JGi JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

School of Computer Science & IT

Department of BCA

SOFTWARE ENGINEERING (22BCA3C01)

MODULE 1: Software Product and Process

UNIT 1   :Software Product and Process

1. Define Software and explain its characteristics   ?

Software is Defined as :

(1) A set of 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.

Software Characteristics:-

1.Software is engineered.

2. Software does not wear out

3. Software is custom built.

Software is a logical rather than a physical system element. Therefore, software has characteristics that are considerably different than those of hardware:

1.Software is developed or engineered; it is not manufactured in the classical sense:

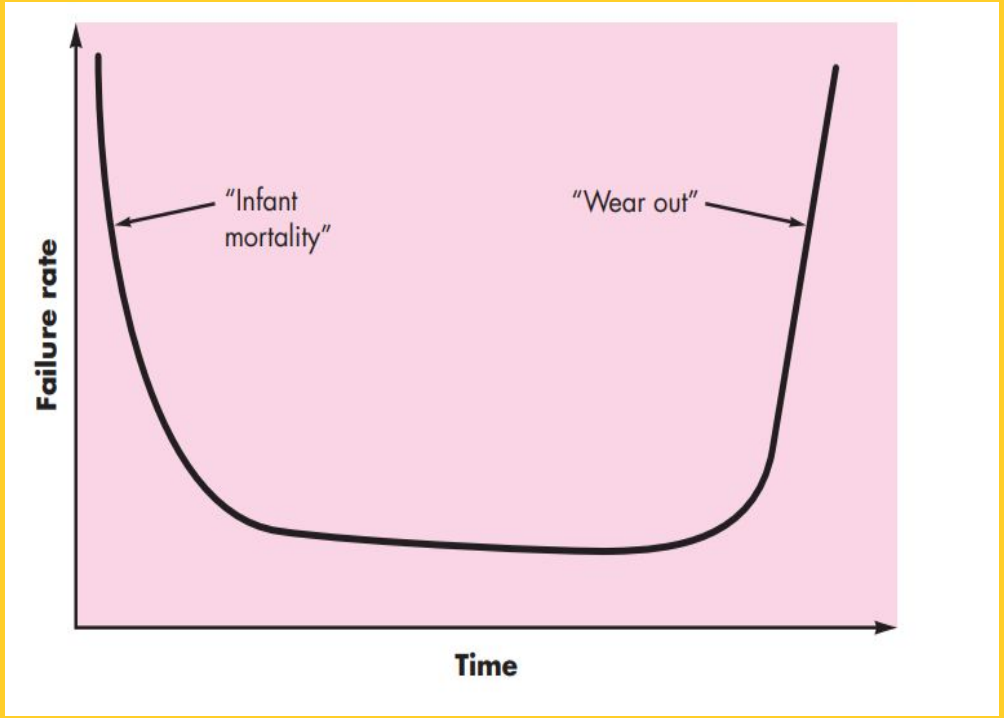
Although some similarities exist between software development and hardware manufacturing, the two activities are fundamentally different.

In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software.

Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different.

Both activities require the construction of a "product," but the approaches are different. Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects.

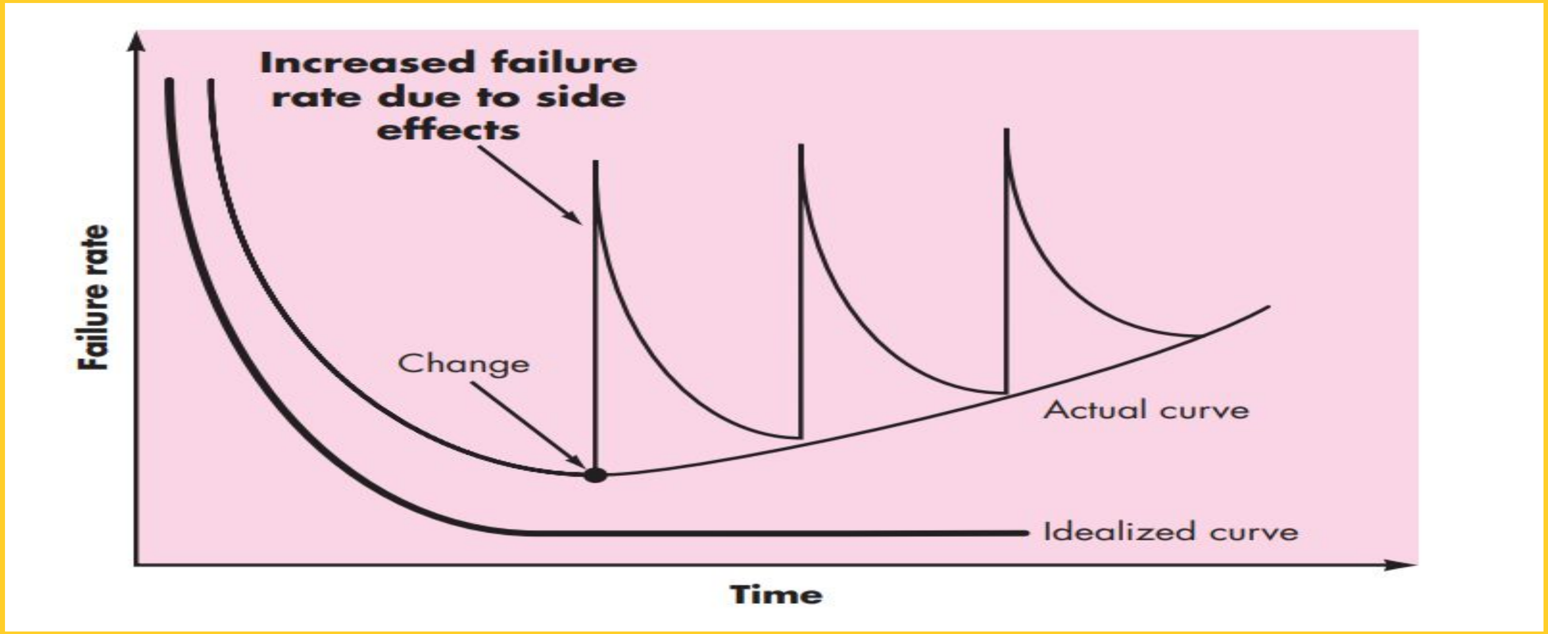
2.Software doesn't "wear out“:



Failure rate as a function of time for hardware

The picture 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(problems).



Failure rate as a function of time for Software

The above picture explains the software failure in development process. During development process, software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate curve to spike as shown in the "actual curve". Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again. Slowly, the minimum failure rate level begins to rise—the software is deteriorating due to change.

3.Although the industry is moving toward component-based assembly, most Software continues to be custom built:

As an engineering discipline evolves, a collection of standard design components is created. Standard screws and off-the-shelf integrated circuits are only two of thousands of standard components that are used by mechanical and electrical engineers as they design new systems.

The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a design, that is, the parts of the design that represent something new. In the hardware world, component reuse is a natural part of the engineering process.

In the software world, it is something that has only begun to be achieved on a broad scale.

A software component should be designed and implemented so that it can be reused in many different programs.

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.

Categories of Software :

System software

■ A collection of programs written to service other programs. System software e.g., compilers, editors, and file management utilities, operating system, networking software, drivers etc.

Application software

■ Programs that process business data in a way that facilitates business operations or management decision making and solve a specific business need. Microsoft application software, music applications, skype, google meet, zoom etc.

Engineering/Scientific software

■ A broad array of “number-crunching programs that range from astronomy to volcanology, from automotive stress analysis to orbital dynamics, and from computer aided design to molecular biology, from genetic analysis to astromony. MATLAB, AUTOCAD,ORCAD etc.

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.

■ e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems.

