which a machine can understand. System software consist of three types of programs – control
(a) System support program – It controls the execution of programs manages the storage and processing resources of the computer.

(b) System Support program - It provides services to other computer programs like copying, deleting, merging, sorting files etc.

(c) System development program - It assists in the creation of application programs. Example - BASIC Interpreter etc.

2) Applications Software - It applies the power of the computer to solve problem by performing specific tasks. It can support individuals, groups and organizations.

3) Utility Software - This software helps to manage, maintain and control computer resources.

Operating system typically contains the necessary tools for this, but separate utility programs can provide improved functionality.

Examples - of utility programs are - antivirus software, backup software and disk tools.

Applications of Software - Software may have a number of applications. Software may be applied in any situation for which a pre-specified set of procedural steps has been defined.

The following are the main application areas of software -

(I) Business Software - Business Information processing is the largest single software application area. Business application has a very broad area such as payroll, account receivable, accounts payable, inventory, purchase. Now-a-days business applications are playing very important role for providing the information.

(II) System Software - System programs are written to provide service to other programs. System software is a collection of Programs. Ex - compilers, editors

(III) Embedded Software - Intelligent consumer products are becoming very popular in industrial market. Embedded software resides in ROM and is used to control products for user. Ex - microwave oven, washing machine, automobiles etc.

(IV) Scientific Software - Engineering and scientific software have been characterized by "number crunching" algorithm. There are many softwares used for scientific uses.

(v) Personal Computer Software - Personal computer software may be said to be a small business software which is used for single user or personal user. This type of application include word processing, Spreadsheets, small databases management etc.

(vi) Real-time Software - Software that analyze events as they occur are called real-time software. Elements of real-time software include a data gathering components that collects information from external environment.

(vii) Web based Software - The web paged retrieved by browser are software that incoporate executable instruction and Data. Web-based software can be used for E-commerce and internet banking.

Software Crisis Characteristics -

1) Software is easy to change - It is true that software is easy to change but making changes without introducing errors is extremely difficult.

2) Software can work right the first time - A software engineer does not prepare any prototype for software, he often accepts the job without building a prototype.

3) Reusing software increases safety code - Reuse is a very powerful tools that can yield dramatic improvement in development efficiency but it still requires analysis to determine its suitability and testing.

4) Computers provide greater reliability than the devices they replace -
There is no limit to how many times a given piece of code can be executed before it "wears out".

5) Once we write the program and get it to work our job is done - In software developing requirement analysis and designing are the important things. Writing the program is not the only thing for software developm

Components of Software -

There are various components of software -

"A software component is a software element that conforms to a component model and can be independently composed without modificatio according to a composition standard.

Software is also a collection of various components. These components are can be trea

as various software programs, various software modules etc.
Reusability is an another important feature of high quality software component. A software component should be designed and implemented so that it can be reused in many different programs. Software components are built using a programming language.
Software components must be catalogue for easy reference -

1. Off the shelf Components — Existing software that can be acquired from a third party or that has been developed internally for a past project. These components are purchased from third part and ready to use for current Project.
2. Full Experience Components - Existing design, code or test data developed for past projects that are similar to the software to be build for the current are experienced components as they had full experience.
3. Partial Experience Components - Existing design, specification or test data developed for past projects that are similar to the software to be build for the current bilt requires only little modifications are known as Partial Experience Components.
4. New Components — Software components that must be build by the software team specifically for the need of the current project.

Test

Characteristics of software -

1. Software is designed like any product based on some software specification.
2. Software is developed in teams not by individuals.
3. It generally included clear and detailed documentation.
4. Software is meant for users who does not have good knowledge of computers
5. Software is designed to run on different platforms.
6. It has a lifeline in years.
7. It generally requires some modification from time to time to accomodate changes taking place in the organization..
8. Protecting privacy issues are also taken into consideration in designing software

Disadvantages of Software –

1. The Understanding Gap between the user and developer is the one of the difficult aspects of software development.
2. Software takes a longer time to develop.
3. Software development is costly.
4. Sometimes there is NO Assurance of Quality of software.
5. Software are sometimes difficult to monitor and control.
6. Software maintenance is a very messy work.

Test

Software Engineering – "A discipline whose aim is the production of quality software, software that is delivered on time, within budget & that satisfies its requirements."

OR

Software engineering is a field of science implemented in computer's area used to develop various methods & techniques in order to achieve high quality software
* at the lower cost * within Scheduled time
* at a consistent Production rate.

Test

Software Engineering – "A systematic approach to the development, operation, maintenance and retirement of software, where software is computer programs, procedures, rules and associated documents and data pertaining to the operation of a computer system".

One of the main objective of software engineering is to help developers obtain high quality software. This Quality is achieved through use of Total Quality Management (TQM) which enables continuous process improvement.

Layers of Software Engineering –

Layers: Tool, Method, Process

1) Process layer – It allows the development on time. It defines an outline for a set of key process.
2) Method layer – It provides technical knowledge for developing software. This area covers a broad array of tasks that includes requirements, analysis, design, coding testing and maintenance phase of the software development.
3) Tools Layer – This layer provides computerized or semi-computerized support for the process and the method layer. 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 for hardware. CASE helps in application development.

Software engineering Requirements —

* A software engineering is an engineering approach for software development.
* A small program can be written without using software engineering principles
* But If one want to develop a large softw the software engineering principles are must to achieve a good quality sofware.
* Without using software engineering principles it would be difficult to develop large programs.
* engineering helps to ~~improve~~ reduce programming complexity.
* There are two techniques used in soft engineering

* Abstraction is the technique in which only those aspects of the problem that are relevant for certain purpose are taken.
* Decomposition is the technique in which a complex problem is divided into several subproblems and then all the subproblems are solved one by one.
* A program can be developed according to the programmer's individual style of development but a software product must be developed using the accepted software engineering principles.

Iterative Enhancement Model —

* This model combines the benefits of both waterfall model and prototype model.
* The basic idea is that the software begins by specifying and implementing a small part of the software.
* This software is then reviewed for further requirements. Then a list of additional requirements is prepared which helps to develop the new version of the software.
* For each cycle the same process is repeated to get a new version of the software.
* In this model four phases are repeated.

Unit 1

Difference between Software Engineering process and Conventional Engineering Process

Software Engineering Process and Conventional Engineering Process, both are processes related to computers and development. In this article, we will see the similarities as well as differences between both, that is Software Engineering Process and the Conventional Engineering Process.

Software Engineering Process

Software Engineering Process is an engineering process that is mainly related to computers and programming and developing different kinds of applications through the use of information technology.

Conventional Engineering Process

Conventional Engineering Process is an engineering process that is highly based on empirical knowledge and is about building cars, machines, and hardware. It is a process that mainly involves science, mathematics, etc.

Conventional Engineering is related to computers, writing programs, and implementing them.

Similarities between Software Engineering Process and Conventional Engineering Process

- Both Software Engineering and Conventional Engineering Processes become automated after some time.
- Both these processes are making our day-to-day place better.
- Both these processes have a fixed working time.
- Both processes must consist of deeper knowledge.

Difference between Software Engineering Process and Conventional Engineering Process

Aspect	Software Engineering Process	Conventional Engineering Process
Process	Software Engineering Process is a process that majorly involves computer	The conventional Engineering Process is a process that majorly

Aspect	Software Engineering Process	Conventional Engineering Process
	science, information technology, and discrete mathematics.	involves science, mathematics, and empirical knowledge.
Focus Area	It is mainly related to computers, programming, and writing codes for building applications.	It is about building cars, machines, hardware, buildings, etc.
Cost	In Software Engineering Process construction and development costs are low.	In Conventional Engineering Process construction and development cost is high.
Application	It can involve the application of new and untested elements in software projects.	It usually applies only known and tested principles to meet product requirements.
Development Effort	In the Software Engineering Process, most development effort goes into building new designs and features.	In Conventional Engineering Process, most development efforts are required to change old designs.
Emphasis	It majorly emphasizes quality.	It majorly emphasizes mass production.

Aspect	Software Engineering Process	Conventional Engineering Process
Product Nature	Product development develops intangible products (software).	Product development develops tangible products (e.g. bridges, buildings).
Design Requirement	Design requirements may change throughout the development process.	Design Requirements are typically well-defined upfront.
Testing	Testing is an integral part of the development process.	Testing occurs mainly after product completion.
Prototyping	Prototyping is common and helps to refine requirements.	Prototyping is less common due to cost and time.
Maintenance	Maintenance and updates are necessary to keep software relevant.	Maintenance is typically scheduled or reactive.
Complexity	Software development often involves complex logic and algorithms.	Conventional engineering may have more complex physical properties to deal with.
Framework	Software development often follows established standards and frameworks.	Conventional engineering may have well-established regulations and standards.
Cost Dynamics	Software development is typically less expensive to start, but costs may increase	Conventional engineering may be more expensive to start due to materials and

Aspect	Software Engineering Process	Conventional Engineering Process
	with maintenance and updates.	construction but may have lower maintenance costs.
Methodologies	Agile methodologies are commonly used in software development.	Conventional engineering may use more traditional project management approaches.

