UNIT-1

PART A

1 Explain is legacy software?

Legacy Software can be defined as

- Older programs are often referred to as legacy software.
- Lagacy software systems were developed decades ago and have been continuously modified to meet changes in business requirements and computing platforms.
- Many legacy systems remain supportive to core business function and are indispensable to the business.
- Legacy software is characterized by longevity and business criticality.

2. Define Software?

Software is defined as

(1) instructions (computer programs) that when executed provide desired features, function, and performance.

(2) data structures that enable the programs to adequately manipulate information, and

(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

3.Demonstrate all the applications of software?

The 7 broad categories of computer software application domain present they are:

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web-applications
- Artificial intelligence software.

4. List the types of software myths?

1. **Management myths:** Manages with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.
2. **Customer myths:** The customer believes myths about software because software managers and practitioners do little to correct misinformation. Myths lead to false expectations and ultimately, dissatisfaction with the developer.
3. **Practitioner's myths**: Myths that are still believed by software practitioners: during the early days of software, programming was viewed as an art from old ways and attitudes die hard.

5.Discuss the architecture of layered technology?

Software engineering is a layered technology. Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software engineering encompasses a process, methods for managing and engineering software, and tools.

The layers are:

- A quality focus.

- Process
- Methods
- Tools

6.List all the umbrella activities in process framework?
The activities are:

- Software project tracking and control
- Risk Management
- Software Quality Assurance
- Formal Technical Reviews
- Measurement
- Software configuration management
- Reusability management
- Work Product preparation and production.

7.Explain Process Framework?
A software process is a framework for the activities, actions, and tasks that are required to build high-quality software. Each framework activity is populated by a set of software engineering actions. Each software engineering action is defined by a task set that identifies the work tasks that are to be completed, the work products that will be produced, the quality assurance points that will be required, and the milestones that will be used to indicate progress.

8.List the types of software models?
SDLC Models are:

- Waterfall model
- Spiral Model
- Agile Model

9.What are the two representations of CMMI?
A representation allows an organization to pursue a different set of improvement objectives. There are two representations for CMMI:

- **Staged Representation**
- **Continuous Representation**

10. List the models in CMMI?
When people refer to CMMI, for example, they could be talking about any of the three CMMI models:

- **CMMI for Development (CMMI-DEV):** CMMI-DEV is used to improve engineering and development processes in an organization that develops products.
- **CMMI for Services (CMMI-SVC):** CMMI-SVC is used to improve management and service delivery processes in an organization that develops, manages, and delivers services.
- **CMMI for Acquisition (CMMI-ACQ):** CMMI-ACQ is used to improve supplier management processes in an organization that deals with multiple suppliers for its business.

11.Explain the levels in continuous representation in CMMI?

Continuous Representation:

- allows selection of specific process areas.
- uses capability levels that measure improvement of an individual process area.
- Continuous CMMI representation allows comparison between different organizations on a process-area-by-process-area basis.
- allows organizations to select processes that require more improvement.
- In this representation, the order of improvement of various processes can be selected which allows the organizations to meet their objectives and eliminate risks.

12. Explain staged representation in CMMI?

Staged Representation:

- uses a pre-defined set of process areas to define an improvement path.
- provides a sequence of improvements, where each part in the sequence serves as a foundation for the next.
- an improved path is defined by maturity level.
- maturity level describes the maturity of processes in organization.
- Staged CMMI representation allows comparison between different organizations for multiple maturity levels.

13.Write the other name of waterfall model and who invented waterfall model?

The Waterfall Model was the first Process Model to be introduced. It is also referred to as **linear-sequential life cycle model** and was first introduced by Winston W Royce in 1970.

14. What are the types of Agile models?

Types of Agile Model in Software Engineering

- Scrum
- Crystal
- Dynamic Software Development Method (DSDM)
- Feature Driven Development (FDD)
- Lean Software Development:
- Extreme Programming (XP
- Adaptive Software Development (ASD)
- Agile Unified Process (AUP)

Part B

1 Explain the evolving role of software?
Evolving Role of Software:
Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer. Software is a generic term used to refer to applications, scripts and programs that run on a device.
Software takes dual role. It is both a product and a vehicle for delivering a product.

As a product: It delivers the computing potential embodied by computer Hardware or by a network of computers.

As a vehicle: It is information transformer-producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as single bit or as complex as a multimedia presentation.

Software delivers the most important product of our time-information.
- It transforms personal data.
- It manages business information to enhance competitiveness.
- It provides a gateway to worldwide information networks.
- It provides the means for acquiring information.

The role of computer software has undergone significant change over a span of little more than 50 years.
- Dramatic Improvements in hardware performance.
- Vast increases in memory and storage capacity.
- A wide variety of exotic input and output options

2 Define software and explain the various characteristics of software?
Define Software
Software is:
(1) instructions (computer programs) that when executed provide desired features, function, and performance.
(2) data structures that enable the programs to adequately manipulate information, and
(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Characteristics Of Software:
1) ***Software is developed or engineered; it is not manufactured in the classical sense.*** Although some similarities exist between software development and hardware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are non-existent (or easily corrected) for software. Both activities require the construction of a "product," but the approaches are different.
2) ***Software does not —wear out***

FIGURE 1.1

Failure curve for hardware

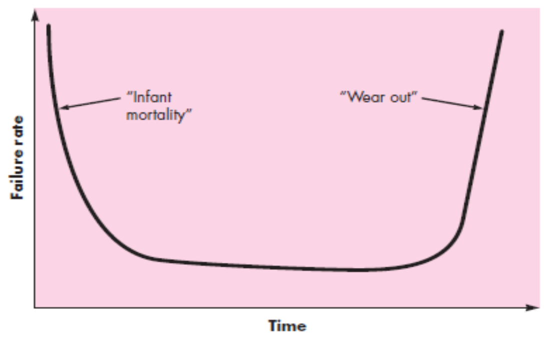
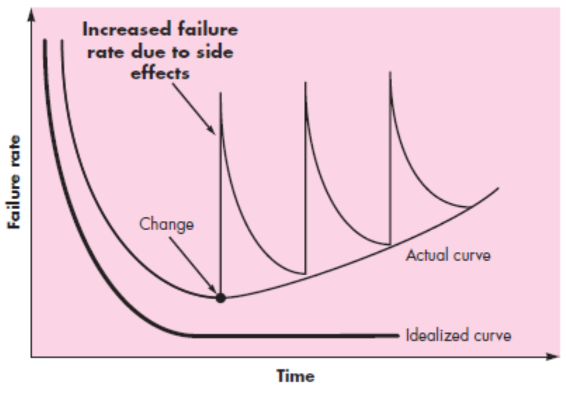


Figure 1.1 depicts failure rate as a function of time for hardware. The relationship, often called the "bathtub curve," indicates that hardware exhibits relatively high failure rates early in its life (these failures are often attributable to design or manufacturing defects); defects are corrected and the failure rate drops to a steady-state level (hopefully, quite low) for some period of time. As time passes, however, the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies. Stated simply, the hardware begins to wear out.

Software is not susceptible to the environmental maladies that cause hardware to wear out. In theory, therefore, the failure rate curve for software should take the form of the "idealized curve" shown in Figure 1.2. Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected and the curve flattens as shown. The idealized curve is a gross oversimplification of actual failure models for software. However, the implication is clear—software doesn't wear out. But it does deteriorate!

FIGURE 1.2

Failure curves for software



3) ***Although the industry is moving toward component-based construction, most software continues to be custom-built.***

A software component should be designed and implemented so that it can be reused in many different programs. Modern reusable components encapsulate both data and the processing that is applied to the data, enabling the software engineer to create new applications from reusable parts. In Hardware world components reused is a natural part of engineering process.

For example, today's interactive user interfaces are built with reusable components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms. The data structures and processing detail required to build the interface are contained within a library of reusable components for interface construction.

3.Describe “Software myth”? Discuss on various types of software myths and the true aspects of these myths?

Software Myths

Beliefs about software and the process used to build it- can be traced to the earliest days of computing myths have a number of attributes that have made them insidious.

1. **Management myths:** Manages with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping, and improve quality.

Myth: We already have a book that’s full of standards and procedures for building software - Wont that provide my people with everything they need to know?

Reality: The book of standards may very well exist but, is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice?

Myth: If we get behind schedule, we can add more programmers and catch up.

Reality: Software development is not a mechanistic process like manufacturing. As new people are added, people who were working must spend time educating the new comers, thereby reducing the amount of time spend on productive development effort. People can be added but only in a planned and well coordinated manner.

Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm built it.

Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

2. **Customer myths:** The customer believes myths about software because software managers and practitioners do little to correct misinformation. Myths lead to false expectations and ultimately, dissatisfaction with the developer.

Myth: A general statement of objectives is sufficient to begin with writing programs - we can fill in the details later.

Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous statement of objectives is recipe for disaster.

Myth: Project requirements continually change, but change can be easily accommodated because software is flexible.

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced and change can cause upheaval that requires additional resources and major design modification.