Product-line software

■ Designed to provide a specific capability for use by many different customers. Product-line software can focus on a limited marketplace (e.g., inventory control products) or address mass consumer. e.g., inventory control products) or address mass consumer.

Web/Mobile applications

■ This network-centric software category spans a wide array of applications and encompasses both browser-based apps and software that resides on mobile devices.

AI software

■ Makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis. Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

Legacy Software

- The older software programs — often referred to as legacy software — have been the focus of continuous attention and concern.
- Legacy software can be described in the following way:

Legacy software systems . . . were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation of such systems cause headaches for large organizations who find them costly to maintain and risky to evolve.

Further, "many legacy systems remain supportive to core business functions and are 'indispensable' to the business." Hence, legacy software is characterized by longevity and business criticality.

- Legacy software has a characteristics – Poor Quality e.g., poor user interface, poor performance, poor documentation etc.

As time passes, legacy systems often evolve for one or more of the following reasons:

■ software must be adapted to meet the needs of new computing environments or technology.
■ software must be enhanced to implement new business requirements.
■ software must be extended to make it interoperable with other more modern systems or databases.
■ software must be re-architected to make it viable within a network environment.

The Changing Nature of Software

Four broad categories of software are evolving and will continue to dominate the industry.

WebApps

■ Modern WebApps are much more than hypertext files with a few pictures

■ WebApps are augmented with tools like XML and Java to allow Web engineers including interactive computing capability

■ WebApps may standalone capability to end users or may be integrated with corporate databases and business applications

■ Semantic web technologies (Web 3.0) have evolved into sophisticated corporate and consumer applications that encompass semantic databases that require web linking, flexible data representation, and application programmer interfaces (API's) for access

■ The aesthetic nature of the content remains an important determinant of the quality of a WebApp.

Mobile Apps

■ Reside on mobile platforms such as cell phones or tablets .

■ Contain user interfaces that take both device characteristics and location attributes .

■ Often provide access to a combination of web-based resources and local device processing and storage capabilities .

■ Provide persistent storage capabilities within the platform .

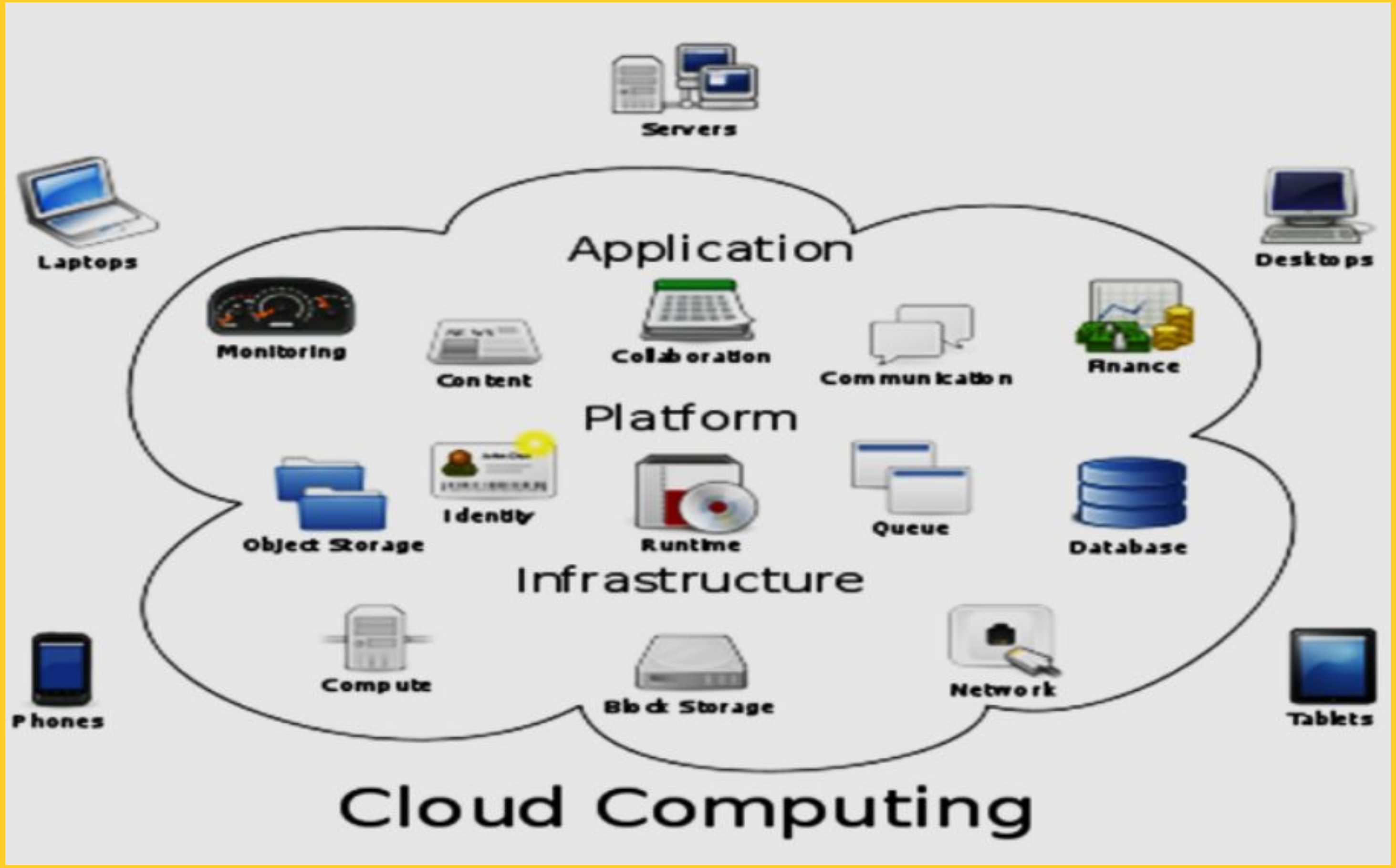
■ A mobile web application allows a mobile device to access to web-based content using a browser designed to accommodate the strengths and weaknesses of the mobile platform .

■ A mobile app can gain direct access to the hardware found on the device to provide local processing and storage capabilities .

■ As time passes these differences will become blurred .

Cloud Computing

- Cloud computing provides distributed data storage and processing resources to networked computing devices



- Computing resources reside outside the cloud and have access to a variety of resources inside the cloud
- Cloud computing requires developing an architecture containing both frontend and backend services
- Frontend services include the client devices and application software to allow access
- Backend services include servers, data storage, and server-resident applications
- Cloud architectures can be segmented to restrict access to private data

Product Line Software

■ Product line software is a set of software-intensive systems that share a common set of features and satisfy the needs of a particular market

■ These software products are developed using the same application and data architectures using a common core of reusable software components

■ A software product line shares a set of assets that include requirements, architecture, design patterns, reusable components, test cases, and other work products

■ A software product line allow in the development of many products that are engineered by capitalizing on the commonality among all products within the product line.

Characteristics of WebApps

■ **Data driven**: The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.

■ **Content sensitive:** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.

■ **Continuous evolution**: Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.

■ **Immediacy:** Although immediacy—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.

■ **Security** : Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.

■ **Aesthetics**: An undeniable part of the appeal of a WebApp is its look and feel.

Defining Software Engineering

- Software engineering is the systematic application of engineering approaches to the development of software.
- Before the product is developed, its requirements are specified, analyzed, designed, verified and tested.
- The aim is to produce quality software that satisfies its requirements, that is delivered on time and within the approved budget.