Software Development Life Cycle - (Models)

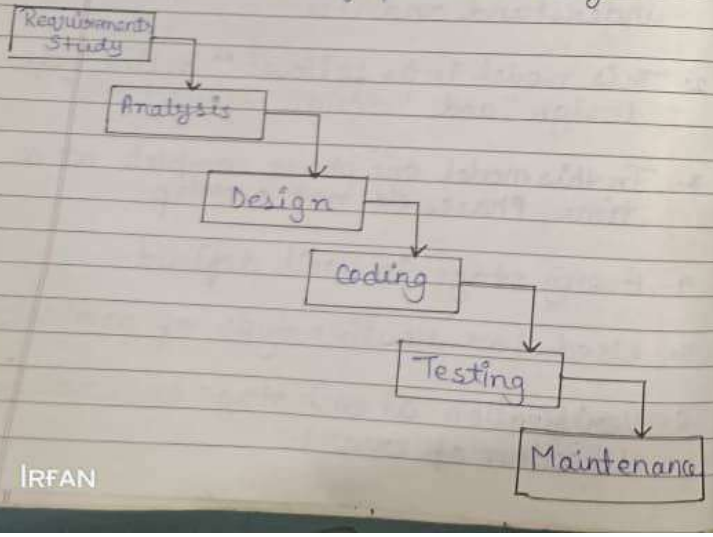
1) The Waterfall Model - * Waterfall model was First process model to be introduced.
* Waterfall model is the simplest software process model.
* It is the oldest model.
* It is also referred to as sequential life cycle model.
* Waterfall model is mostly used for small projects.
* In a waterfall model each phase must be completed fully before the next phase can begin.
* In waterfall model the output of one phase is used as the input for the second phase.
* This model is called "The Waterfall Model" because the flow of the process from one phase to the next resembles the flow of water falling from steps.

There are the following different phases for the software development in the model.

1. System Requirements - In this phase requirements for the larger system are established. This stage ensures that the system will interface properly with peoples, hardware and other software.
2. Analysis - In the next phase all the activities of the requirement analysis are

Disadvantages of Waterfall Model -

1. This model expects requirements early in the process.
2. Once an application is in the testing pha It is very difficult to go back and mak changes.
3. High amounts of risk.
4. Not a good model for complex problem
5. User training is not given enough importance.
6. The next stage begins only after the completion of previous stage.



Iterative Waterfall Model -

* Iterative Waterfall Model has the same phases as the waterfall model but with fewer restrictions.
* To overcome the shortcomings of the waterfall model, a process model also known as Iterative waterfall Model was developed.
* In iterative waterfall model the full specification of requirements are not needed at the start.
* This model is also known as incremental model because we increment the requirements at each phase.
* This model overcomes the limitations of the original model by adding an iterative loop to the end of each cycle. That is in order to keep changes requirements the analysis phase is revisited at the end of each cycle.

Advantages of Iterative Waterfall Model

* This model can result in better testing as testing each increment is likely to be easier than testing the whole system.
* The major advantage is that the costumer does not have to pay for the entire system at a time.
* It is used for big projects.

done to determine software requirem
When the requirements are fully determined then the Design phase begins.

3) Design - The goal of this phase is to transform the requirements specification into a structure that is suitable for converting in some program language. The purpose of the design phase is to specify a particular software system that will meet the stated requirements.

4) Coding - The main purpose of this pha is to translate the software design into source code. The end product of this phase is a set of program modules.

5) Testing - In this phase effective testing will contribute to the deliver of higher quality software products, more satisfied user. In this phase unit as well as combine module testing is performed.

6) Implementation - This is the process in which the developed syste is handed over to the client. The whole system is installed at the required location if any updation is needed the required resources are made available. In this phase users are also given training how to use the soft

7) Maintenance - Software maintenance is a very broad activity that includes error correction, enhancement of requirements, deletion of obsolete capabilities.

* In waterfall model At the end of each phase, a review takes place to determine whether the software is in right path or not.
* In this model testing starts only after the mod software is complete.
* no phases overlap each other.

Advantages of waterfall model -

1. This model is simple and easy to understand and use.
2. This model is as follows "define before design" and "design before code".
3. In this model one phase complete at a time. Phases do not overlap.
4. Every stage is well defined.
5. Used for routine type of projects.
6. Verification at each stage ensures early detection of errors.

IRFAN

Prototype Model –

* A prototype is a working system that is developed to test ideas about the new system.

* A prototyping is a process of building a model of the system to be developed.

* The basic idea of prototyping is thus instead of fixing requirements before design and coding, a prototype is built to understand the requirements.

* On the basis of known requirements the prototype is built.

* Designing the prototype enables the user to actually feel how the system will work.

* The interaction with the prototype enable the user to better understand the requirements of the desired system.

Advantages of the Prototype Model –

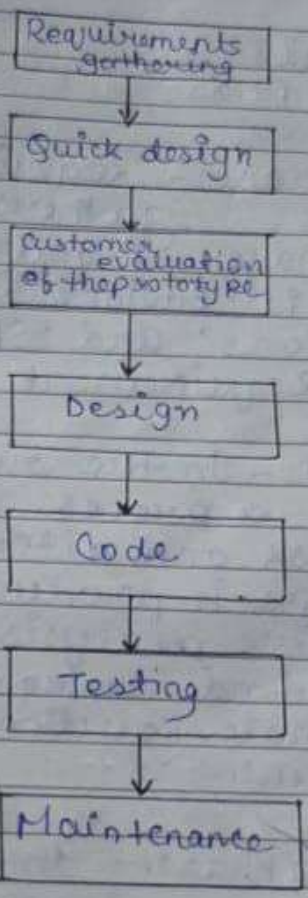
* well suited where the user's need are not clearly idenfiable/definable in advance.
* It is useful in the development of very large systems.

* It provides better understanding of the customers needs.
* Helps to determine the feasibility of th system.
* Users are actively involved in the develo Hence it reduces the communication gap b/w the developer and the user.
* Errors can be detected much earlier.
* Reduces the risks associated with the projects.
* Modification of prototype is faster and cheaper than the modification of final software.
* It reduces maintenance costs.

Disadvantages –

* Developer may ignore the quality of the system in order to get the prototype working quickly.
* Extra cost is required to develop two systems.
* Users on seeing a working prototype often start expecting the actual system to be ready soon after.

* An inappropriate O.S or programming language may be used for prototype.
* This model may increase the complexity of the system.



Prototyping Model

COLORZ
Date:
Page No.

Spiral Model – The spiral model is similar to the incremental (Iterative waterfall model.) with more emphasis placed on risk analysis. The spiral model has four phases –

* Planning
* Risk Analysis
* Engineering Phase
* Evaluation phase

Planning Phase – Requirements are gathered during the Planning Phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications'

Risk Analysis – In the risk analysis pha a process is undertaken t identify risk and alternative solution A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analys then alternate solutions are suggeste and implemented.

Engineering Phase – In this phase softwa is developed along with testing at the end of the phase. Hence in this phase the development and testing is done.

Evaluation phase – This phase allows the customer to evalua the output of the project to date befo

continues to the next spiral.

Advantages of Spiral Model -

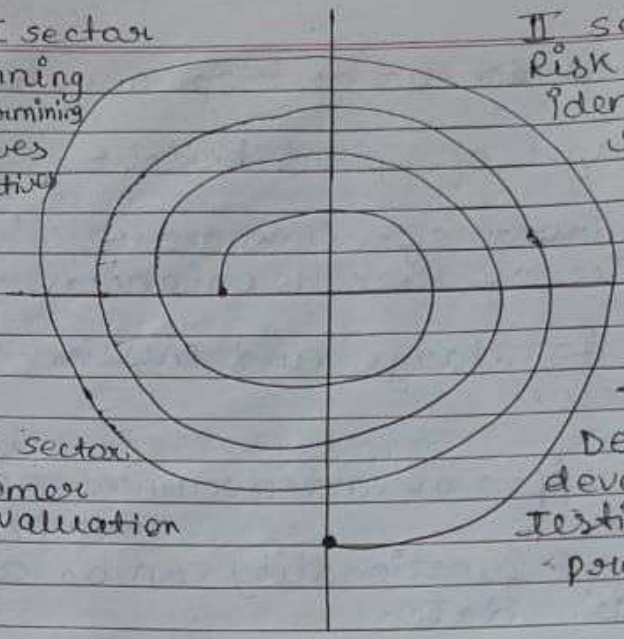
* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Strong approval and documentation control
* Additional functionality can be added at a later date.
* Software is produced early in the software life cycle.