3. **Practitioner’s myths**: Myths that are still believed by software practitioners: during the early days of software, programming was viewed as an art from old ways and attitudes die hard.

Myth: Once we write the program and get it to work, our jobs are done.

Reality: Someone once said that the sooner you begin writing code, the longer it'll take you to get done. Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

Myth: The only deliverable work product for a successful project is the working program.

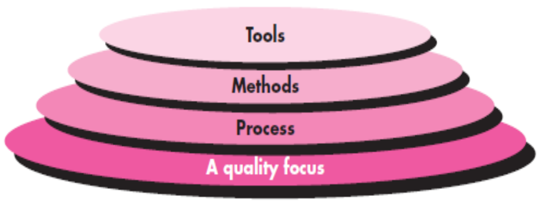
Reality: A working program is only one part of a software configuration that includes many elements. Documentation provides guidance for software support.

Myth: software engineering will make us create voluminous and unnecessary documentation and will invariably slows down.

Reality: software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

4.Explain software Engineering? Explain the software engineering layers?
Software Engineering- A Layered Technology

FIGURE 1.3
Software engineering layers



Software engineering is a layered technology. Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software engineering encompasses a process, methods for managing and engineering software, and tools.
A quality focus: Any engineering approach must rest on an organizational commitment to quality. The bedrock that supports software engineering is a quality focus.
Process: The foundation for software engineering is the process layer. Software engineering process is the glue that holds the technology layers. Process defines a framework that must be established for effective delivery of software engineering technology.
Methods: Software engineering methods provide the technical how-to's for building software. Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support. Software engineering methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.
Tools: Software engineering tools provide automated or semiautomated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established.

5.Explain in detail the capability Maturity Model Integration (CMMI)?

Capability Maturity Model Integration (CMMI) is a successor of CMM and is a more evolved model that incorporates best components of individual disciplines of CMM like Software CMM, Systems Engineering CMM, People CMM, etc. Since CMM is a reference model of matured practices in a specific discipline, so it becomes difficult to integrate these disciplines as per the requirements. This is why CMMI is used as it allows the integration of multiple disciplines as and when needed.

Objectives of CMMI :

1. Fulfilling customer needs and expectations.
2. Value creation for investors/stockholders.
3. Market growth is increased.
4. Improved quality of products and services.
5. Enhanced reputation in Industry.

CMMI Representation – Staged and Continuous : A representation allows an organization to pursue a different set of improvement objectives. There are two representations for CMMI :

- **Staged Representation:**
  - uses a pre-defined set of process areas to define improvement path.
  - provides a sequence of improvements, where each part in the sequence serves as a foundation for the next.
  - an improved path is defined by maturity level.
  - maturity level describes the maturity of processes in organization.
  - Staged CMMI representation allows comparison between different organizations for multiple maturity levels.
- **Continuous Representation :**
  - allows selection of specific process areas.
  - uses capability levels that measures improvement of an individual process area.
  - Continuous CMMI representation allows comparison between different organizations on a process-area-by-process-area basis.
  - allows organizations to select processes which require more improvement.
  - In this representation, order of improvement of various processes can be selected which allows the organizations to meet their objectives and eliminate risks.

CMMI Model – Maturity Levels : In CMMI with staged representation, there are five maturity levels described as follows :

Maturity level 1 : Initial

- processes are poorly managed or controlled.
- unpredictable outcomes of processes involved.
- ad hoc and chaotic approach used.
- No KPAs (Key Process Areas) defined.
- Lowest quality and highest risk.

Maturity level 2 : Managed

- requirements are managed.

- processes are planned and controlled.
- projects are managed and implemented according to their documented plans.
- This risk involved is lower than Initial level, but still exists.
- Quality is better than Initial level.

Maturity level 3 : Defined

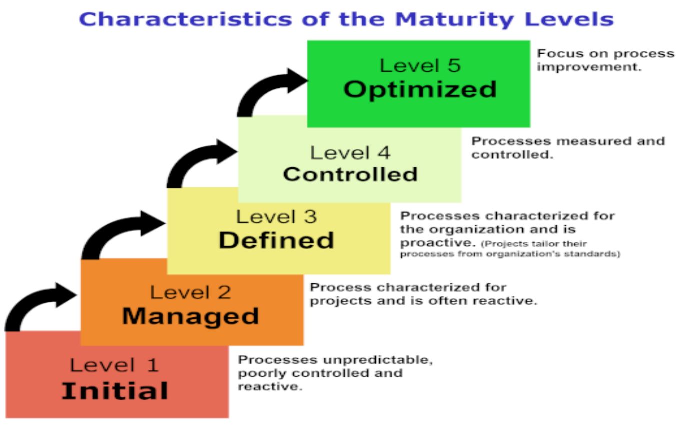
- processes are well characterized and described using standards, proper procedures, and methods, tools, etc.
- Medium quality and medium risk involved.
- Focus is process standardization.

Maturity level 4 : Quantitatively managed

- quantitative objectives for process performance and quality are set.
- quantitative objectives are based on customer requirements, organization needs, etc.
- process performance measures are analyzed quantitatively.
- higher quality of processes is achieved.
- lower risk

Maturity level 5 : Optimizing

- continuous improvement in processes and their performance.
- improvement has to be both incremental and innovative.
- highest quality of processes.
- lowest risk in processes and their performance.


Characteristics of the Maturity Levels
Level 5
Optimized
Focus on process improvement.
Level 4
Controlled
Processes measured and controlled.
Level 3
Defined
Processes characterized for the organization and is proactive. (Projects tailor their processes from organization's standards)
Level 2
Managed
Process characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive.

CMMI Model – Capability Levels A capability level includes relevant specific and generic practices for a specific process area that can improve the organization's processes associated with that process area. For CMMI models with continuous representation, there are six capability levels as described below :

Capability level 0 : Incomplete

- incomplete process – partially or not performed.
- one or more specific goals of process area are not met.
- No generic goals are specified for this level.
- this capability level is same as maturity level 1.

Capability level 1 : Performed

- process performance may not be stable.
- objectives of quality, cost and schedule may not be met.

    - a capability level 1 process is expected to perform all specific and generic practices for this level.
    - only a start-step for process improvement.

Capability level 2 : Managed

- process is planned, monitored and controlled.
- managing the process by ensuring that objectives are achieved.
- objectives are both model and other including cost, quality, schedule.
- actively managing processing with the help of metrics.

Capability level 3 : Defined

- a defined process is managed and meets th organization’s set of guidelines and standards.
- focus is process standardization.

Capability level 4 : Quantitatively Managed

- process is controlled using statistical and quantitative techniques.
- process performance and quality is understood in statistical terms and metrics.
- quantitative objectives for process quality and performance are established.

Capability level 5 : Optimizing

- focuses on continually improving process performance.
- performance is improved in both ways – incremental and innovation.
- emphasizes on studying the performance results across the organization to ensure that common causes or issues are identified and fixed.

6.Describe with the help of the diagram discuss in detail waterfall model. Give certain reasons for its failure?

Waterfall Model:

The classical waterfall model is the basic software development life cycle model. It is very simple but idealistic. Earlier this model was very popular but nowadays it is not used. However, it is very important because all the other software development life cycle models are based on the classical waterfall model.

What is the Waterfall Model?

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

Features of the Waterfall Model

- **Sequential Approach:** The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
- **Document-Driven**: The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.

- **Quality Control**: The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.
- **Rigorous Planning**: The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project lifecycle.

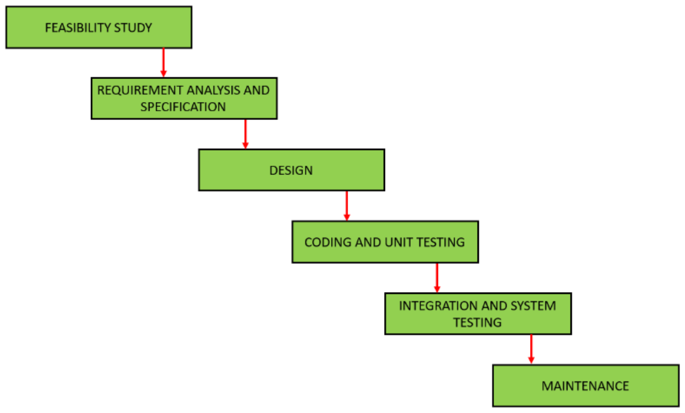
Importance of Waterfall Model

Clarity and Simplicity: The linear form of the Waterfall Model offers a simple and unambiguous foundation for project development.
Clearly Defined Phases: The Waterfall Model's phases each have unique inputs and outputs, guaranteeing a planned development with obvious checkpoints.
Documentation: A focus on thorough documentation helps with software comprehension, upkeep, and future growth.
Stability in Requirements: Suitable for projects when the requirements are clear and steady, reducing modifications as the project progresses.
Resource Optimization: It encourages effective task-focused work without continuously changing contexts by allocating resources according to project phases.
Relevance for Small Projects: Economical for modest projects with simple specifications and minimal complexity.

Phases of Waterfall Model

The Waterfall Model is a classical software development methodology that was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases that must be completed in a specific order.