Seminal / Text Book Definition:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

- Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

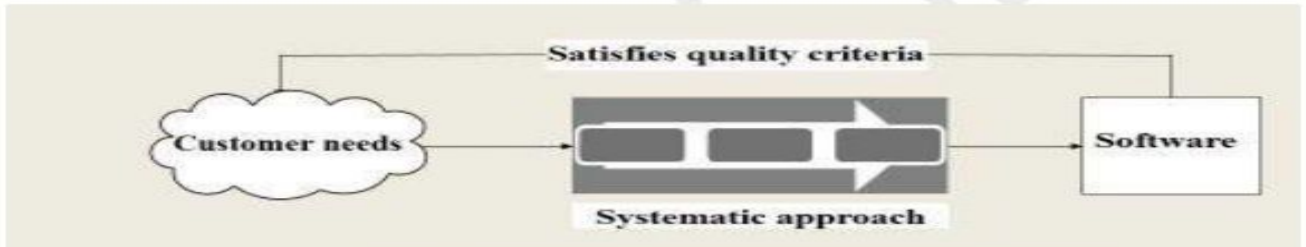
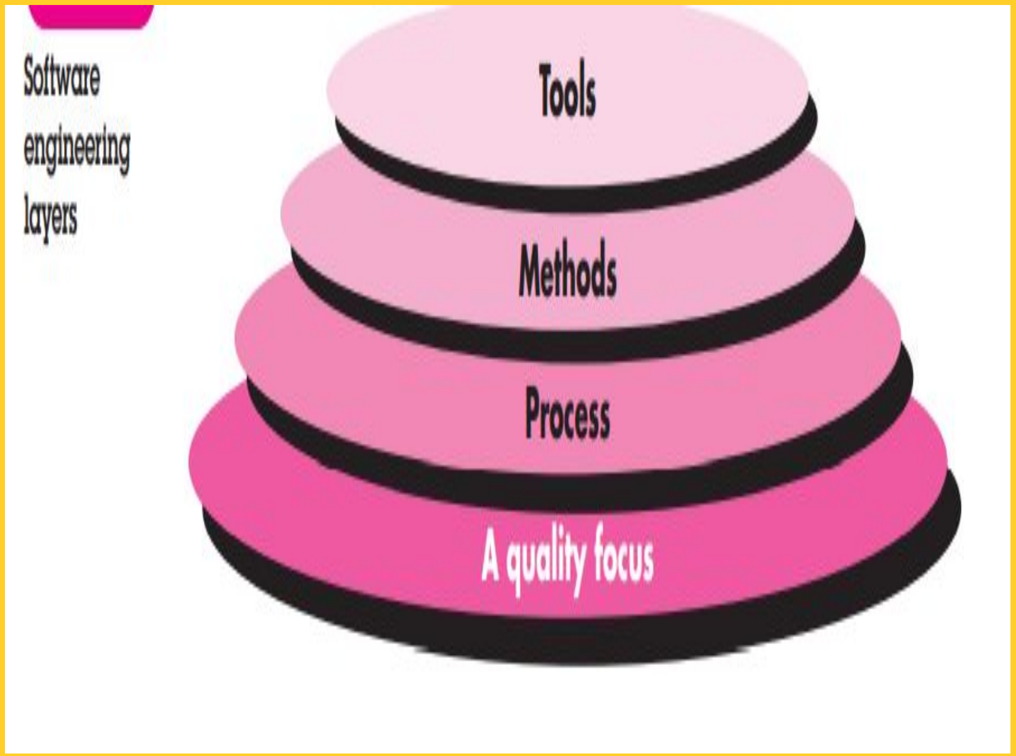


Figure 1.3: Software engineering view

SOFTWARE ENGINEERING - A LAYERED TECHNOLOGY

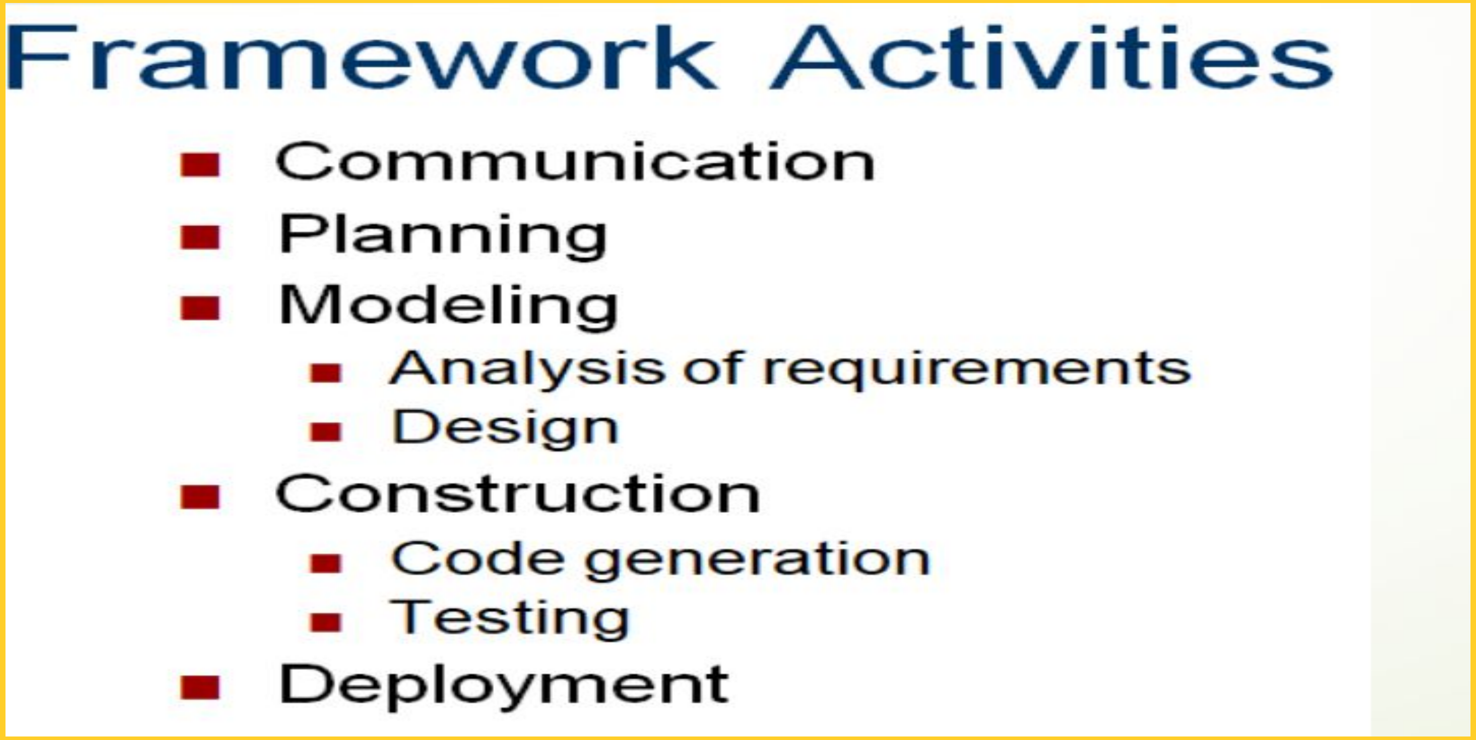


Software engineering is a layered technology.

- The bedrock / fundamental principle that supports software engineering is a quality focus.
- The foundation for software engineering is the process layer. Process defines a framework that must be established for effective delivery of software.
- The software process forms the basis for management control of software projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms, etc.) are produced, milestones are established, quality is ensured, and change is properly managed.
- 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 tools provide automated or semi-automated support for the process and the methods.

A Process Framework

- A process framework establishes the foundation for a complete software engineering 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 is required for representing common process activities.