Disadvantages of Spiral model.

* Can be a costly model to use
* Risk analysis requires highly specific expertise
* Projects success is highly dependent on the risk analysis phase.
* Doesn't work for smaller projects.
* It provides more flexibility
* This model is not widely used because it is relatively new.

I sector
Planning
i.e determining
objectives
alternatives
etc

II sector
Risk analysis i.e
identification and
resolving of
risks.



III sector
Development i.e
development and
Testing of the
product

IV sector
customer
evaluation

Spiral model

Iterative Waterfall Model –

* Iterative Waterfall Model has the same phases as the waterfall model but with fewer restrictions.
* To overcome the shortcomings of the waterfall model, a process model also known as Iterative waterfall Model was developed.
* In iterative waterfall model the full specification of requirements are not needed at the start.
* This model is also known as incremental model becaus we increment the requirements at each phase.
* This model overcomes the limitations of the original model by adding an iterative loop to the end of each cycle That is in order to keep changes in requirements the analysis phase is revisited at the end of each cycle.

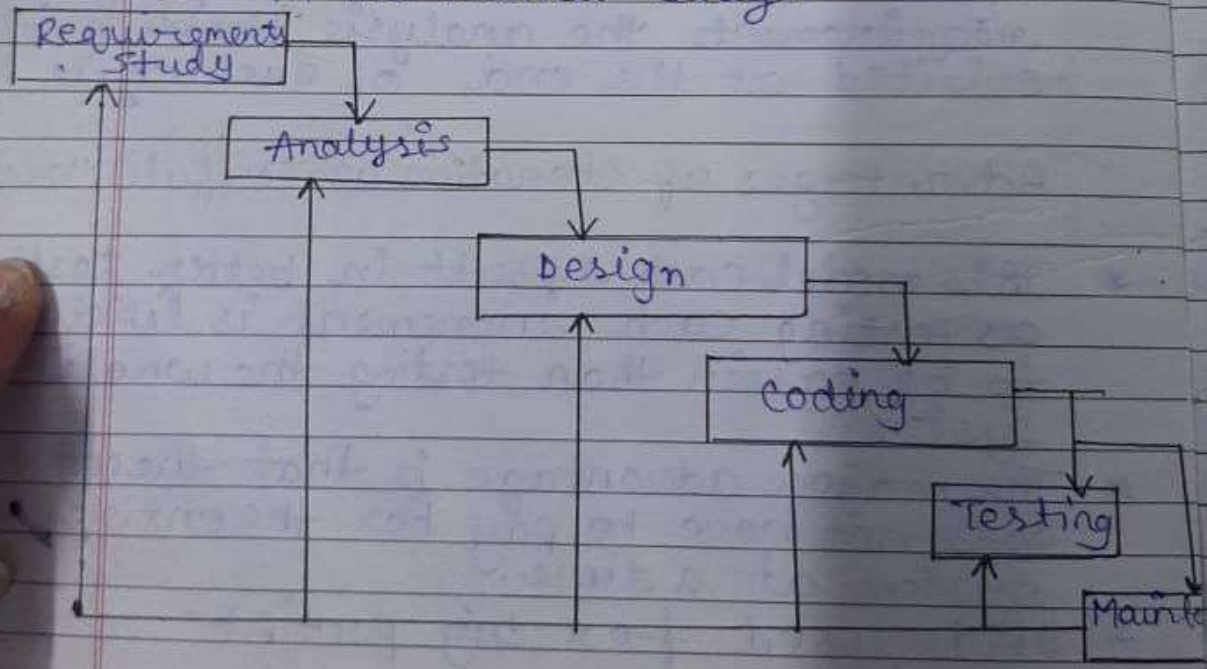
Advantages of Iterative Waterfall Model

* This model can result in better testing as testing each increment is likely to be easier than testing the whole system.
* The major advantage is that the costumer does not have to pay for the entire system at a time.
* It is used for big projects.

* In iterative model we are building and improving the product step by step. Hence we can track the defects step by step at early stage.
* In iterative model we can get the reliable user feedback.
* In this model less time is spent on documenting and more time is given for designing.

Disadvantages of iterative waterfall model –

* It is very difficult to calculate the cost of the entire project.
* Requirements are not always clear in the initial stage.



* Abstraction is the technique in which only those aspects of the problem that are relevant for certain purpose are taken.

* Decomposition is the technique in which a complex problem is divided into several subproblems and then all the subproblems are solved one by one.

* A program can be developed according to the programmer's individual style of development but a software product must be developed using the accepted software engineering principles.

Iterative Enhancement Model –

* This model combines the benefits of both waterfall model and prototype model.

* The basic idea is that the software begins by specifying and implementing a small part of the software.

* This software is then reviewed for further requirements. Then a list of additional requirements is prepared which helps to develop the new version of the software.

* For each cycle the same process is repeated to get a new version of the software.

* In this model four phases are repeated.

(i) Requirement Phase - In this phase, a list of requirements is prepared.

(ii) Design Phase -

* In this phase the software is designed according to the requirements.

* The design may be a new one or an extension to the earlier one.

(iii) Implementation phase - In this phase the software is implemented

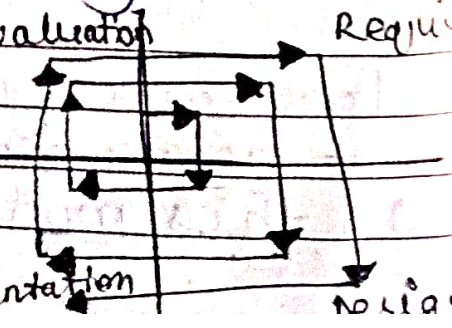
(iv) Evaluation Phase * In this phase, the software is evaluated the software is reviewed for further requirements.

* This model is useful if the core of the problem is well understood and increments can be easily defined and negotiated.

Advantages -

* Testing is better as testing at each iteration is easier than testing the whole system.

* The prototype is not thrown away but it is further refined according to the user's needs.



Agile Development Models - Software Engineering

In earlier days, the **Iterative Waterfall Model** was very popular for completing a project. But nowadays, developers face various problems while using it to develop software.

The main difficulties included handling customer change requests during project development and the high cost and time required to incorporate these changes. In the mid-1990s, the **Agile Software Development Model** was proposed to overcome these drawbacks of the **Waterfall Model**.

What is Agile Model?

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

Also, anything that is a waste of time and effort is avoided. The Agile Model refers to a group of development processes.

Steps in the Agile Model

The Agile Model is a combination of iterative and incremental process models. The phases involve in **Agile (SDLC) Model** are:

1. **Requirement Gathering**
2. **Design the Requirements**
3. **Construction / Iteration**
4. **Testing / Quality Assurance**
5. **Deployment**
6. **Feedback**

Agile Model Steps

1. Requirement Gathering

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

- Meet with the customer to really understand their needs and what they expect from the software.
- Identify the key requirements and business goals to make sure everyone is on the same page.
- Estimate how much time and effort it will take to develop the software.

- Assess if the project is technically possible and whether it's worth the investment from both a technical and economic standpoint.

2. Design the Requirements

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

- **Designing the system**: Once the requirements are gathered, the next step is to design the system's overall architecture based on those needs. This helps to verify the software is structured in a way that meets the user's expectations.
- **Creating wireframes**: Next, wireframes for the user interface (UI) are created. These are simple blueprints that show how the software will look and how users will interact with it, ensuring it's user-friendly and easy to navigate.
- **High-level design with UML diagrams**: At this stage, high-level designs using UML (Unified Modeling Language) diagrams are created to visually represent the software's structure and how different parts will work together.
- **Prototyping for feedback**: Prototypes are made to give stakeholders an early look at the software. This helps gather feedback early in the process and allows for adjustments before the full development begins.

3. Construction / Iteration

In this step, development team members start working on their project, which aims to deploy a working product. Each cycle typically consist between **1-4 weeks**, and at the end, the team delivers a working version of the software.