The classical waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after the completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. Here the phases do not overlap with each other. The different sequential phases of the classical waterfall model are shown in the below figure.


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

Let us now learn about each of these phases in detail.

1. Feasibility Study

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

2. Requirements Analysis and Specification

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

Requirement gathering and analysis: Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (an inconsistent requirement is one in which some part of the requirement contradicts some other part).
Requirement specification: These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.

3. Design

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

4. **Coding and Unit Testing**

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

5. Integration and System testing

Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over several steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.
System testing consists of three different kinds of testing activities as described below.
Alpha testing: Alpha testing is the system testing performed by the development team.
Beta testing: Beta testing is the system testing performed by a friendly set of customers.
Acceptance testing: After the software has been delivered, the customer performs acceptance testing to determine whether to accept the delivered software or reject it.

6. Maintenance

Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are three types of maintenance.
Corrective Maintenance: This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
Perfective Maintenance: This type of maintenance is carried out to enhance the functionalities of the system based on the customer’s request.

Adaptive Maintenance: Adaptive maintenance is usually required for porting the software to work in a new environment such as working on a new computer platform or with a new operating system.

Advantages of the Classical Waterfall Model
The classical waterfall model is an idealistic model for software development. It is very simple, so it can be considered the basis for other software development life cycle models. Below are some of the major advantages of this SDLC model.

- **Easy to Understand:** The Classical Waterfall Model is very simple and easy to understand.
- **Individual Processing**: Phases in the Classical Waterfall model are processed one at a time.
- **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
- **Clear Milestones:** The classical Waterfall model has very clear and well-understood milestones.
- **Properly Documented:** Processes, actions, and results are very well documented.
- **Reinforces Good Habits:** The Classical Waterfall Model reinforces good habits like define-before-design and design-before-code.
- **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.

Disadvantages of the Classical Waterfall Model
The Classical Waterfall Model suffers from various shortcomings we can't use it in real projects, but we use other software development lifecycle models which are based on the classical waterfall model. Below are some major drawbacks of this model.

- **No Feedback Path:** In the classical waterfall model evolution of software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phase. Therefore, it does not incorporate any mechanism for error correction.
- **Difficult to accommodate Change Requests**: This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but the customer's requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
- **No Overlapping of Phases**: This model recommends that a new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase efficiency and reduce cost, phases may overlap.
- **Limited Flexibility**: The Waterfall Model is a rigid and linear approach to software development, which means that it is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed, it is difficult to make changes or go back to a previous phase.
- **Limited Stakeholder Involvement**: The Waterfall Model is a structured and sequential approach, which means that stakeholders are typically involved in the early phases of the project (requirements gathering and analysis) but may not be involved in the later phases (implementation, testing, and deployment).

- **Late Defect Detection:** In the Waterfall Model, testing is typically done toward the end of the development process. This means that defects may not be discovered until late in the development process, which can be expensive and time-consuming to fix.
- **Lengthy Development Cycle:** The Waterfall Model can result in a lengthy development cycle, as each phase must be completed before moving on to the next. This can result in delays and increased costs if requirements change or new issues arise.
- **Not Suitable for Complex Projects:** The Waterfall Model is not well-suited for complex projects, as the linear and sequential nature of the model can make it difficult to manage multiple dependencies and interrelated components.

When to Use the Waterfall Model?
Here are some cases where the use of the Waterfall Model is best suited:

- **Well-understood Requirements**: Before beginning development, there are precise, reliable, and thoroughly documented requirements available.
- **Very Little Changes Expected**: During development, very little adjustments or expansions to the project's scope are anticipated.
- **Small to Medium-Sized Projects**: Ideal for more manageable projects with a clear development path and little complexity.
- **Predictable:** Projects that are predictable, low-risk, and able to be addressed early in the development life cycle are those that have known, controllable risks.
- **Regulatory Compliance is Critical**: Circumstances in which paperwork is of utmost importance and stringent regulatory compliance is required.
- **Client Prefers a Linear and Sequential Approach**: This situation describes the client's preference for a linear and sequential approach to project development.
- **Limited Resources**: Projects with limited resources can benefit from a set-up strategy, which enables targeted resource allocation.

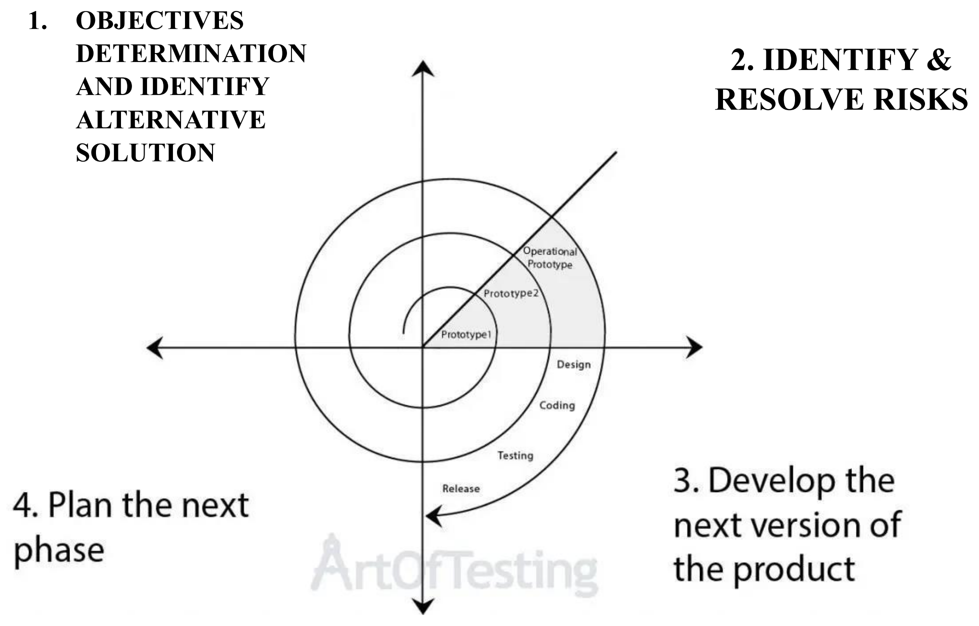
7.Explain the Spiral model in detail?
The Spiral Model
The Spiral Model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. This article focuses on discussing the Spiral Model in detail.

What is the Spiral Model?
The Spiral Model is a Software Development Life Cycle (SDLC) model that provides a systematic and iterative approach to software development. In its diagrammatic representation, looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process.

- The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.
- As the project manager dynamically determines the number of phases, the project manager has an important role in developing a product using the spiral model.
- It is based on the idea of a spiral, with each iteration of the spiral representing a complete software development cycle, from requirements gathering and analysis to design, implementation, testing, and maintenance.

What Are the Phases of Spiral Model?
The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:
1. Planning
The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.
2. Risk Analysis
In the risk analysis phase, the risks associated with the project are identified and evaluated.
3. Engineering
In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.
4. Evaluation
In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.
5. Planning
The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.

The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development. It is also well-suited to projects with significant uncertainty or high levels of risk.The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.


1. OBJECTIVES DETERMINATION AND IDENTIFY ALTERNATIVE SOLUTION
2. IDENTIFY & RESOLVE RISKS
Operational Prototype
Prototype2
Prototype1
Design
Coding
Testing
Release
3. Develop the next version of the product
4. Plan the next phase
ArtOfTesting

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

- **Objectives determination and identify alternative solutions**: Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- **Identify and resolve Risks**: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that

solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

- **Develop the next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
- **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started.

Risk Handling in Spiral Model

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype.

- The spiral model supports coping with risks by providing the scope to build a prototype at every phase of software development.
- The Prototyping Model also supports risk handling, but the risks must be identified completely before the start of the development work of the project.
- But in real life, project risk may occur after the development work starts, in that case, we cannot use the Prototyping Model.
- In each phase of the Spiral Model, the features of the product dated and analyzed, and the risks at that point in time are identified and are resolved through prototyping.
- Thus, this model is much more flexible compared to other SDLC models.

Why Spiral Model is called Meta Model?

- The Spiral model is called a Meta-Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model.
- The spiral model incorporates the stepwise approach of the Classical Waterfall Model.
- The spiral model uses the approach of the Prototyping Model by building a prototype at the start of each phase as a risk-handling technique.
- Also, the spiral model can be considered as supporting the Evolutionary model – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of the Spiral Model

Below are some advantages of the Spiral Model.

- **Risk Handling**: The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- **Flexibility in Requirements**: Change requests in the Requirements at a later phase can be incorporated accurately by using this model.
- **Customer Satisfaction**: Customers can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

- **Iterative and Incremental Approach**: The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.
- **Emphasis on Risk Management**: The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.
- **Improved Communication**: The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.
- **Improved Quality**: The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability.

Disadvantages of the Spiral Model
Below are some main disadvantages of the spiral model.

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

When To Use the Spiral Model?