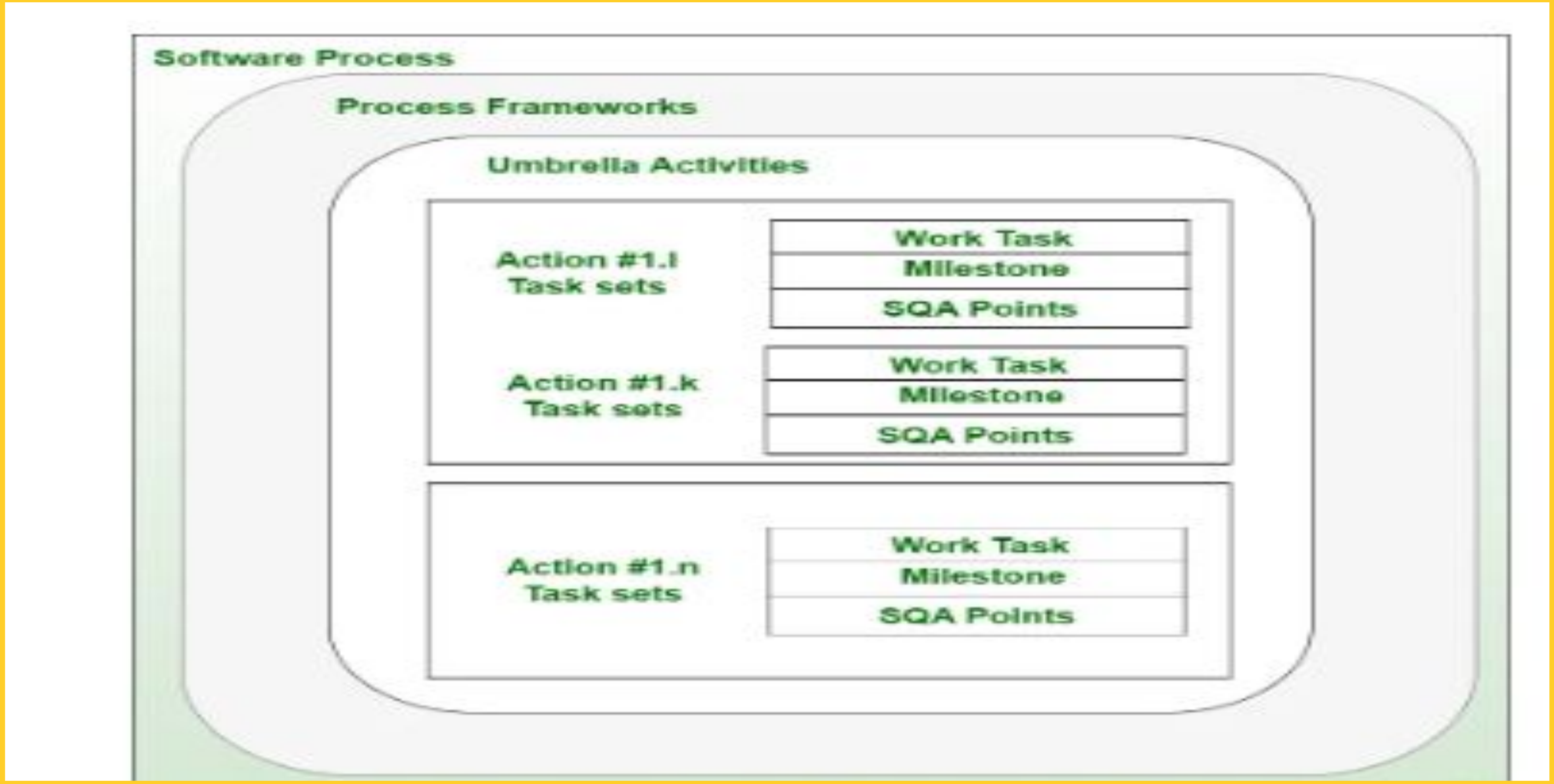


Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

The Software process framework is required for representing common process activities. Five framework activities are described in a process framework for software engineering. Communication, planning, modeling, construction, and deployment are all examples of framework activities. Each engineering action defined by a framework activity comprises a list of needed

- ✔ work outputs,
- ✔ project milestones,
- ✔ software quality assurance (SQA) points.



Software Process includes:
Tasks: They focus on a small, specific objective.
Action: It is a set of tasks that produce a major work product.
Activities: Activities are groups of related tasks and actions for a major Objective.

Process Framework Activities:

1. **Communication**

- Communication involves gathering requirements from customers and stakeholders to determine the system's objectives and the software's requirements.

Activities:

- **Requirement Gathering**: Engaging with consumers and stakeholders through meetings, interviews, and surveys to understand their needs and expectations.
- **Objective Setting**: Clearly defining what the system should achieve based on the gathered requirements.

2. Planning

- Planning involves establishing an engineering work plan, describing technical risks, listing resource requirements, and defining a work schedule.

Activities:

- **Work Plan**: Creating a detailed plan that outlines the tasks and activities needed to develop the software.
- **Risk Assessment**: Identifying potential technical risks and planning how to mitigate them.
- **Resource Allocation**: Determining the resources (time, personnel, tools) required for the project.
- **Schedule Definition**: Setting a timeline for completing different phases of the project.

3. Modeling

- Modeling involves creating architectural models and designs to better understand the problem and work towards the best solution.

Activities:

- **Analysis of Requirements**: Breaking down the gathered requirements to understand what the system needs to do.
- **Design**: Creating architectural and detailed designs that outline how the software will be structured and how it will function.

4. Construction

- Construction involves creating code, testing the system, fixing bugs, and confirming that all criteria are met.

The software design is mapped into a code by:

- **Code Generation**: Writing the actual code based on the design models.
- **Testing**: Running tests to ensure the software works as intended, identifying and fixing bugs.

5. Deployment

- Deployment involves presenting the completed or partially completed product to customers for evaluation and feedback, then making necessary modifications based on their input.

Activities:

- **Product Release**: Delivering the software to users, either as a full release or in stages.
- **Feedback Collection**: Gathering feedback from users about their experience with the software.
- **Product Improvement**: Making changes and improvements based on user feedback to enhance the product.

Umbrella Activities: In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.

Umbrella Activities that take place during a software development process for improved project management and tracking.