- **Development of Features**: The team works on the features identified during the requirement and design phases.
- **Coding and Implementation**: New functionalities are coded and integrated into the software based on the goals for that specific iteration.
- **Delivering a Working Product**: After each iteration, a usable version of the software is ready.
- **Incremental Improvement**: With every cycle, the software is enhanced, adding more features and refining existing ones.

4. Testing / Quality Assurance

Testing involves Unit Testing, Integration Testing, and System Testing, Which help in the agile development models:

- **Unit Testing:** Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.
- **Integration Testing:** Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.

- **System Testing:** Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. Deployment

In this step, the development team will deploy the working project to end users.

Once an iteration is finished and fully tested, the software is ready to be released to the end users. In Agile, deployment isn't a one-time event—it's an ongoing process. Updates and improvements are rolled out regularly, making sure the software keeps evolving and getting better with each release.

- Deploy the software to a test or live environment so that it can be used by customers or end-users.
- Make the software accessible to users, verifying they can start using it as expected.
- Verify the deployment goes smoothly and fix any issues that come up quickly

6. Feedback

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

- **Take feedback** from customers, users, and stakeholders after each iteration.
- Understand how well the product meets user needs and identify areas for improvement.
- **Check for bugs** or issues that need fixing.
- **Make adjustments** to the development plan based on feedback to improve the product further.

Agile Methodologies in Software Engineering

Agile is an umbrella term that includes various frameworks, each with its own way of implementing Agile principles. Some of the most popular Agile methodologies are:

1. Scrum

Scrum is one of the most widely used Agile methodologies. It organizes development work into time-boxed sprints, typically lasting 2–4 weeks, where small, deliverable increments of software are produced. It defines clear roles such as Scrum Master, Product Owner, and Development Team to maintain project focus and collaboration. Scrum teams conduct daily stand-up meetings to monitor progress, identify obstacles, and stay aligned. Each sprint concludes with a sprint review and retrospective, allowing teams to continuously improve their processes and outcomes.

2. Kanban

Kanban focuses on visualizing the entire workflow to optimize efficiency and ensure smooth progress. Teams use a Kanban board, usually divided into columns like “To-Do,” “In Progress,” and “Done,” to manage tasks and monitor status at a glance. Work items are “pulled” based on team capacity, preventing overload and promoting a steady delivery pace. By reducing bottlenecks and identifying process inefficiencies, Kanban helps teams achieve continuous, predictable flow in software development.

3. Extreme Programming (XP)

Extreme Programming (XP) is an Agile methodology that emphasizes technical excellence and high-quality software through practices like frequent releases, pair programming, and test-driven development (TDD). XP fosters a collaborative environment where developers work closely together, continuously testing and improving their code. The focus on small, rapid releases ensures that feedback is quickly incorporated, and high-quality standards are maintained throughout the development cycle.

4. Lean Development

Lean Development, inspired by lean manufacturing principles, aims to eliminate waste, maximize value, and optimize development processes. It promotes delivering features faster by minimizing unnecessary work, encouraging continuous learning, and maintaining close interaction with customers. Lean Development emphasizes efficiency without sacrificing quality, helping teams deliver better software in less time.

5. Feature-Driven Development (FDD)

Feature-Driven Development (FDD) is a client-centric Agile methodology that structures the development process around building valuable features. It emphasizes short, feature-based iterations, ensuring that each addition directly benefits the user. In FDD, the overall model is created first, and then small, client-valued features are developed systematically, leading to a scalable and well-organized software product.

Advantages of Agile Model in Software Engineering

a) Faster Delivery

Agile’s iterative approach ensures that functional parts of the software are delivered quickly and frequently. By breaking down the project into small, manageable chunks, teams can provide working versions of the product faster, allowing early detection of issues and quicker value delivery to customers.

b) Flexibility

One of the biggest strengths of Agile is its ability to adapt to changing requirements. Even if changes are requested late in the development process, Agile teams can easily adjust their priorities and plans without causing major disruptions, ensuring the final product better matches user needs.

c) Improved Collaboration

Agile fosters a culture of constant communication among developers, testers, and customers. Daily meetings, sprint reviews, and regular feedback loops create a collaborative environment where everyone stays aligned and works together to achieve project goals efficiently.

d) Higher Quality

Continuous integration and regular testing are central practices in Agile, which lead to early identification and correction of defects. This ongoing focus on quality helps produce a more stable, reliable, and high-performing product by the time of final release.

e) Customer Satisfaction

Agile emphasizes involving customers throughout the development process. Frequent demos and feedback sessions ensure that the software evolves according to user expectations, resulting in a final product that satisfies customer needs and builds stronger client relationships.

Challenges of Agile Model in Software Engineering

a) Requires Close Collaboration

Agile demands active participation from all team members and stakeholders. Without consistent engagement, communication gaps can occur, leading to misunderstandings, missed requirements, and a decline in project quality and efficiency.

b) Difficult to Predict Costs & Time

Because Agile welcomes changes at any stage, it becomes difficult to estimate the total time and cost at the start of the project. Frequent adjustments to scope and features can result in unpredictable budgets, making financial planning more challenging.

c) Not Suitable for Large, Complex Projects

Managing very large projects with numerous teams can be difficult under Agile. Coordination among multiple groups becomes complicated, and without strong leadership and communication strategies, the Agile process may lose effectiveness on a big scale.

d) Frequent Changes Can Lead to Scope Creep

While Agile encourages flexibility, uncontrolled or excessive changes can lead to scope creep, where the project's size and complexity gradually grow beyond the original plan. Without careful management, this can delay delivery and strain resources.

Models in Software Engineering

1. Waterfall Model in Software Engineering

The **Waterfall Model** is one of the earliest and most widely known **software development life cycle (SDLC)** models. It follows a **linear and sequential** approach, where each phase must be completed before the next begins. It is often referred to as a **plan-driven** or **traditional** model.

Phases of the Waterfall Model

1. **Requirements Analysis:**
    - Collect and document all software requirements.
    - Understand what the client needs.
    - Output: Software Requirement Specification (SRS).
2. **System Design:**
    - Translate requirements into system architecture and design.
    - Define data structures, software architecture, and interfaces.
3. **Implementation (Coding):**
    - Developers write the actual code based on the design.
    - The system is developed as per specifications.
4. **Integration and Testing:**
    - Test the software for defects and verify that it meets requirements.
    - Integration testing ensures modules work together.
5. **Deployment:**
    - Install the software on the client's environment.
    - Make the system operational.
6. **Maintenance:**
    - Fix bugs and issues that arise after deployment.
    - Enhance or modify the system based on user feedback.

Key Characteristics:

- Each phase depends on the deliverables of the previous one.
- No overlap between phases.
- Documentation-heavy approach.
- Best suited for projects with well-defined requirements.

Advantages of Waterfall Model:

- Simple and easy to understand.
- Clear milestones and deliverables for each phase.
- Good for small projects with stable requirements.

- Easy to manage due to structured approach.

Disadvantages of Waterfall Model:

- Difficult to accommodate changes after a phase is completed.
- Late discovery of issues (testing happens only after implementation).
- High risk if requirements are unclear or change frequently.
- Delayed delivery of working software (users see results late).

When to Use the Waterfall Model?

- When requirements are clear, fixed, and unlikely to change.
- For small or medium-sized projects.
- When technology used is well understood.
- When strict documentation and regulatory compliance are needed (e.g., defense, healthcare projects).

Comparison with Iterative/Incremental Models:

- **Waterfall:** Linear, one phase after another, no going back.
- **Iterative/Incremental:** Develops software in repeated cycles; feedback-driven; more flexible for changes

A Waterfall Model Example

Spiral Model Real-World Example: Creating an Online Banking System

1. **Requirements analysis and specification phase**
   In order to determine the essential features of the online banking system, including account management, fund transfers, bill payments, and loan applications, this phase will be responsible for compiling all available data on customer banking requirements, transactions, and security protocols.
2. **Design Phase**
   Fine-tuning the parameters set in the analysis phase is the main focus of the design phase in this Waterfall Model example. The architecture of the system will be created to guarantee excellent speed, prevent transactional mistakes, and securely handle sensitive data. To safeguard user accounts, this involves multi-factor authentication, encryption techniques, database architecture, and UI design.
3. **Implementation**

In order to determine how accurately the online banking system can handle transactions, balance inquiries, cash transfers, and bill payments, this crucial step entails doing dummy runs of the system using a preliminary set of banking transactions and user data. These findings are to be compared with those of banking specialists and auditors who guarantee adherence to banking laws and transaction correctness.