- When a project is vast in software engineering, a spiral model is utilized.
- A spiral approach is utilized when frequent releases are necessary.
- When it is appropriate to create a prototype
- When evaluating risks and costs is crucial
- The spiral approach is beneficial for projects with moderate to high risk.
- The SDLC's spiral model is helpful when requirements are complicated and ambiguous.
- If modifications are possible at any moment
- When committing to a long-term project is impractical owing to shifting economic priorities.

8.Explain changing nature of software in detail? Or what are the different software applications?

The 7 broad categories of computer software present continuing challenges for software engineers:

- System software
- Application software
- Engineering/scientific software

- Embedded software
- Product-line software
- Web-applications
- Artificial intelligence software.

System software: System software is a collection of programs written to service other programs. The systems software is characterized by heavy interaction with computer hardware heavy usage by multiple users concurrent operation that requires scheduling, resource sharing, and sophisticated process management complex data structures multiple external interfaces
E.g. compilers, editors and file management utilities.

Application software:
Application software consists of standalone programs that solve a specific business need. It facilitates business operations or management/technical decision making.
It is used to control business functions in real-time
E.g. point-of-sale transaction processing, real-time manufacturing process control.

Engineering/Scientific software: Engineering and scientific applications range
-from astronomy to volcanology
- from automotive stress analysis to space shuttle orbital dynamics
- from molecular biology to automated manufacturing
E.g. computer aided design, system simulation and other interactive applications.

Embedded software:
Embedded software resides within a product or system and is used to implement and control features and functions for the end-user and for the system itself. It can perform limited and esoteric functions or provide significant function and control capability.
E.g. Digital functions in automobile, dashboard displays, braking systems etc.

Product-line software: Designed to provide a specific capability for use by many different customers, product-line software can focus on a limited and esoteric market place or address mass consumer markets
E.g. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications
Web-applications: WebApps are evolving into sophisticated computing environments that not only provide standalone features, computing functions, and content to the end user, but also are integrated with corporate databases and business applications.

Artificial intelligence software: AI software makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Application within this area includes robotics, expert systems, pattern recognition, artificial neural networks, theorem proving, and game playing.

9. Explain the new challenges in Software Development?
The following are the new challenges on the horizon:

- **Ubiquitous computing**
- **Netsourcing**

- **Open source**
- **The “new economy”**

Ubiquitous computing: The challenge for software engineers will be to develop systems and application software that will allow small devices, personal computers and enterprise system to communicate across vast networks.

Net sourcing: The challenge for software engineers is to architect simple and sophisticated applications that provide benefit to targeted end-user market worldwide.

Open Source: The challenge for software engineers is to build source that is self descriptive but more importantly to develop techniques that will enable both customers and developers to know what changes have been made and how those changes manifest themselves within the software.

The “new economy”: The challenge for software engineers is to build applications that will facilitate masscommunication and mass product distribution.

9. Explain Process Framework Activities?

Process Framework:

- Software process must be established for effective delivery of software engineering technology.
- A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- The process framework encompasses a set of umbrella activities that are applicable across the entire software process.
- Each framework activity is populated by a set of software engineering actions.
- Each software engineering action is represented by a number of different task sets- each a collection of software engineering work tasks, related work products, quality assurance points, and project milestones.

In brief

"A process defines who is doing what, when, and how to reach a certain goal."

A Process Framework

- establishes the foundation for a complete software process
- identifies a small number of framework activities
- applies to all s/w projects, regardless of size/complexity.
- also, a set of umbrella activities
- applicable across the entire s/w process.

Each framework activity has

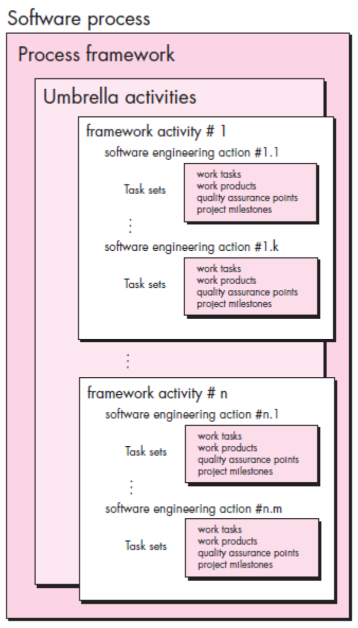
- set of s/w engineering actions.

Each s/w engineering action (e.g., design) has a collection of related tasks (called task sets):

- work tasks.
- work products (deliverables)

- quality assurance points
- project milestones.

FIGURE 2.1
A software process framework



A generic process framework for software engineering defines five framework activities—

- **communication:** This framework activity involves heavy communication and collaboration with the customer and encompasses requirements gathering and other related activities.
- **planning:** This activity establishes a plan for the software engineering work that follows. It describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.
- **Modelling:** This activity encompasses the creation of models that allow the developer and customer to better understand software requirements and the design that will achieve those requirements. The modeling activity is composed of 2 software engineering actions- analysis and design.
- ✓ Analysis encompasses a set of work tasks.
- ✓ Design encompasses work tasks that create a design model.
- **Construction:** This activity combines core generation and the testing that is required to uncover the errors in the code.
- **Deployment:** The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evolution.

In addition, a set of umbrella activities—
1**) Software project tracking and control** – allows the software team to assess progress against the project plan and take necessary action to maintain schedule.
2) Risk Management - assesses risks that may effect the outcome of the project or the quality of the product.

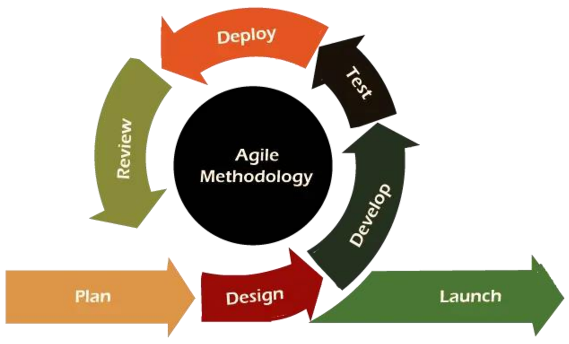
3) Software Quality Assurance - defines and conducts the activities required to ensure software quality.
4) Formal Technical Reviews - assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next action or activity.
5) Measurement - define and collects process, project and product measures that assist the team in delivering software that needs customer’s needs, can be used in conjunction with all other framework and umbrella activities.
6) Software configuration management - manages the effects of change throughout the software process.
7) Reusability management - defines criteria for work product reuse and establishes mechanisms to achieve reusable components.
8) Work Product preparation and production - encompasses the activities required to create work products such as models, document, logs, forms and lists

10. Explain about Agile model?

The digital world's increasing need for complex software applications has led to a high failure rate in software development projects. The traditional Iterative Waterfall approach faced challenges in handling customer change requests, which were time-consuming and expensive. In 2001, the Agile model in Software engineering emerged as a solution. Agile was initially designed for software development but has now expanded to manage projects across different industries and markets.

Agile refers to something **quick or adaptable**. The Agile model is an SDLC model focused on **iterative software product development**. The Agile paradigm is an incremental model in which software is built in quick increments. The most significant aspect of the Agile model is determining the project scope, requirements, number, and duration of iterations at the start of the development process.

Phases Of The Agile Model



Following are the Agile Model phases:

- **Requirement Gathering (Plan)**
  In this stage, the project's critical needs are specified. This stage discusses the key features and plans the project's time and effort. At this point, the team must establish the criteria. They should define the business potential for the project and quantify the time and effort needed to complete it. Based on these data, you can examine technical and economic feasibility.

- **Design the Requirement**
  Once the project requirements have been gathered, the development team must collaborate with stakeholders to design requirements. A user flow diagram or a high-level UML diagram can be used to demonstrate the functionality of new features and how they will interact with the existing system.

- **Develop/Iteration**
  The real effort begins after the team specifies and designs the requirements. After the requirements have been gathered and defined clearly, the software developers begin working on projects with the goal of creating a viable product. All of this occurs throughout an iteration or sprint. Before being released, the product will go through multiple rounds of development.

- **Testing/Quality Assurance**
  In this phase, the QA team tests the product's performance and looks for bugs in the code.

- **Deployment**
  In this phase, the team creates a product for the user's work environment.

- **Feedback**
  This phase involves taking feedback from the users after the product has been released. The team may need to implement changes if necessary.

Iteration of the Agile Model in Software Engineering

An Agile iteration is a short period of time in which a piece of work is **developed and tested**. Each iteration has a deadline by which all deliverables must be completed. Iterations are the fundamental building element of Agile development. They usually last from **one to four weeks**. Each iteration yields minor incremental releases that build on previous capabilities. This individual release is rigorously tested to maintain the software's quality.

The entire project is divided into smaller portions or sprints with Agile to reduce project delivery time and hazards. An iteration requires a team to go through the entire software development cycle. Within a single iteration, an Agile team will

1. map out the requirements.
2. develop the user stories.
3. test their software.
4. produce an end deliverable.
5. request for user feedback.

Applications of the Agile Model

- The Agile model was primarily conceived to help a project adapt quickly to **changing requests**. So, the main aim of the Agile model is to enable quick project completion.
- To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.

- Also, anything that is a waste of time and effort is avoided. Agile finds applications in various **medium to large software projects**.