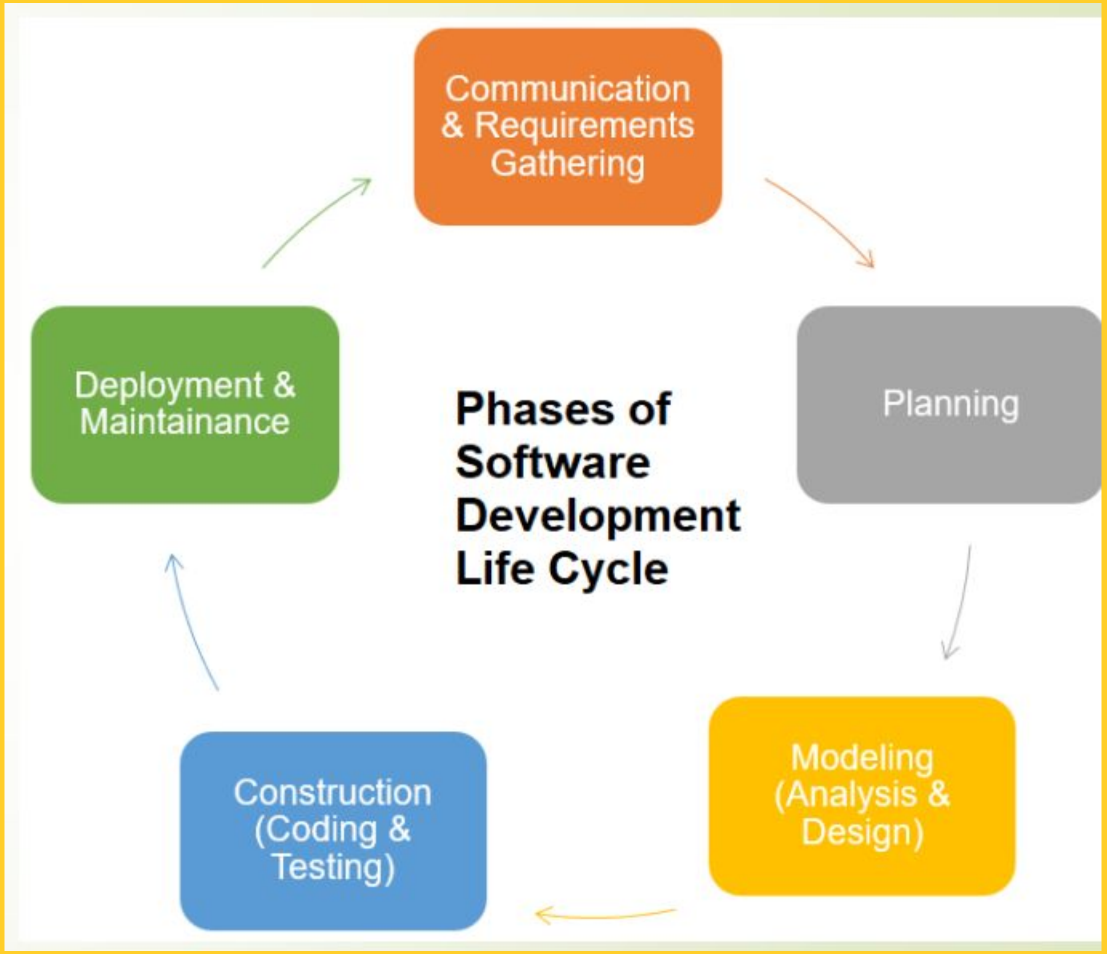
1. **Software project tracking and control:** This is an activity in which the team can assess progress and take corrective action to maintain the schedule. Take action to keep the project on time by comparing the project’s progress against the plan.
2. **Risk management:** The risks that may affect project outcomes or quality can be analyzed. Analyze potential risks that may have an impact on the software product’s quality and outcome.
3. **Software quality assurance:** These are activities required to maintain software quality. Perform actions to ensure the product’s quality.
4. **Formal technical reviews:** It is required to assess engineering work products to uncover and remove errors before they propagate to the next activity. At each level of the process, errors are evaluated and fixed.
5. **Software configuration management:** Managing of configuration process when any change in the software occurs.
6. **Work product preparation and production:** The activities to create models, documents, logs, forms, and lists are carried out.
7. **Reusability management:** It defines criteria for work product reuse. Reusable work items should be backed up, and reusable software components should be achieved.
8. **Measurement:** In this activity, the process can be defined and collected. Also, project and product measures are used to assist the software team in delivering the required software.

QUESTIONS

1. Define Software and explain its characteristics ?
2. What are the Categories of Software ?
3. Define Software Engineering ?and explain software engineering approach (layered approach)?
4. Define Legacy software ?
5. Write a note on changing nature of software ?
6. write the Characteristics of WebApps ?
7. Explain process frame work activities and Umbrella Activities in detail?

Software Development Life Cycle (SDLC)

- The software development lifecycle (SDLC) is a framework (i.e. a systematic approach for building software) that development teams use to produce high-quality software in a cost-effective way within the time frame that satisfies customer expectations.
- It defines the step-by-step approach to developing successful software from gathering the initial requirements for a new product to the deployment and maintenance of the software product.
- Started in 1960s, it has undergone several refinements.
- It has several distinct phases that suggest what activities to be carried out in each phase.


Communication & Requirements Gathering
Planning
Modeling (Analysis & Design)
Construction (Coding & Testing)
Deployment & Maintainance
Phases of Software Development Life Cycle

Phase-1: Communication & Requirement Gathering

- During this phase, all the relevant information is collected from the customer to develop a product as per their expectation.
- The intent is to understand stakeholders’ objectives for the project and to gather requirements that help define software features and functions.
- Understand the application domain, business process, users / stakeholders’ requirements and record it with clarity.
- It is Carried out by a team of technical and domain experts.
- Use various methods: interview, observation, reading documents/records, questionnaire etc.
- Conduct a feasibility analysis: Economical, Financial, Technical, Operational
- Once the requirements are clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also reviewed by the customer for future reference.

<u>Phase-2: Planning</u>

- A software project is a complicated work. So, a proper planning and scheduling is created for the software development project that helps guide the team.
- A software project plan defines the software engineering work by describing
  - the technical tasks to be conducted,
  - the risks that are likely to occur,
  - the resources that will be required,
  - the work products to be produced, and
  - a work schedule.

<u>Phase-3: Modeling or Designing:</u>

The software requirements are analyzed to develop various **analysis and design models.**

- **<u>Analysis Models</u>**:
  - several models like Use Case Model, Interaction Models, Activity Diagram etc. are used .
  - An overall system **architecture** is also created.

- **Testing:**

- Testing starts once the coding is complete and the modules are released for testing.
- Testing Strategy used: **Unit Testing and Integration Testing**.
- In this phase, the developed software is tested thoroughly by QA team and any **defects found** are assigned to developers to get them fixed.
- The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.
- Testers **refer to SRS document** to make sure that the software is as per the customer’s standard.
- Testing Strategy used: **White-Box and Black-Box**.

Phase-5: Deployment & Maintenance

- Once the product is tested, it is deployed in the production (or user or customer) environment for evaluation and user acceptance testing (i.e. Beta Testing).
- Based on the feedback given by the users, necessary modifications are made and the final software is released. Once the software released, maintenance occurs.
- Maintenance is a repeated activity happens over and over again in the software life.

- Once the system is deployed, and customers start using the developed system, following 3 activities occur:
  - Bug fixing - bugs are reported because of some scenarios might not be tested.
  - Upgrade - Upgrading the application to the newer versions of the Software (i.e. improvements terms of technology, designs/ logic)
  - Enhancement - Adding some new features into the existing software
- The main focus of this SDLC phase is to ensure that customer needs continue to be met and th the system continues to perform over a period of time.

QUESTIONS

1) What is SDLC?
2) What are the phases of SDLC?
3) Describe a particular SDLC phase of your choice.
4) Why is it called a life cycle?

PRESCRIPTIVE PROCESS Models:

- Prescriptive process models were originally proposed to bring order to the software development.
- Prescriptive process models define a prescribed set of **process elements** and a **predictable process workflow**.
- Each **process model** follows a **Series of steps** unique to its type to **ensure success** in the process of **software development**.

Characteristics of Software Process

- _**Predictability** of a process determines how accurately the outcome of following that process in a project can be predicted before the project is completed.
- **Support Testability and Maintainability** in the life of software the maintenance costs generally exceed the development costs.
- **Support Change**
- **Early Defect Removal**
- **Process Improvement and Feedback**

Software Process Model :

- A software process model is an abstract representation of the software development process.
- A prescriptive process model strives for structure and order in software development, advocate an orderly approach to software engineering.
- It specifies the various stages of the software process and the order / sequence in which they are carried out. How often they are carried out.
- The goal of a software process model is to provide guidance for controlling and coordinating the process activities to achieve the quality software as effectively as possible.
- There are several process models that meet different requirements / circumstances of software development project.