4. **Testing**
   As with any Waterfall Model example, the testing phase's goal is to make sure the online banking system's features all work as intended. Testing for security flaws, transaction correctness, performance under high load, and responsiveness of the user interface are all included in this. Tests of safe logins, data encryption, and making sure sensitive data is handled appropriately across the system are given particular focus.
5. **Maintenance**
   In addition to the anticipated addition of new features or modifications to banking rules, the online banking system should be examined in the last stage for any upgrades or modifications that could be needed. Security fixes, performance enhancements, and the introduction of new services like mobile banking, fast loans, or tailored financial advice will all require regular

2. What is Prototyping Model?

The Prototyping Model is one of the most popularly used **Software Development Life Cycle Models (SDLC models)**. This model is used when the customers do not know the exact project requirements beforehand.

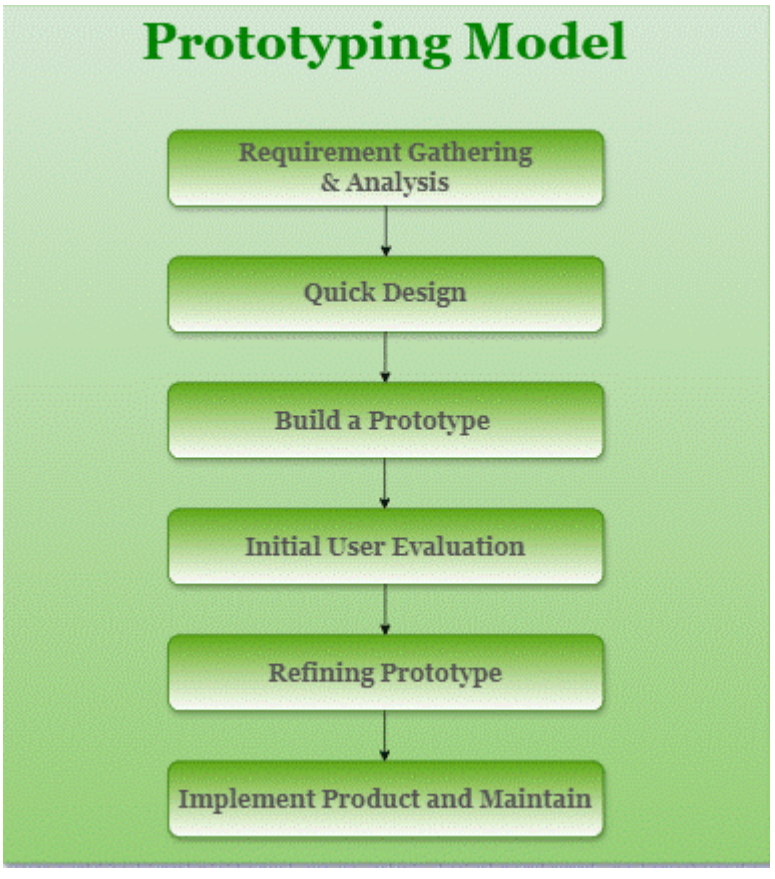
In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

In this process model, the system is partially implemented before or during the analysis phase thereby allowing the customers to see the product early in the life cycle. The process starts by interviewing the customers and developing the incomplete high-level paper model.

This document is used to build the initial prototype supporting only the basic functionality as desired by the customer. Once the customer figures out the problems, the prototype is further refined to eliminate them. The process continues until the user approves the prototype and finds the working model to be satisfactory.

Phases of Prototyping Model

Prototyping Model has six phases as follows:



Prototyping Model Phases

1. Requirements gathering and analysis

Requirement analysis is the first step in developing a prototyping model. During this phase, the system's desires are precisely defined. During the method, system users are interviewed to determine what they expect from the system.

2. Quick design

The second phase could consist of a preliminary design or a quick design. During this stage, the system's basic design is formed. However, it is not a complete design. It provides the user with a quick overview of the system. The rapid design aids in the development of the prototype.

3. Build a Prototype

During this stage, an actual prototype is intended to support the knowledge gained from quick design. It is a small low-level working model of the desired system.

4. Initial user evaluation

The proposed system is presented to the client for preliminary testing at this stage. It is beneficial to investigate the performance model's strengths and weaknesses. Customer feedback and suggestions are gathered and forwarded to the developer.

5. Refining prototype

If the user is dissatisfied with the current model, you may want to improve the type that responds to user feedback and suggestions. When the user is satisfied with the upgraded model, a final system based on the approved final type is created.

6. Implement Product and Maintain

The final system was fully tested and distributed to production after it was developed to support the original version. To reduce downtime and prevent major failures, the programmer is run on a regular basis.

Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.

- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.
- Early feedback from customers and stakeholders can help guide the development process and ensure that the final product meets their needs and expectations.
- Prototyping can be used to test and validate design decisions, allowing for adjustments to be made before significant resources are invested in development.
- Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.
- Prototyping can facilitate communication and collaboration among team members and stakeholders, improving overall project efficiency and effectiveness.
- Prototyping can help bridge the gap between technical and non-technical stakeholders by providing a tangible representation of the product.

Disadvantages of the Prototyping Model

- Costly concerning time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.
- The prototype may not be scalable to meet the future needs of the customer.
- The prototype may not accurately represent the final product due to limited functionality or incomplete features.
- The focus on prototype development may shift away from the final product, leading to delays in the development process.
- The prototype may give a false sense of completion, leading to the premature release of the product.

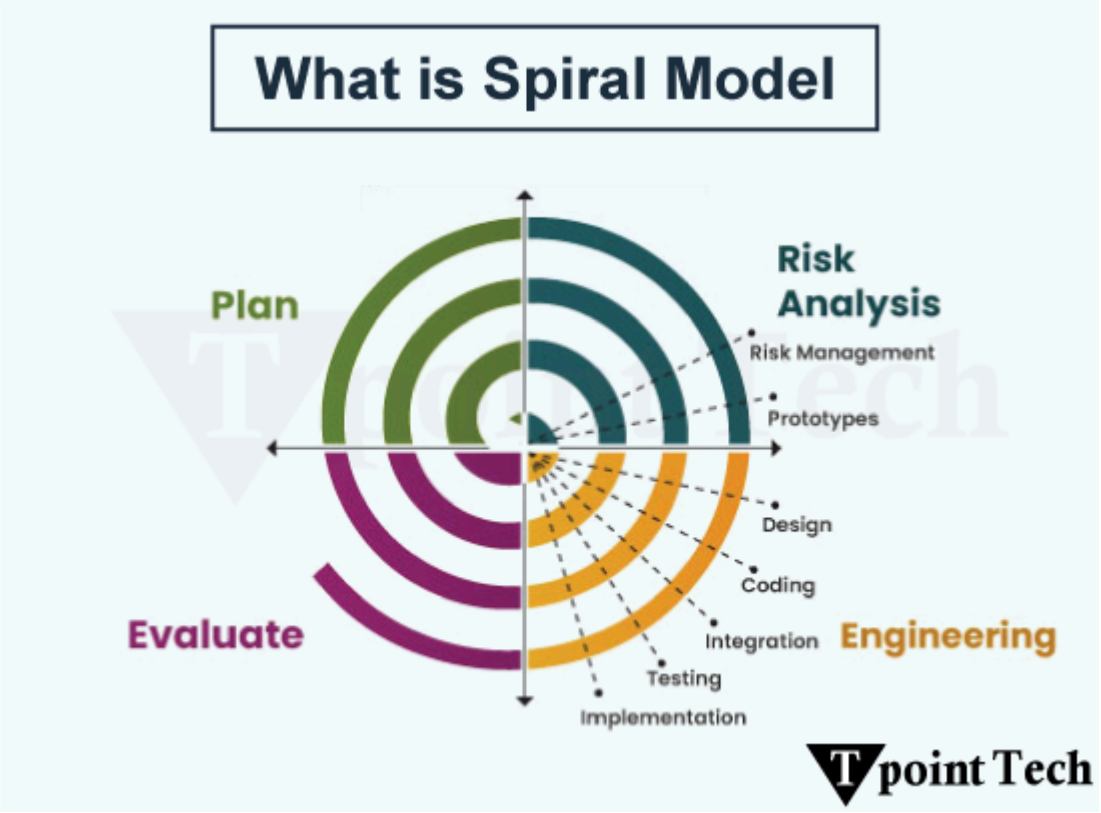
- The prototype may not consider technical feasibility and scalability issues that can arise during the final product development.
- The prototype may be developed using different tools and technologies, leading to additional training and maintenance costs.
- The prototype may not reflect the actual business requirements of the customer, leading to dissatisfaction with the final product.