When to Use the Agile Model?

- Agile is helpful when there is a need for implementing new changes as the flexibility provided by the Agile model becomes crucial.
- It is helpful when frequent addition of new features is needed at minimal cost in agile.
- Developers can easily roll back and incorporate new features, typically requiring only a few days or even hours of work.
- Agile is suitable when project requirements are expected to change or evolve over time.
- Agile is well-suited for projects that require rapid and incremental delivery of usable software.
- Agile is effective for complex and innovative projects that require an iterative and adaptive approach.
- Agile works best with cross-functional teams, fostering collaboration and shared ownership.

Advantages of the Agile Model

- Here are some advantages of the agile methodology:
- Continuous delivery
- Encourages **one-on-one** communication with clients
- Changes in requirements can be accommodated at any moment.
- Reduced **development time**.
- Efficient and fulfilling business standards.

Disadvantages of the Agile Model

- Here are some disadvantages of the agile methodology:
- It depends heavily on **customer interaction**, so the team can be driven in the wrong direction if the customer is unclear about their requirements.
- Due to the absence of **proper documentation**, maintenance of the developed project can become a problem when the project is completed and the developers are assigned to another project. As a result, there is a very high individual dependency. Transfer of technology to new team members may be pretty tricky. Hence, maintenance can be challenging.

Because of the lack of official records, there can be misunderstandings, and crucial decisions made during various phases can be misunderstood at any point by different team members.

11.What are the Principles of Agile?

Agility Principles

The Agile Alliance defines 12 agility principles for those who want to achieve agility:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self– organizing teams.

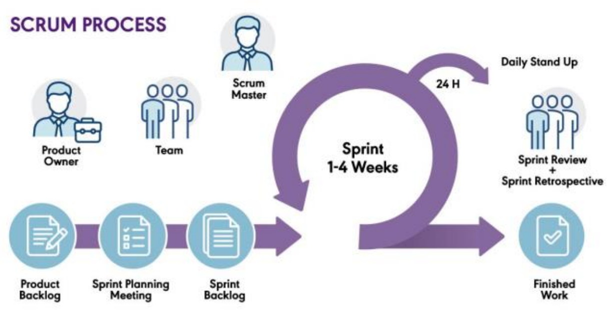
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

12. Explain about Scrum in Agile Methodology?

Scrum: It is the most popular variant of the agile methodology. It is a team-based agile development process that focuses on task management. It encourages collaboration in small groups and believes in empowering the development team. Each iteration of a scrum is termed a Sprint. A scrum team has three key positions, each with its own set of responsibilities, which are outlined below:

Product Owner	**Scrum Master**	**The Team**
He/She defines features of the product.	He/She manages the team and look after the team's productivity	The team is usually about 5-9 members
Product Owner decides the release date and corresponding features	He/She maintains the block list and removes barriers in the development	It includes developers, designer and sometimes testers, etc.
They prioritize the features according to the market value and profitability of the product	He/She coordinates with all roles and functions	The team organizes and schedule their work on their own

Product Owner	**Scrum Master**	**The Team**
He/She is responsible for the profitability of the product	He/She shields team from external interferences	Has right to do everything within the boundaries of the project to meet the sprint goal
He/She can accept or reject work item result	Invites to the daily scrum, sprint review and planning meetings	Actively participate in daily ceremonies



Product Backlog: is a prioritized features list, containing short descriptions of all functionalities desired in the product. When applying Scrum, it's not necessary to start a project with a lengthy, upfront effort to document all requirements.
Sprint Backlog: A sprint backlog is a list of work items your team plans to complete during a project sprint. These items are usually pulled from the product backlog during the sprint planning session.
Sprint: A Sprint is a time box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.
Sprint Review: If the product still has some non-achievable features, it will be checked in this stage and then passed to the Sprint Retrospective stage.
Sprint Retrospective: In this stage quality or status of the product is checked.A sprint retrospective is a team meeting that takes place at the end of a sprint, aimed at introspecting the sprint on what worked and what didn't, and how to improve in the upcoming sprints.

How Scrum Works
In a rugby scrum, all the players literally put their heads together. When it comes to software development, a scrum can be characterized by developers putting their heads together to address complex problems.

Scrum software development starts with a wish list of features — a.k.a. a product backlog. The team meets to discuss:

- The backlog.
- What still needs to be completed.
- How long it will take.

Scrum relies on an agile software development concept called sprints:

- Sprints are periods of time when software development is actually done.
- A sprint usually lasts from one week to one month to complete an item from the backlog.
- The goal of each sprint is to create a saleable product.
- Each sprint ends with a sprint review.
- Then the team chooses another piece of backlog to develop — which starts a new sprint.
- Sprints continue until the project deadline, or the project budget is spent.

In daily scrums, teams meet to discuss their progress since the previous meeting and make plans for that day.

- The meetings should be brief — no longer than 15 minutes.
- Each team member needs to be present and prepared.
- The ScrumMaster keeps the team focused on the goal.

Advantage of using Scrum framework:

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It’s like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of using Scrum framework:

- Scrum framework do not allow changes into their sprint.
- Scrum framework is not fully described model. If you wanna adopt it you need to fill in the framework with your own details like Extreme Programming(XP), Kanban, DSDM.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

13.Explain about Crystal in Agile Methodology?

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

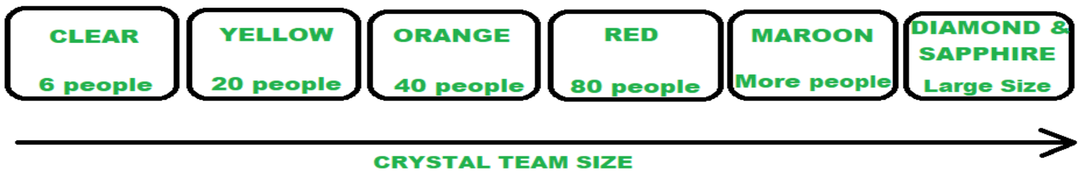
Let’s know about the history of the Crystal Method: The crystal method was developed by an American scientist named Alistair Cockburn who worked at IBM. He decided not to focus on step-by-step developmental strategies, but to develop team collaboration and communication. Some of the traits of Cockburn’s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

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

How does Crystal function?

Till now, we got to know that crystal is a family of various developmental approaches, and it is not a group of prescribed developmental tools and methods. In the beginning, the approach is set by considering the business requirements and the needs of the project. Various methodologies in the Crystal family also known as weights of the Crystal approach are represented by different colors of the spectrum. Crystal family consists of many variants like Crystal Clear, Crystal Yellow, Crystal Red, Crystal Sapphire, Crystal Red, Crystal Orange Web, and Crystal Diamond.

1. **Crystal Clear-** The team consists of only 1-6 members that is suitable for short-term projects where members work out in a single workspace.
2. **Crystal Yellow-** It has a small team size of 7-20 members, where feedback is taken from Real Users. This variant involves automated testing which resolves bugs faster and reduces the use of too much documentation.
3. **Crystal Orange-** It has a team size of 21-40 members, where the team is split according to their functional skills. Here the project generally lasts for 1-2 years and the release is required every 3 to 4 months.
4. **Crystal Orange Web-** It has also a team size of 21-40 members were the projects that have a continually evolving code base that is being used by the public. It is also similar to Crystal Orange but here they do not deal with a single project but a series of initiatives that required programming.
5. **Crystal Red-** The software development is led by 40-80 members where the teams can be formed and divided according to requirements.
6. **Crystal Maroon-** It involves large-sized projects where the team size is 80-200 members and where methods are different and as per the requirement of the software.
7. **Crystal Diamond & Sapphire-** This variant is used in large projects where there is a potential risk to human life.



Properties of Crystal Agile Framework:

1. **Frequent Delivery-** It allows you regularly deliver the products and test code to real users. Without this, you might build a product that nobody needs.
2. **Reflective Improvement-** No matter how good you have done or how bad you have done. Since there are always areas where the product can be improved, so the teams can implement to improve their future practices.
3. **Osmotic Communication-** Alistair stated that having the teams in the same physical phase is very much important as it allows information to flow in between members of a team as in osmosis.
4. **Personal Safety-** There are no bad suggestions in a crystal team, team members should feel safe to discuss ideas openly without any fear.
5. **Focus-** Each member of the team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and works towards the same goal.
6. **Easy access to expert users-** It enhances team communication with users and gets regular feedback from real users.
7. **Technical tooling-** It contains very specific technical tools which to be used by the software development team during testing, management, and configuration. These tools make it enable the team to identify any error within less time.
8. **Continuous learning** – The framework emphasizes on continuous learning, enabling team members to acquire new skills and knowledge, and apply them in their work.
9. **Teamwork** – The framework stresses on the importance of teamwork, promoting collaboration, and mutual support among team members.