❑The most popular models are as follows:

- Waterfall model, Incremental model, RAD model, Prototype model, Spiral model, V model, Agile model, Rational Unified Process

❑**Factors (to choose a process model)**:

- Nature of Requirements – Simple/ complex, known/ not known with clarity, changes frequently
- Customer involvement- customer participation needed / not needed
- Developers experience- experiences, expertise of development team in technology, domain
- Development Team size- large team, small team

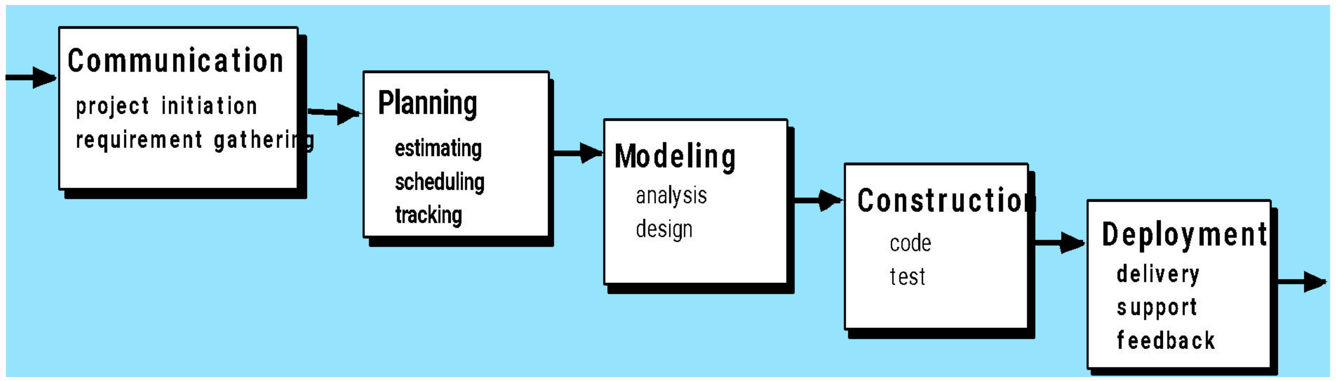
Types of Process Models

- The Waterfall Model
- V-Model
- Incremental Process Models
- RAD model
- Prototyping
- Spiral Model
- Agile Model

Waterfall Model

The oldest model and suffers from several criticisms.

- The waterfall model is a **sequential, plan driven-process** where you must plan and schedule all your activities before starting the project. Each activity in the waterfall model is represented as a separate phase arranged in linear order.
- Customer requirements are gathered at the beginning of the project, and then a sequential project plan is created to accommodate those requirements.
- It assumes that customer requirements are reasonably well-understood, complete and fixed.
- Each phase is carried out completely (for all requirements) before proceeding to the next. Each of these phases produces one or more documents that need to be approved before the next phase begins.
- The process is strictly sequential - no backing up or repeating phases.
- The waterfall model is so named because each phase of the project cascades into the next, following steadily down like a waterfall.



Advantages:

- Simple, easy to understand and follow. Highly structured, therefore good for beginners.
- After specification is complete, low customer involvement required.

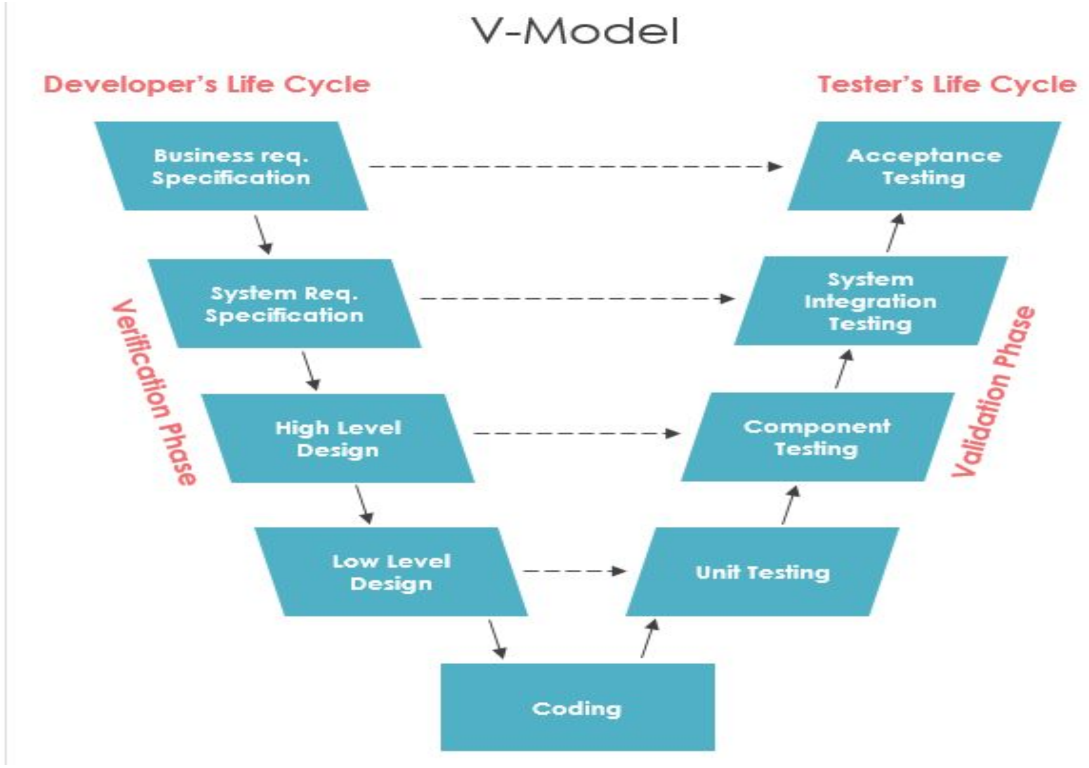
Disadvantages:

- Inflexible - can't adapt to changes in requirements. In reality, projects are rarely sequential.
- The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to change.

Sl no	advantages	disadvantages	When to use
1	It allows for departmentalization and managerial control.	it assumes that no development error is ever committed by the engineers during any of the life cycle phases. However, in practical development environments, the engineers do commit a large number of errors in almost every phase of the life cycle	Requirements are well understood
2	Simple and easy to understand and use	It is difficult for customer to state all requirements explicitly(without any change)	Automation of existing manual system
3	Phases are processed and completed one at a time.	Working version of software will not be available until late in project span	Short duration project
4	Works well for smaller projects where requirements are very well understood.	User feedback not encouraged Not a good model for complex and object-oriented projects.	
5	A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.	. Once a defect is detected, the engineers need to go back to the phase where the defect had occurred and redo some of the work done during that phase and the subsequent phases to correct the defect and its effect on the later phases	

V-Model

- The V-model is considered as an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding / implementation phase, to form the typical V shape.
- The V-model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.
- The horizontal and vertical axes represent project completion time and level of abstraction respectively. Similarly to waterfall model the process steps follow each other in a sequential order but V-model allows the parallel execution of development and testing activities.



- It visualizes how Verification and Validation actions carried out in the software development.
- A developer has to verify if the requirements of a specific development phase are met. Doing the things the right way.
- A tester has to validate that the system meets the needs of the user, the customer or other stakeholders.
- The model emphasizes both verifications and validations.
- Typically, at a higher abstraction level, more validation than verification is conducted.

INCREMENTAL PROCESS MODELS:

The process models in this category tend to be amongst most widely used(and effective) in industry .