Applications of Prototyping Model

- The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
- The prototyping model can also be used if requirements are changing quickly.
- This model can be successfully used for developing user interfaces, high-technology software-intensive systems, and systems with complex algorithms and interfaces.
- The prototyping Model is also a very good choice to demonstrate the technical feasibility of the product.

3. Spiral Model in Software Engineering

The spiral model combines the iterative development process model and aspects of the Waterfall model. It is a systems development lifecycle (SDLC) approach for risk management. Software developers employ the spiral model, which is preferred for complex, large-scale projects.



The spiral model of software development resembles a coil with several loops when seen as a diagram. Depending on the project, the number of loops is determined by the project manager**. Every spiral loop represents a stage in the model of the software development process.** Through each stage of the spiral, a software product may be refined and released gradually thanks to the spiral model. Additionally, this risk-driven strategy makes it possible to construct prototypes at every stage. Making a prototype enables the model to handle possible risks once the project has started, which is its most crucial characteristic.

Each cycle in the spiral is divided into four parts:

- **Objective setting:** Each cycle in the spiral starts with the identification of the purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exist.
- **Risk Assessment and Reduction:** The next phase in the cycle involves calculating these various alternatives based on the goals and constraints. The focus of evaluation in this stage is on the project's risk perception.
- **Development and validation:** The next phase involves developing strategies to resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.
- **Planning:** Finally, the next step is planned. The project is reviewed, and a choice is made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, suppose performance or user-interface risks are treated more as essential as program development risks. In that case, the next phase may be evolutionary development, which includes developing a more detailed prototype to solve the risks.

The **spiral model's risk-driven** feature allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or other type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

The Spiral Model's Steps

Each quadrant is further subdivided into stages, even if the phases are divided into quadrants. The following are the steps in the spiral model:

1. The needs for the new system are outlined as thoroughly as feasible. For this, many users who represent all internal and external interests, as well as other facets of the current system, are typically interviewed. A draft design is made for the new system.
2. The first design is used to build the new system's prototype. In most cases, this is a simplified system that approximates the features of the finished software.
3. The process of creating a second prototype involves four steps: (1) assessing the risks, flaws, and strengths of the prototype; (2) establishing the second prototype's requirements; (3) planning and designing the second prototype; and (4) building and testing the second prototype.
4. If the danger is judged to be too high, the project is terminated. Overruns in development costs, inaccurate operational cost estimates, and other elements that might lead to a subpar product are examples of risk factors.
5. The current prototype is assessed in the same way as the prior prototype, and if required, a new prototype is created using the four steps mentioned above.

6. The previous processes are repeated until the client is happy that the improved prototype accurately depicts the intended final product.
7. Based on the improved prototype, the finished product is built.
8. The finished product undergoes extensive testing and evaluation. Continuous routine maintenance is performed to save downtime and avoid major malfunctions.

Examples of Spiral Model Projects in the Real world

A variety of sectors use the spiral approach to iteratively enhance projects. Here are a few examples:

- **Creation of software:** Software projects are tested iteratively by developers who follow user input to direct enhancements. This is particularly true for mobile apps, whose functionality is subject to quick changes and necessitates debugging in order to meet stakeholder and user expectations.
- **Gaming:** Before releasing a finished product, game makers evaluate gameplay and enhance graphics using this iterative methodology. These improvements are also based on customer input.
- **Shop:** Based on user preferences and industry trends, e-commerce website developers employ spiral modeling to continually improve the client experience by adding new features.
- **Medical care:** The spiral model ensures that electronic health record systems adhere to current laws, such as the Health Insurance Portability and Accountability Act and industry standards.
- **Space:** Before being deployed in space, space exploration technologies like satellites and rovers are prototypes that are tested through simulations. The spiral model guides their growth to ensure they are not prone to problems.

Benefits of the Spiral Model

The spiral model is an excellent choice for complicated, large-scale projects. The model's progressive structure enables developers to divide large projects into smaller ones and work on each feature separately, making sure nothing is overlooked. Because the prototype building is completed gradually, it might occasionally be simpler to estimate the project's overall cost.

The following are some additional advantages of the spiral model:

1. **Adaptability:** After work has begun, requirements changes are readily accepted and integrated.
2. **Control of risks:** By incorporating risk analysis and management into each stage, the spiral approach enhances security and increases the likelihood of preventing breaches and assaults. Risk reduction is also made easier by the iterative development process.
3. **Client contentment:** The spiral model facilitates customer feedback. If the program is being created for the customer, the customer can view and assess their product at

every stage. This saves the development team time and money by allowing them to voice concerns and seek adjustments prior to the product being completely produced.

Challenges with the spiral model

The spiral model has the following drawbacks:

1. **Expensive:** Due to its high cost, the spiral model is not appropriate for minor projects.
2. **Reliance on risk assessment:** Effective risk management is necessary for a project to be completed successfully. Therefore, project participants must possess proficiency in risk assessment.
3. **Complexity:** Compared to other SDLC choices, the spiral model is more complicated. Protocols must be strictly adhered to for it to function well. Moreover, more documentation is needed to monitor the intermediate stages.
4. **Difficulties in time management:** Time management is nearly hard as the number of necessary stages is sometimes unknown before the project begins. As a result, there's always a chance of running over budget or behind time.

OR

The **Spiral Model** is a **risk-driven software development model** that combines features of the **Waterfall model** and **Iterative/Prototyping approaches**. It is especially useful for **large, complex, and high-risk projects** where requirements are not fully known at the start.

It was proposed by **Barry Boehm** in 1986.

Key Idea of the Spiral Model

- The project is developed in **spirals (cycles)** rather than a single linear path.
- Each spiral consists of **four main activities**, and every loop adds more detail to the system.
- **Risk assessment and mitigation** are central to each cycle.

Phases (Activities) in Each Spiral Loop

1. **Planning (Determine Objectives):**
    - Identify project objectives, alternatives, and constraints.
    - Define what to achieve in this iteration.
2. **Risk Analysis (Identify & Resolve Risks):**
    - Analyze and identify potential risks for the iteration.
    - Explore alternatives and strategies to mitigate these risks.
    - Often involves building prototypes to reduce uncertainty.

3. **Engineering (Development & Testing):**
    - Develop and verify the product for this iteration.
    - Includes coding, testing, integration of features.
4. **Evaluation (Customer Review & Feedback):**
    - Stakeholders evaluate the progress.
    - Provide feedback to refine objectives for the next spiral.

Each cycle results in either a **prototype** or a **working version** of the product, with the level of detail increasing in every loop.

Key Characteristics:

- Focuses on **risk management**.
- Incorporates **iterative development** (like in Agile) with **structured planning** (like in Waterfall).
- Customer involvement in every cycle ensures evolving requirements are addressed.

Advantages:

- Excellent for large and high-risk projects.
- Allows for changing requirements.
- Early risk detection and mitigation.
- Customer feedback integrated at each stage.

Disadvantages:

- Can be expensive and time-consuming due to repeated evaluations.
- Requires expertise in **risk analysis**.
- More complex to manage compared to Waterfall or Iterative models.

When to Use the Spiral Model?

- When requirements are unclear or expected to evolve.
- For large, mission-critical, and high-risk projects (e.g., defense, aerospace, banking systems).
- When continuous risk assessment is essential.

Comparison with Other Models:

- **Waterfall:** Linear; no feedback until the end.
- **Iterative:** Cyclic but lacks systematic risk analysis.
- **Spiral:** Cyclic **with explicit risk management** and customer feedback.

4. Evolutionary Model in Software Engineering

This evolutionary model concept comes into the picture after the user faces the partially developed system rather than waiting for the fully developed version. The idea of this evolutionary model comes from developing the core module and then improving the software product by using iterative and incremental techniques with appropriate feedback.

In this evolutionary process model, the software product is in the sensitive version, which is made through many iterations, and then the final product is prepared. This evolutionary approach also suggests breaking down all the models into maintainable smaller chunks. After that, we can prioritize all the smaller products. After completing all the products, we can deliver those chunks one at a time to the users.

It plays a vital role in large projects where we can easily search the module for the implementation of incremental things. It is also an influential model because it collects customer feedback throughout the development process, and the customer can use the base feature of the project before the release of the entire working version.

What is the Evolutionary Model?

The evolutionary model is also known as successive versions or incremental models. The main aim of this evolutionary model is to deliver the products in total processes over time. It also combines the iterative and collective model of the software development life cycle (SDLC).

Based on the evolutionary model, we can divide the development model into many modules to help the developer build and transfer incrementally. On the other hand, we can also develop the skeleton of the initial product. Also, it refines the project to increase levels of capability by adding new functionalities in successive versions.