10. **Timeboxing –** The framework adopts timeboxing to manage project deadlines, ensuring that the team delivers within set timelines.
11. **Incremental development –** The framework promotes incremental development, enabling the team to deliver working software frequently, and adapt to changes as they arise.
12. **Automated testing –** The framework emphasizes on automated testing, enabling the team to detect and fix bugs early, reducing the cost of fixing errors at later stages.
13. **Customer involvement –** The framework emphasizes on involving customers in the development process, promoting customer satisfaction, and delivering products that meet their needs.
14. **Leadership –** The framework encourages leadership, enabling team members to take ownership of their work and make decisions that contribute to the success of the project.

Benefits of using the Crystal Agile Framework :

- Facilitate and enhance team communication and accountability.
- The adaptive approach lets the team respond well to the demanding requirements.
- Allows team to work with the one they see as the most effective.
- Teams talk directly with each other, which reduces management overhead.
- **Faster delivery –** The framework enables the team to deliver working software faster, which can help gain a competitive advantage in the market.
- **Higher quality –** The framework emphasizes on quality, enabling the team to detect and fix defects early in the development process, resulting in a higher quality product.
- **Improved customer satisfaction –** The framework promotes customer involvement, enabling the team to deliver products that meet customer needs, resulting in higher customer satisfaction.
- **Increased productivity –** The framework enables the team to focus on delivering the highest value features, which can increase productivity and reduce waste.
- **Flexibility –** The framework is highly adaptable, enabling the team to adjust to changing requirements, and make decisions based on real-time feedback.
- **Empowerment –** The framework promotes empowerment, enabling team members to take ownership of their work, and make decisions that contribute to the success of the project.
- **Reduced risk –** The framework promotes risk management, enabling the team to identify and mitigate potential risks early in the development process, reducing the likelihood of project failure.

Drawbacks of using the Crystal Agile Framework :

- A lack of pre-defined plans may lead to confusion and loss of focus.
- Lack of structure may slow down inexperienced teams.
- Not clear on how a remote team can share knowledge informally.
- **Lack of predictability –** The framework's emphasis on adaptability and flexibility may result in a lack of predictability, making it difficult to plan and estimate project timelines and budgets.

- **Lack of documentation –** The framework’s emphasis on communication and collaboration may result in a lack of documentation, making it difficult to track progress and maintain a record of decisions.
- **Limited scalability –** The framework may not be suitable for large or complex projects, as the lack of structure and predefined plans may make it difficult to manage teams at scale.
- **Dependence on team expertise –** The framework relies heavily on the expertise and skills of the development team, which may not be suitable for teams with limited experience or knowledge.
- **Lack of clarity on roles and responsibilities –** The framework’s emphasis on self-organizing teams may result in a lack of clarity on roles and responsibilities, leading to confusion and a loss of focus.
- **Inability to handle regulatory requirements –** The framework may not be suitable for projects with strict regulatory requirements, as the lack of documentation and structure may not meet compliance standards.
- **Potential for informal knowledge sharing –** The framework’s emphasis on osmotic communication may result in informal knowledge sharing, which may be difficult to track and monitor for accuracy and completeness.

15. Explain about Dynamic Software Development Method?

DSDM is an Agile model that provides an iterative and incremental framework for software development. It emphasizes active user involvement throughout the project and places a strong emphasis on delivering business value. DSDM promotes collaboration, prototyping, and continuous feedback from stakeholders. It includes phases like feasibility study, business study, functional model iteration, design and build iteration, and implementation.

Like the wider agile family of methodologies, dynamic systems development method is an iterative approach to software development but adds additional discipline and structure to the process. Central to DSDM is the principle that “any project must be aligned to clearly defined strategic goals and focus upon early delivery of real benefits to the business.”

History of dynamic systems development method

In the 1990s, the rapid application development (RAD) approach was becoming increasingly popular, which enabled developers to show their users and customers possible solutions quickly with easy to build prototypes.

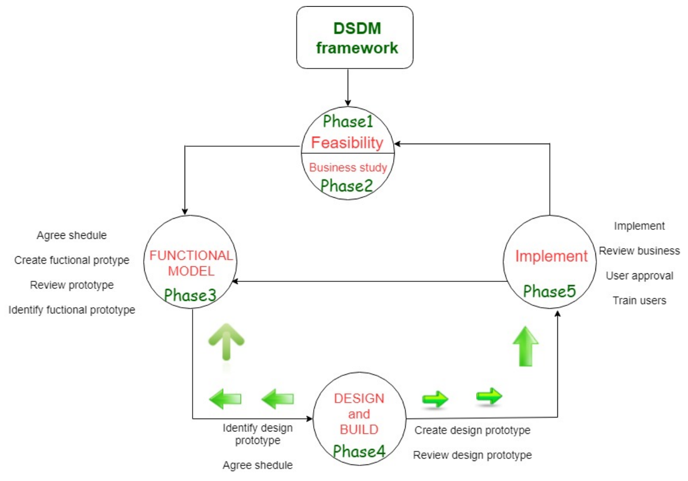
However, this approach was often unstructured, with no common processes between RAD teams. This led to each organization building their approaches and frameworks, splitting standards and making it difficult to recruit experienced RAD practitioners.

To solve this issue, the DSDM consortium was founded with the objective of "jointly developing and promoting an independent RAD framework" and DSDM was born.

DSDM is structured around eight key principles:

1. **Focus on the business need:** DSDM teams must establish a valid business case and ensure organizational support throughout the project

2. **Deliver on time:** Work should be time-boxed and predictable, to build confidence in the development team.
3. **Collaborate:** DSDM teams must involve stakeholders throughout the project and empower all members of the team to make decisions.
4. **Quality:** To ensure high quality, the level of quality should be agreed with the business at the start of the project. This is enforced through continuous testing, review, and documentation.
5. **Build incrementally from firm foundations:** Teams must do enough design work up front (EDUF) to ensure they know exactly what to build, but not too much to slow development.
6. **Developer Iteratively:** Take feedback from the business and use this to continually improve with each development iteration. Teams must also recognize that details emerge as the project or product develops and they must respond to this.
7. **Communicate continuously and clearly:** Holding daily stand-up sessions, encouraging informal communication, running workshops and building prototypes are all key DSDM tools. Communicating through documents is discouraged - instead, documentation must be lean and timely.
8. **Demonstrate control:** The project manager and team leader should make their plans and progress visible to all and focus on successful delivery.



Dynamic Systems Development Method life cycle

Phases of DSDM

1. **Feasibility Study:** It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.
2. **Business Study:** It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.
3. **Functional Model Iteration:** It produces a collection of progressive prototypes that demonstrate practicality for the client. (Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm.
4. **Design and Build Iteration:** It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner

that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time.

5. **Implementation:** It places the newest code increment (an "operationalized" prototype) into the operational surroundings. It ought to be noted that:
    - **(a)** the increment might not 100% complete or,
    - **(b)** changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.

Advantages of dynamic systems development method

- Projects are delivered on time, whilst still allowing flexibility
- Progress can be easily understood across the organization
- Business cases are at the core of the DSDM model, ensuring delivered projects have real business value

Disadvantages of the dynamic systems development method

- Large management overhead and costly implementation makes this unsuitable for small organizations
- DSDM can be restrictive and inhibit developer creativity. Projects are likely to be completed exactly as specified, even if more elegant solutions are available.

16. Explain about Feature Driven Development (FDD)

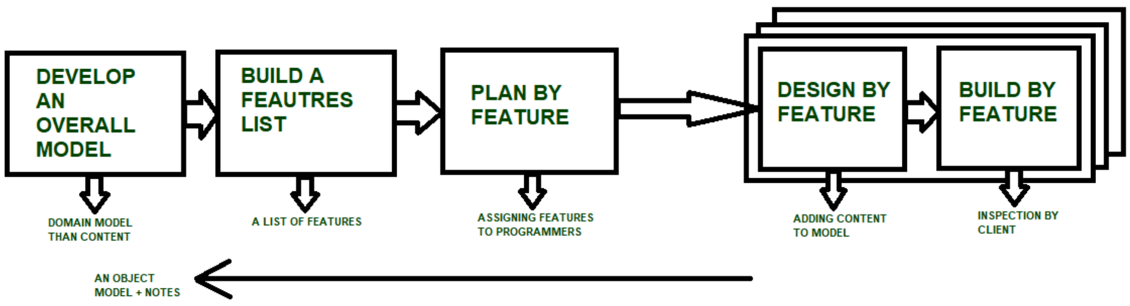
It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive of feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

History

FDD was first applied in the year 1997 on a real-world application by Jeff De Luca for large software development with specific needs of 15-month and 50 persons and published as a discussion in book Java Modeling in Color with UML in the year 1999.

FDD Lifecycle

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



Characteristics of FDD

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

Advantages of FDD

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

Disadvantages of FDD

- This agile practice is not good for smaller projects.
- There is high dependency on lead programmers, designers and mentors.
- There is lack of documentation which can create an issue afterwards.

17.Explain about Lean Software Development

Lean Software Development is an Agile model inspired by lean manufacturing principles. It aims to eliminate waste, optimize flow, and maximize value delivery. Lean focuses on continuous improvement, customer collaboration, and quick feedback cycles. It encourages practices such as value stream mapping, just-in-time development, and continuous delivery.