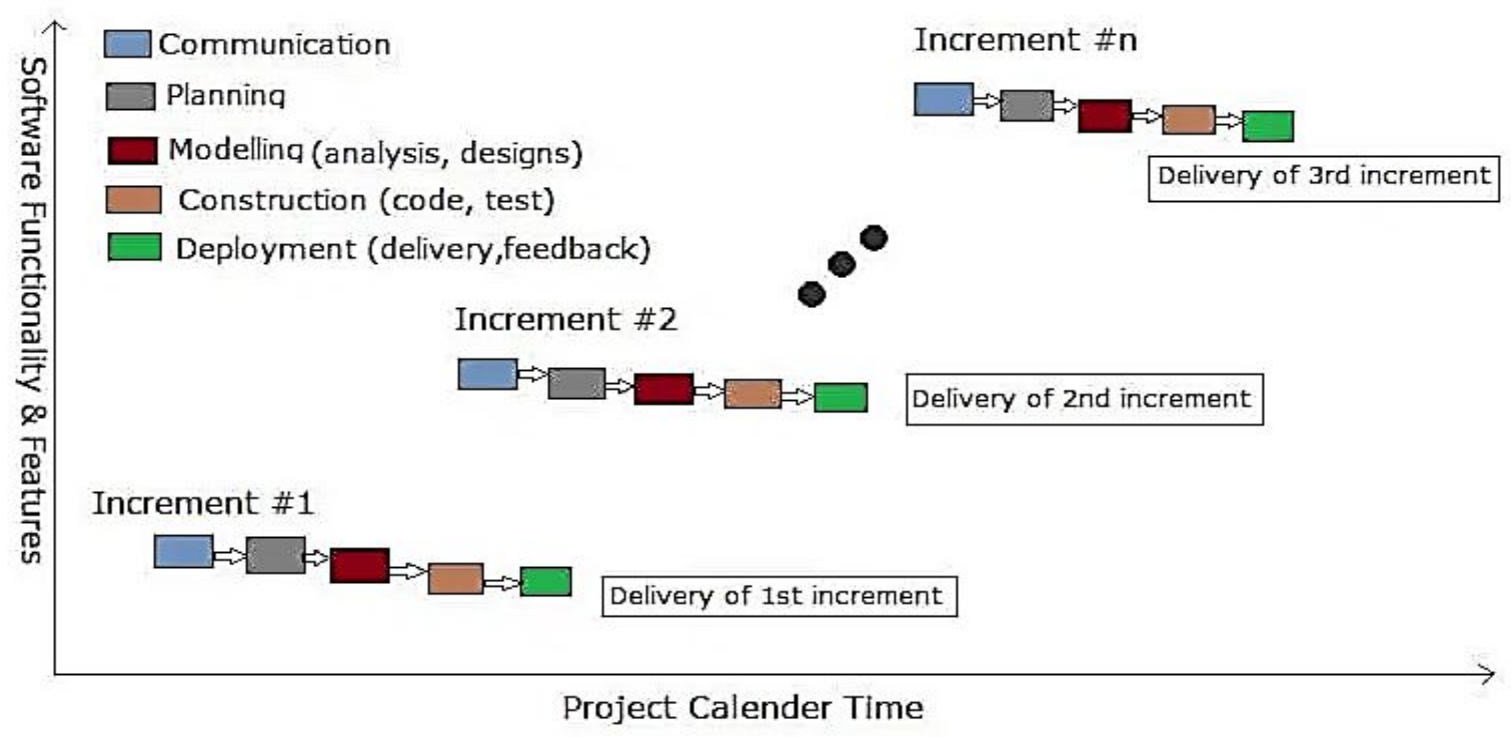
Models among them are –

1. Incremental model

2. RAD model

Incremental Model

- It combines the elements of waterfall model in an iterative fashion. The requirements are well-understood, but there is a compelling need to provide a software with limited set of functionalities quickly and then expand the functionalities with later software releases.
- The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments.
- The subsequent increments expand on the previous ones until everything has been updated and implemented.


Communication
Planning
Modelling (analysis, designs)
Construction (code, test)
Deployment (delivery,feedback)
Increment #1
Delivery of 1st increment
Increment #2
Delivery of 2nd increment
Increment #n
Delivery of 3rd increment
Software Functionality & Features
Project Calender Time

- Development team is small or adequate staff is unavailable for complete implementation.
- Project involves some sort of risks.
- Customer get a working software early and provides feedback for later increments.
- The incremental model is great for projects that have loosely-coupled parts and projects with complete and clear requirements.

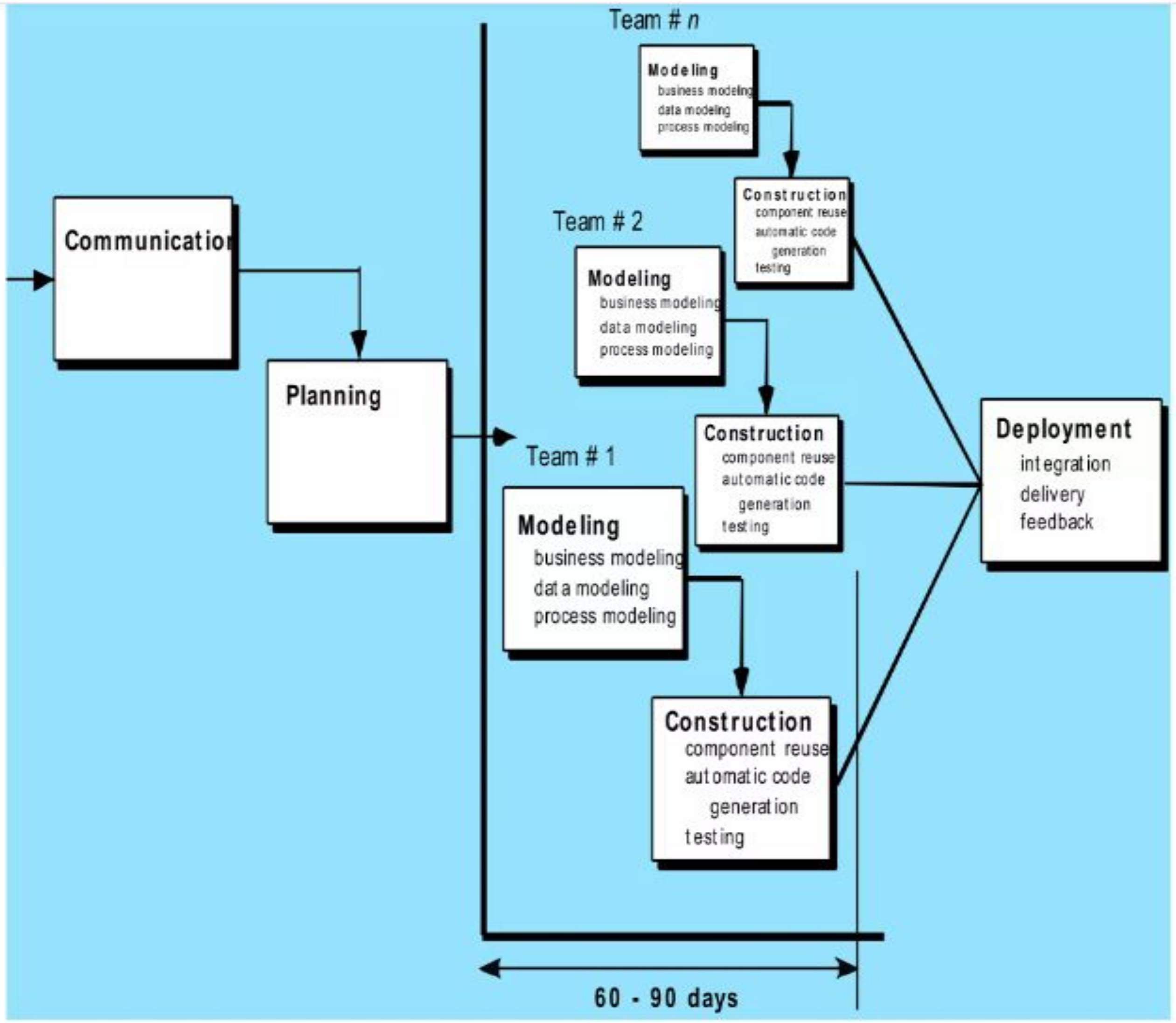
Advantages of using Incremental process model :

- It is easy for breakdown of tasks because of divide and conquer approached used.
- It has lowers initial delivery cost.
- It has incremental resource deployment.
- It can deduct errors easily because core modules are used by the customer from the beginning of the phase and then 5. These are tested thoroughly.
- It is good to use when requirements are known up-front.
- It is good to use when projects having lengthy developments schedules.
- It is good to use when projects uses new Technology.
- It is good to use when Funding Schedule, Risk, Program Complexity, or need for early realization of benefits.
- It generates working software quickly and early during the software life cycle.
- It is more flexible and less costly to change scope and requirements.
- It is easy to manage each iteration.
- It is easy to manage risk because of iterations.