Characteristics of the Evolutionary Model

There are so many characteristics of using the evolutionary model in our project. These characteristics are as follows.

- We can develop the evolutionary model with the help of an iterative waterfall model of development.
- There are three types of evolutionary models. These are the Iterative model, Incremental model and Spiral model.

- Many primary needs and architectural planning must be done
- to implement the evolutionary model.
- When the new product version is released, it includes the new functionality and some changes in the existing product, which are also released with the latest version.
- This model also permits the developer to change the requirement, and the developer can divide the process into different manageable work modules.
- The development team also have to respond to customer feedback throughout the development process by frequently altering the product, strategy, or process.

5. Spiral Model

This model is also known as the risk-driven process model. With the help of this spiral model, we can generate the software project. During the risk analysis, if a risk is found, an alternate solution should be suggested and implemented in the spiral model. We can also say that it combines the sequential or waterfall models and the prototype models. All the required activities are done in one iteration. The output of the large project is small.

The framework activities of the spiral model are as shown below.

Advantages of the Spiral Model

There are so many benefits of using the spiral model. These are as follows:

- The amount of risk should be reduced in this model.
- This model is best for large and critical projects.
- It provides the developer with solid approval and documentation control.
- The software production time is less in this spiral model.

Disadvantages of Spiral Model

It also has some disadvantages to using the spiral model. These are as follows.

- We need a lot of financial support to develop a project.
- We should not use this model for a very small project.

Iterative Enhancement Model

The **Iterative Enhancement Model** (also called the **Iterative Development Model** or **Incremental Model**) is a software engineering process model where the software is developed and delivered in small, manageable portions (iterations) rather than building the

complete system all at once. Each iteration adds new functionality or improves existing functionality until the final system is complete.

Key Characteristics of the Iterative Enhancement Model:

1. **Incremental Development:**
   The software is divided into small modules or components that are developed and delivered step by step.
2. **Feedback-Oriented:**
   After each iteration, feedback is taken from stakeholders/users and used to refine requirements and improve the next iteration.
3. **Progressive Enhancement:**
   The product evolves with each iteration, incorporating additional features and improvements.
4. **Parallel Activities:**
   Requirements analysis, design, coding, and testing can overlap across iterations.

Phases of Iterative Enhancement Model:

Each iteration typically includes:

1. **Requirements Gathering:** Identify the features for the current iteration.
2. **Design:** Prepare or refine the system architecture for those features.
3. **Implementation:** Develop and integrate the new features.
4. **Testing:** Verify the correctness and performance of the iteration.
5. **Deployment and Feedback:** Deliver the iteration to the users for review and gather feedback for improvement.

Advantages:

- Early delivery of working software (improves customer satisfaction).
- Easier to manage risk as problems can be identified early.
- Requirements can evolve over time; flexible for changes.
- Continuous user feedback leads to higher quality.

Disadvantages:

- Requires active user involvement throughout the process.
- May lead to scope creep if changes are uncontrolled.

- Requires careful planning and design to avoid integration issues.
- May not be suitable for very small projects with fixed requirements.

Examples of Use:

- Agile methodologies (like Scrum) follow iterative and incremental principles.
- Large systems where requirements are expected to evolve, e.g., banking software, ERP systems.

When to Use Iterative Enhancement Model?

- When requirements are not fully understood at the start.
- When customer feedback is critical to the success of the product.
- When project complexity makes it difficult to deliver in one go.

Iterative enhancement model in software engineering

The **Iterative Enhancement Model** is a **software development approach** in which the system is developed **incrementally through multiple iterations**. Each iteration adds new functionality or enhances the existing system until the final product meets all requirements.

Key Features:

1. **Incremental Development** – Software is built in small, manageable parts.
2. **Feedback-Oriented** – After each iteration, feedback is gathered from users or stakeholders.
3. **Risk Reduction** – Problems are detected early since partial systems are delivered frequently.
4. **Adaptability** – Requirements can be refined as development progresses.
5. **Testing at Every Step** – Each version is tested before moving to the next iteration.

Phases of Iterative Enhancement Model:

1. **Requirements Analysis** – Identify core requirements.
2. **Design** – Create initial architecture and design for the first iteration.

3. **Implementation (Coding)** – Develop a small functional module.
4. **Testing & Evaluation** – Test and gather feedback.
5. **Enhancement** – Improve and add features based on feedback.
6. **Repeat** – Continue until the final product is ready.

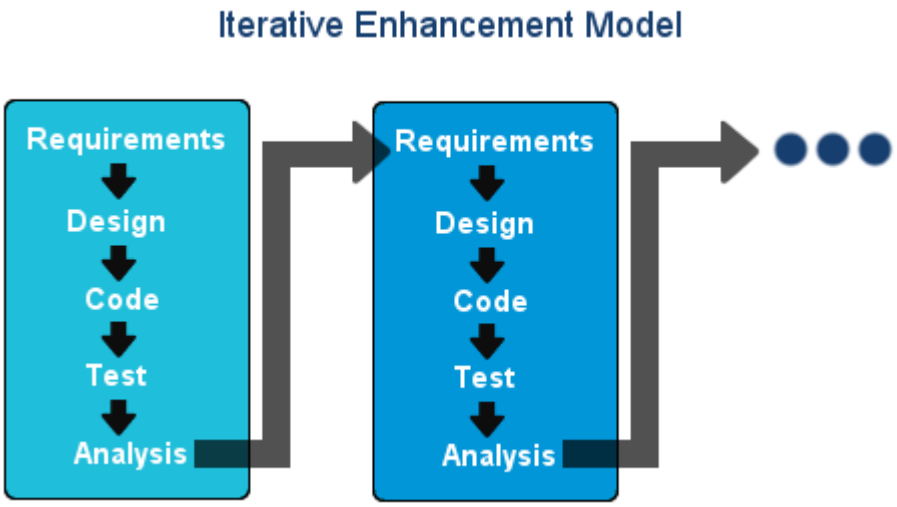


Diagram of Iterative Enhancement Model:

Advantages:

- Handles **changing requirements** effectively.
- Delivers **usable software early**.
- Easier to **detect and fix errors** early.
- Involves users at every stage.

Disadvantages:

- Requires **constant communication** with users.
- May lead to **scope creep** if changes are uncontrolled.
- Iterations can extend overall schedule if not managed well.

What is an Agile Model?

The Agile Model is an incremental and iterative process of software development. It defines each iteration's number, duration, and scope in advance. Every iteration is considered a short "frame" in the Agile process model, which mostly lasts from two to four weeks.

Agile Model divides tasks into time boxes to provide specific functionality for the release. Each build is incremental in terms of functionality, with the final build containing all the attributes. The division of the entire project into small parts helps minimize the project risk and the overall project delivery time.

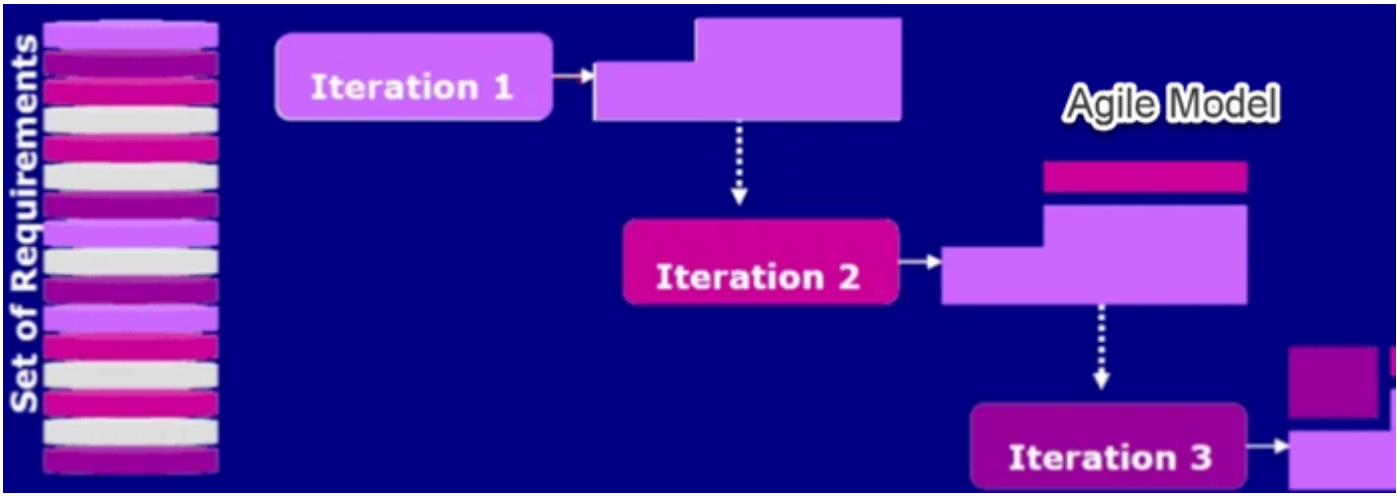


Table of Content:

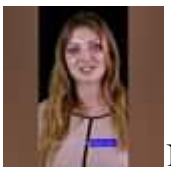
What are the important Agile Model Manifestos?