Lean Software Development (LSD) is an agile framework that is used to streamline & optimize the software development process. It may also be referred to as Minimum Viable Product (MVP) strategy as these ways of thinking are very much alike since both intend to speed up development by focusing on new deliverables.

Toyota has been credited to inspire the lean development approach which is meant for optimizing production and minimize waste. Seeing Toyota's lean approach many other manufacturing teams started to follow the same strategy. And it was first adopted in software development in 2003.

Key Principles of Lean Software Development :

Eliminating the Waste: To identify and eliminate wastes e.g. unnecessary code, delay in processes, inefficient communication, the issue with quality, data duplication, more tasks in the log than completed, etc. regular meetings are held by Project Managers. Which allows team members to point out faults and suggest changes in the next turn.

Fast Delivery: Previously long time planning used to be the key success in business, but in the passage of time it is found that engineers spend too much time on building complex systems with unwanted features. So they came up with an MVP strategy which resulted in the building products quickly that included a little functionality and launch the product to

market and see the reaction. Such an approach allows them to enhance the product on the basis of customer feedback.
Amplify Learning: Learning is improved through ample code reviewing, meeting that is cross-team applicable. It is also ensured that particular knowledge isn't accumulated by one engineer who's writing a particular piece of code so paired programming is used.
Builds Quality: LSD is all about prevent waste, keeping an eye on not sacrificing quality. Developers often apply test-driven programming to examine the code before it is written. The quality can also be gained to get constant feedback from team members and project managers.
Respect Teamwork: LSD focuses on empowering team members, rather than controlling them. Setting up a collaborative atmosphere, keep perfect balance when there are short deadlines and immense workload. This method becomes very much important when new members join a well-established team.
Delay the Commitment: In traditional project management it often happens when you make your application and it turns out to be completely unfit for the market. LSD method recognizes this threat and makes room for improvement by postponing irreversible decisions until all experiment is done. This methodology always constructs software as flexible, so the new knowledge is available and engineers can make improvements.
Optimizing the whole system: lean's principle allows managers to break an issue into small constituent parts to optimize the team's workflow, create unity among members, and inspire a sense of shared responsibility which results in enhancing the team performance.

Key feature of LSD:

- Focus on continuous improvement and waste elimination
- Iterative and collaborative approach
- Development of Minimum Viable Product (MVP)
- Customer involvement and feedback
- Flexible and adaptive approach
- Prioritization of essential functions and minimization of waste

Advantages of LSD :
LSD has proved to improve software development in the following ways :

- LSD removes the unnecessary process stages when designing software so that it acts as a time saver as simplifies the development process.
- With a focus on MVP, Lean Software Development prioritizes essential functions so this removes the risk of spending time on valueless builds.
- It increases the involvement power of your team as more and more members participate due to which the overall workflow becomes optimized and losses get reduced.
- LSD simplifies the development process and saves time by removing unnecessary stages.
- It prioritizes essential functions and removes the risk of spending time on valueless builds.
- It increases the involvement of team members, optimizing the workflow and reducing losses.
- LSD encourages collaboration and communication among team members and stakeholders.
- It fosters innovation through experimentation and creativity.
- It reduces waste by minimizing unnecessary features.

- LSD increases customer satisfaction through customer involvement and feedback.
- It improves flexibility by allowing for adaptability to changing requirements and circumstances.

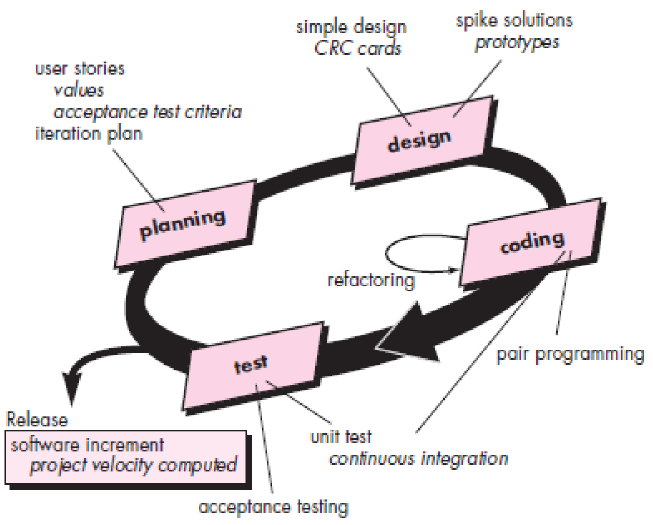
Weakness in LSD :

- Make it scalable as other frameworks since it strongly depends on the team involved.
- It is hard to keep pace so it is not easy for developers to work with team members as conflict may occur between them.
- It leads to a difficult decision-making process as it is mandatory for customers to clearly set their requirements for the development not to be interrupted.
- Lack of documentation
- Inflexibility for structured projects or strict deadlines
- Lack of predictability
- Dependence on customer involvement

18.Explain about Extreme Programming (XP)

Extreme Programming is an Agile model that emphasizes customer satisfaction, teamwork, and high-quality software. XP promotes short development cycles, frequent customer feedback, and continuous testing. It encourages practices like pair programming, test-driven development, continuous integration, and collective code ownership.

FIGURE 3.2
The Extreme Programming process



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

Planning, the first stage, is when the customer meets the development team and presents the requirements in the form of user stories to describe the desired result. The team then estimates the stories and creates a release plan broken down into iterations needed to cover the required functionality part after part. If one or more of the stories can't be estimated, so-called spikes can be introduced which means that further research is needed.

Designing is a part of the planning process but can be set apart to emphasize its importance. A good design brings logic and structure to the system and allows to avoid unnecessary complexities and redundancies.

Coding is the phase during which the actual code is created by implementing specific XP practices such as coding standards, pair programming, continuous integration, and collective code ownership.
Testing is the core of extreme programming. It is the regular activity that involves both unit tests (automated testing to determine if the developed feature works properly) and acceptance tests (customer testing to verify that the overall system is created according to the initial requirements).
Listening is all about constant communication and feedback. The customers and project managers are involved in describing the business logic and value that is expected.

Extreme Programming (XP) in software engineering has many advantages, including:

- Clear code: XP developers create simple code that can be improved at any time
- Error avoidance: Pair programming helps avoid errors
- No overtime: XP doesn't involve overtime
- Team work: Teams can work at their own pace
- Changes at short notice: Changes can be made quickly
- Avoids employee burnout: XP prevents unnecessary work
- Continuous feedback: Frequent demonstrations of working software provide feedback from customers and stakeholders.

However, XP also has some disadvantages, including:

- Time waste: XP can lead to time waste
- Not enough documentation: XP projects may not have good defect documentation
- Requires self-discipline: XP requires self-discipline to practice
- Relatively high costs: Pair programming means double salary, which can be problematic for smaller teams
- Focus on code: XP may be too focused on code rather than design
- Learning curve: Adopting XP requires a significant learning curve for team members who are not familiar with its practices and principles
- Not suitable for all projects: XP may not be suitable for projects with stable and well-defined requirements, or for very large projects

19.Explain about Adaptive Software Development (ASD)

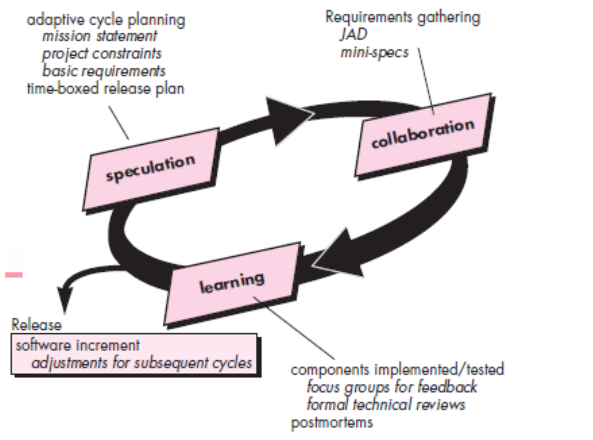
Adaptive Software Development practices provide ability to accommodate change and are adaptable in turbulent environments with products evolving with little planning and learning.

Phases of ASD Life Cycle

Adaptive Software Development is cyclical like the Evolutionary model, with the phase names reflecting the unpredictability in the complex systems. The phases in the Adaptive development life cycle are −

- **Speculate:** During this phase project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs, customer mission statement, etc, to define set of release cycles that the project wants.
- **Collaborate:** It is the difficult part of ASD as it needs the workers to be motivated. It collaborates communication and teamwork but emphasizes individualism as individual creativity plays a major role in creative thinking. People working together must trust each others.
- **Learn:** The workers may have a overestimate of their own understanding of the technology which may not lead to the desired result. Learning helps the workers to increase their level of understanding over the project. Learning process is of three ways: Focus groups, Technical reviews, Project postmortem

FIGURE 3.3
Adaptive software development



Advantages of Adaptive Software Development :

- Useful for rapid and complex software product development.
- Easy software incremental adjustment
- Focus on end-users, meeting requirements, and fulfilling demands
- Allows on-time delivery with maximum customer satisfaction
- Provides high transparency between developers and clients
- Reduced vulnerabilities and bugs as undergoes multiple testing

Disadvantages of Adaptive Software Development :

- Working in an uncertain environment is challenging
- Proceeding only with mission needs broad exploration and constant focus
- Requires high user/client involvement
- Requires testing into each iteration which increases the cost
- Frequent changes undergo with less documentation
- Requires strict time commitment between different teams involved in project

20. Explain Agile Unified Process (AUP)

Agile Unified Process (AUP) is a lightweight, iterative, and adaptable **software development methodology** that combines the best practices of agile development with the disciplined approach of the **Unified Process** (UP). AUP is designed to deliver high-quality software that meets the changing needs of the stakeholders in an efficient and effective manner. In this article, we will explore the key principles, phases, and practices of AUP, and highlight its benefits and limitations.

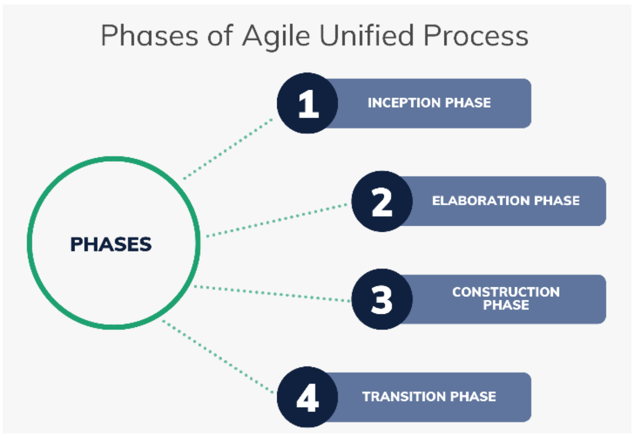
Key Principles of Agile Unified Process

The following are the key principles of AUP:

1. **People and Communication:** AUP emphasizes the importance of people and communication in *software development*. It promotes face-to-face communication between stakeholders, customers, and development teams to ensure that everyone is on the same page.
2. **Continuous Improvement:** AUP is based on the principle of continuous improvement. It encourages teams to learn from their experiences, and to continuously improve their processes and practices.
3. **Collaboration:** AUP emphasizes the importance of collaboration between stakeholders, customers, and development teams. It promotes the use of collaborative tools and techniques to enhance communication and collaboration.
4. **Iterative and Incremental:** AUP is an iterative and incremental process that delivers working software in small, incremental releases. It allows teams to gather feedback and incorporate changes at each iteration, which leads to better quality software.
5. **Risk-Driven:** AUP is a risk-driven process that focuses on identifying, prioritizing, and mitigating risks throughout the development lifecycle. It encourages teams to manage risks proactively to avoid potential problems and delays.

Phases of Agile Unified Process

AUP consists of four phases: **Inception**, **Elaboration**, **Construction**, and **Transition**. Each phase has a specific set of goals, activities, and deliverables that must be completed before moving to the next phase.



1. **Inception Phase:** The Inception phase is the initial phase of the project, where the scope, objectives, and requirements of the project are defined. The key deliverables of this phase include the Project Vision, Business Case, Use-Case Model, and Risk List.
2. **Elaboration Phase:** The Elaboration phase is the most critical phase of the project, where the requirements are analyzed, designed, and validated. The key deliverables of this phase include the System Architecture, Software Requirements Specification, Software Architecture Document, and Test Plan.
3. **Construction Phase:** The Construction phase is the phase where the actual coding, testing, and integration of the software take place. The key deliverables of this phase include the Source Code, Unit Test Cases, and User Documentation.

4. **Transition Phase:** The Transition phase is the final phase of the project, where the software is deployed and delivered to the stakeholders. The key deliverables of this phase include the User Acceptance Test Report, Deployment Plan, and User Manual.

Practices of Agile Unified Process

AUP employs a set of practices that are essential for the success of the project. The following are the key practices of AUP:

1. **Agile Modeling:** Agile Modeling is a set of practices that enable teams to create and maintain effective models of the system. It promotes the use of simple, visual models that are easy to understand and communicate.
2. **Test-Driven Development:** Test-Driven Development (TDD) is a practice that involves writing automated test cases before writing the code. It ensures that the code meets the requirements and is of high quality.
3. **Continuous Integration:** Continuous Integration (CI) is a practice that involves integrating the code changes frequently and automatically. It ensures that the software is always in a working state and reduces the risk of integration issues.
4. **Refactoring:** Refactoring is a practice that involves improving the design and code quality of the software without changing its behavior. It helps teams to maintain and improve the software in an efficient and effective manner.
5. **Iterative Development:** Iterative Development is a practice that involves delivering working software in small, incremental releases. It allows teams to gather feedback and incorporate changes at each iteration, which leads to better quality software.
6. **Continuous Improvement:** Continuous Improvement is a practice that involves reflecting on the development process and finding ways to improve it. It helps teams to identify and eliminate inefficiencies, and to continuously improve their processes and practices.

Benefits of Agile Unified Process (Advantage)

AUP has several benefits for software development teams and organizations. The following are the key benefits of AUP:

1. **Flexibility:** AUP is a flexible methodology that can be tailored to meet the specific needs and requirements of the project. It allows teams to adapt to changing requirements and to deliver software in an efficient and effective manner.
2. **High-Quality Software:** AUP emphasizes the importance of quality throughout the development lifecycle. It promotes the use of best practices and techniques that lead to high-quality software that meets the needs of the stakeholders.
3. **Faster Time-to-Market:** AUP delivers working software in small, incremental releases, which enables teams to deliver software faster and to respond to changing requirements quickly.
4. **Increased Collaboration:** AUP promotes collaboration between stakeholders, customers, and development teams. It encourages the use of collaborative tools and techniques that enhance communication and collaboration.
5. **Risk Management:** AUP is a risk-driven methodology that focuses on identifying, prioritizing, and mitigating risks throughout the development lifecycle. It helps teams

to manage risks proactively, which leads to a more predictable and successful project outcome.

Limitations of Agile Unified Process (Disadvantage)

AUP also has some limitations that must be considered when choosing a software development methodology. The following are the key limitations of AUP:

1. **Lack of Documentation:** AUP is a lightweight methodology that emphasizes working software over comprehensive documentation. It may not be suitable for projects that require extensive documentation for regulatory or legal reasons.
2. **Limited Scalability:** AUP may not be suitable for large, complex projects that require a high degree of coordination and management. It may be difficult to scale AUP to larger teams or organizations.
3. **Dependency on Team Experience:** AUP relies on the experience and expertise of the development team to deliver high-quality software. It may not be suitable for teams that lack the necessary skills or experience.
4. **Lack of Formal Process:** AUP is a flexible and adaptable methodology that may not provide the level of structure and formal process that some organizations require.

21.Explain Agile model vs Waterfall model

Agile Model	**Waterfall Model**
The Agile model follows an iterative and incremental development approach, allowing for continuous improvement and adaptation.	The Waterfall model follows a sequential development approach, with each phase completed before moving to the next.
The Agile model encourages customer collaboration and feedback, ensuring continuous customer involvement in the development process.	The Waterfall model involves limited customer involvement during the development phase, with less frequent opportunities for feedback.
The Agile model focuses on regular and frequent delivery of usable software increments.	The Waterfall model delivers the final software product at the end of the project.
In the Agile model, testing and quality assurance are integrated throughout the development process, ensuring continuous improvement of the software.	In the Waterfall model, testing typically occurs at the end of the development process, after all other stages are completed.
The Agile model is suitable for projects with evolving requirements and a need for adaptability and responsiveness to change.	The Waterfall model is suitable for projects with well-defined requirements and a focus on stability and predictability.

22.Difference between Waterfall Model and Spiral Model

Waterfall Model	**Spiral Model**
The_Waterfall model is simple and easy.	The spiral model_is a lot more complex.
The waterfall model works in a sequential method.	While the spiral model works in the evolutionary method.
In the waterfall model errors or risks are identified and rectified after the completion of stages.	In the spiral model errors or risks are identified and rectified earlier.
The waterfall model is adopted by customers.	While the spiral model is adopted by developers.
The waterfall model is applicable for small projects.	While the Spiral model is used for large projects.
In waterfall model requirements and early stage planning is necessary.	While in spiral model requirements and early stage planning is necessary if required.
Flexibility to change in waterfall model is Difficult.	Flexibility to change in spiral model is not Difficult.
There is high amount risk in waterfall model.	There is low amount risk in spiral model.
Waterfall model is comparatively inexpensive.	While cost of spiral model is very expensive.
Customer involvement is minimum in Waterfall Model	In the Spiral Model Customer involvement is high.
It requires least maintenance.	It requires typical maintenance.
It is based on linear framework type.	It is based on linear and iterative framework type.
Testing is done after the coding phase in the development life cycle.	Testing is done after the engineering phase in the development cycle.

Waterfall Model	**Spiral Model**
Reusability is extremely unlikely.	To a certain extent, reusability is possible.
Customer control over the administrator is very limited.	Customers have control over the administrator as compared to waterfall model.