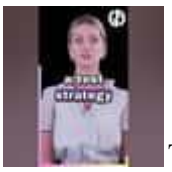
Here is the essential manifesto of the Agile Model:

- Individuals and interactions are given priority over processes and tools.
- Adaptive, empowered, self-organizing team.
- Focuses on working software rather than comprehensive documentation.
- Agile Model in software engineering aims to deliver complete customer satisfaction by rapidly delivering valuable software.
- Welcome changes in requirements, even late in the development phase.
- Daily co-operation between businesspeople and developers.

- Priority is customer collaboration over contract negotiation.
- It enables you to satisfy customers through early and frequent delivery.
- A strong emphasis is placed on face-to-face communication.
- Developing working software is the primary indicator of progress.
- Promote sustainable development pace.
- A continuous focus is placed on technical excellence and sound design.
- An improvement review is conducted regularly by the team.

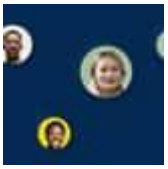
EXPLORE MORE What to bring to a JOB Interview: Interview Tips01:50


Naive Bayes Classifier in Machine Learning01:00


Test Strategy in Software Testing01:00


5 BEST Funded Trader Programs & Firms 💰13:58

What to wear in interview for Women | Working women01:44


5 Best Virtual Business Phone Number Apps (FREE Options)08:43


5 BEST CRM Software for Small Business (🆓 Options)10:31

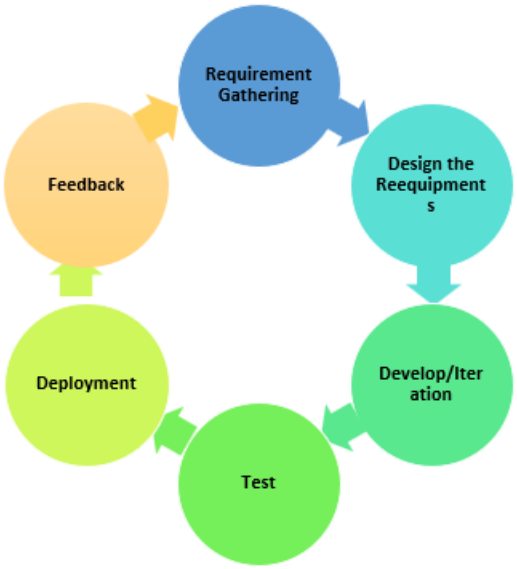

5 BEST Network Monitoring Tools (Free/Paid)08:45


How to Find Unknown Person Name and Details with Just a Picture04:48


Defect Management Process01:00The video player is currently playing an ad.

Phases of Agile Model

Here are the different phases of Agile:



Here are the important stages involved in the Agile Model process in the SDLC life cycle:

- **Requirements Gathering:** In this Agile model phase, you must define the requirements. The business opportunities and the time and effort required for the project should also be discussed. By analyzing this information, you can determine a system's economic and technical feasibility.
- **Design the Requirements:** Following the feasibility study, you can work with stakeholders to define requirements. Using the UFD diagram or high-level UML diagram, you can determine how the new system will be incorporated into your existing software system.
- **Develop/Iteration:** The real work begins at this stage after the software development team defines and designs the requirements. Product, design, and development teams start working, and the product will undergo different stages of improvement using simple and minimal functionality.
- **Test:** This phase of the Agile Model involves the testing team. For example, the Quality Assurance team checks the system's performance and reports bugs during this phase.
- **Deployment:** In this phase, the initial product is released to the user.
- **Feedback:** After releasing the product, the last step of the Agile Model is feedback. In this phase, the team receives feedback about the product and works on correcting bugs based on the received feedback.

Compared to Waterfall, Agile cycles are short. There may be many such cycles in a project. The phases are repeated until the product is delivered.

Types of Agile

Here are some important Agile Types:

Scrum: This agile method focuses primarily on managing tasks in team-based development conditions. In the Scrum Agile model, the team should strictly follow a work plan for each Sprint. Moreover, people involved in this type of project have predefined roles.

Crystal: Using Crystal methodology is one of the most straightforward and most flexible approaches to developing software, recognizing that each project has unique characteristics. Therefore, policies and practices need to be tailored to suit them.

Crystal methodologies are categorized as below:

- **CLEAR:** User for small and low critical efforts.
- **ORANGE:** User for moderately larger and critical projects.
- **ORANGE WEB:** Typically, electronic business

Dynamic Software Development Method (DSDM): This Rapid Application Development (RAD) approach involves active user involvement, and the teams are empowered to make decisions with the goal of frequent product delivery.

Feature Driven Development (FDD): This Agile method focuses on "designing & building" features. It is divided into several short phases of work that must be completed for each feature separately. It includes domain walkthrough, design inspection, code inspection, etc.

Lean Software Development: This methodology is based on the principle of "Just-In-Time Production." It helps to increase the speed of software development and decrease costs.

As a result of a lean development model, waste is eliminated, learning is amplified, early delivery is achieved, and integrity is built.

Extreme Programming (XP): Extreme Programming is a useful Agile model when there are constantly changing requirements or demands from clients. It is also used when there is no sure about the system's functionality.

When to use the Agile Model?

Here are the common scenarios where the Agile method is used:

- It is used when there are frequent changes that need to be implemented.
- Low-regulatory-requirement projects
- Projects with not very strict existing process
- Projects where the product owner is highly accessible
- Projects with flexible timelines and budget

Advantages of the Agile Model

Here are some common pros/benefits of the Agile Model:

- Communication with clients is on a one-on-one basis.
- Provides a very realistic approach to software development

- Agile Model in software engineering enables you to draft efficient designs and meet the company's needs.
- Updated versions of functioning software are released every week.
- It delivers early partial working solutions.
- Changes are acceptable at any time.
- You can reduce the overall development time by utilizing this Agile Model.
- It allows concurrent development and delivery within an overall planned context.
- The final product is developed and available for use within a few weeks.

Disadvantages of Agile Model

Here are some common cons/drawbacks of the Agile Model:

- There is a higher risk of sustainability, maintainability, and extensibility.
- In some corporations, self-organization and intensive collaboration may not be compatible with their corporate culture.
- Documentation and design are not given much attention.
- Without clear information from the customer, the development team can be misled.
- Not a suitable method for handling complex dependencies.

Agile Model Vs. Waterfall Model

Agile and Waterfall models are two different methods for the software development process. Despite their differences in approach, both methodologies can be used at times, depending upon the project and the requirements.

Agile Model	**Waterfall Model**
Agile methodologies propose incremental and iterative approaches to software design	Software development flows sequentially from start point to end point.
The Agile Model in software engineering is broken into individual models that designers work on	The design process is not broken into individual models
The customer has early and frequent opportunities to look at the product and make decisions and changes.	The customer can only see the product at the end of the project.
The Agile Model is considered unstructured compared to the waterfall model	Waterfall models are more secure because they are plan oriented
Small projects can be implemented very quickly. For large projects, it isn't easy to estimate the development time.	All sorts of the project can be estimated and completed.

Agile Model	Waterfall Model
Test plan is reviewed after each Sprint	The test plan is hardly discussed during the test phase.

Summary

- The Agile Model is an incremental and iterative process of software development.
- It focuses on working software rather than comprehensive documentation.
- Agile model is divided into various stages like 1) Requirements Gathering, 2) Design the Requirements, 3) Develop/Iteration, 4) Test, 5) Deployment 6) Feedback.
- Various types of Agile types are:

  1) Scrum,

  2) Crystal,

  3) Dynamic Software Development Method (DSDM):

  4) Feature Driven Development (FDD),

  5) Lean Software Development

  6) Extreme Programming (XP).
- The agile model is used when frequent changes need to be implemented.
- It provides a very realistic approach to software development
- This model has a greater risk of sustainability, maintainability, and extensibility.
- Agile methodologies in Software Testing adopt incremental and iterative approaches to software design, whereas software development flows sequentially from the starting point to the endpoint.