Disadvantages of using Incremental process model :

- It requires a good planning designing.
- It is costlier than waterfall model.
- Definition of system should be complete and clear.
- Increased complexity: As the project is developed incrementally, it can become more complex as each increment is added. This can make it harder to manage and maintain, as well as increase the risk of errors and bugs.
- Higher costs: Since each increment requires its own planning, design, coding, testing, and deployment, the overall cost of the project can be higher than with other development methodologies.
- Difficulty in tracking progress: With multiple increments being developed simultaneously, it can be challenging to track the progress of the project as a whole. This can make it harder to identify potential issues early on and take corrective action.
- Increased communication overhead: With each increment being developed by a different team or individual, there can be a significant increase in the communication overhead required to ensure that everyone is on the same page.
- More time spent on testing: With each increment requiring its own testing phase, more time may be spent on testing overall, which can delay the project’s completion.

RAD Model(Rapid Application Development model)



- It is a high-speed adaption of the waterfall model, in which rapid development is achieved using a component-based construction approach and RAD tools.
- When requirements are well-understood and a fully functional system has to be developed within a very short time period, RAD model is prescribed.
- It requires highly skilled developers with clearly separable software modules.
- Requires large number of developers for several RAD teams.
- Each RAD team develops a major part of the software which then integrated to form a whole.
- RAD is not suitable:
  - If software can not be modularized properly.
  - If software involves risk and high performance

	advantages	disadvantages	When to use
1	Progress can be measured. Iteration time can be short with use of powerful RAD tools.	Dependency on technically strong team members for identifying business requirements. Sufficient human recourse is needed to create right no of RAD team	Suitable for project requiring shorter development times.
2	Integration from very beginning solves a lot of integration issues.	Only system that can be modularized can be built using RAD. Sufficient human recourse is needed to create right no of RAD team	Requirements are understood and project scope is constrained
3	Encourages customer feedback, Changing requirements can be accommodated.	Requires highly skilled developers/designers.High dependency on modeling skills.Performances issue	Fully functional system is to be delivered in short span of time
4	Reduced development time. Increases reusability of components	Management complexity is more. Suitable for systems that are component based and scalable	For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
5	Productivity with fewer people in short time.	Requires user involvement throughout the life cycle.	
		Inapplicable to cheaper projects as cost of modeling and automated code generation is very high	

Evolutionary Process Models :

- The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users can get access to the product at the end of each cycle.

Some of the models are :

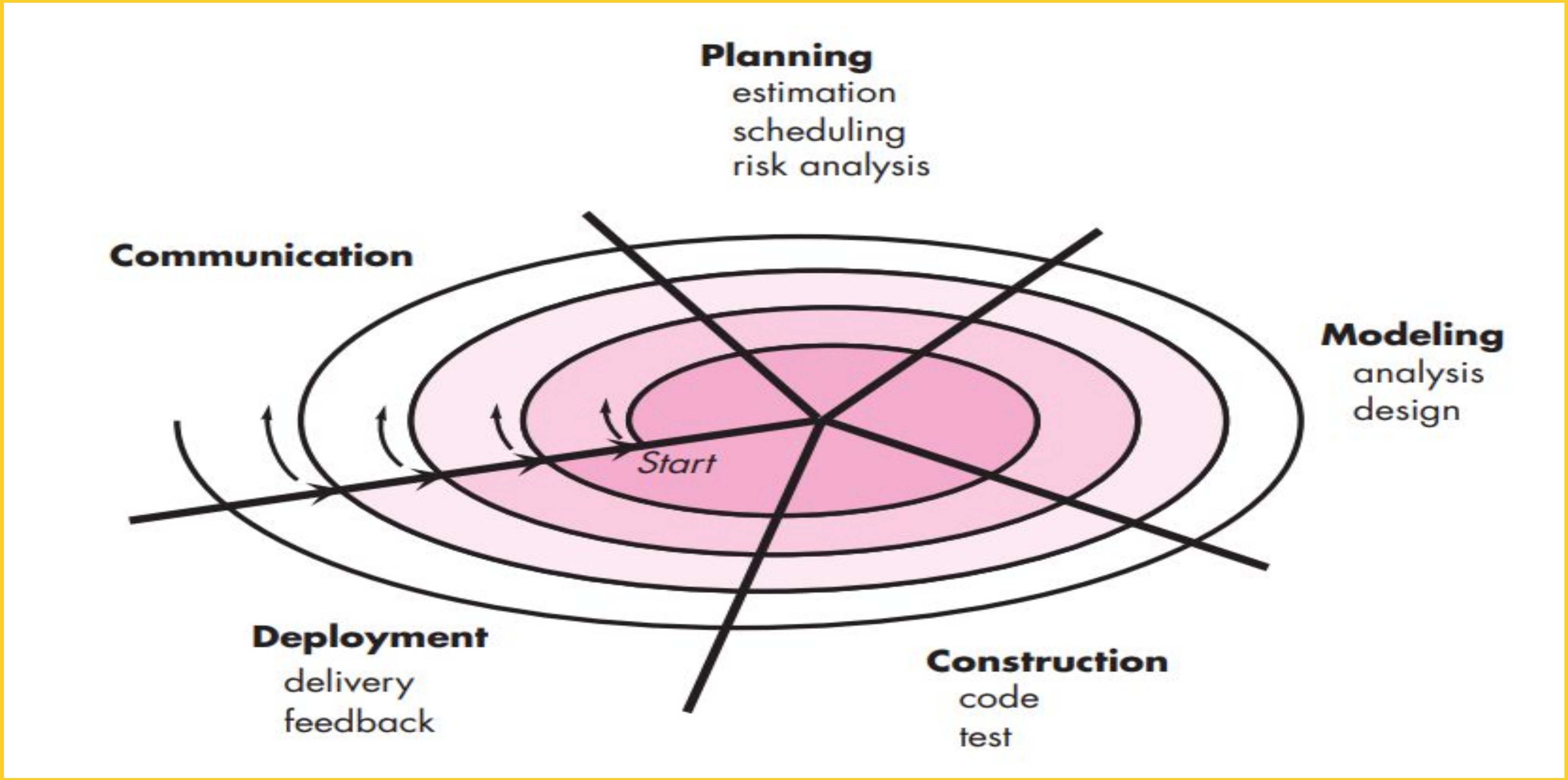
1. Prototyping model
2. Spiral model

Write a note on spiral models and prototyping model? Or write about di evolutionary process models ?

- Proposed by Barry Boehm in 1986, is a risk-driven software development process model for dealing with the shortcomings in the traditional waterfall model.
- Spiral model is a software development process model,which has characteristics of both **iterative and waterfall models**.
- This model is used in projects which are **large and complex.**
- This model was named **spiral,**as shown in the figure, as it looks like a spiral, in which a long curved line **starts from the center point and makes many loops around it**. The number of loops in the spiral is not decided in advance but it depends on the size of the project and the changing requirements of the user. Each loop of the spiral is called a phase of the software development process.
- A software project goes through these loops again and again in iterations. After each iteration a more and more complete version of the software is developed.
- The most special thing about this model is that **risks are identified in each phase and they are resolved through prototyping. This feature is also called Risk Handling.**

Since it also includes the approaches of other SDLC models, it is also called Meta Model.

Spiral Model = Concept of Waterfall lifecycle+ Iterative model.



Spiral model

In Spiral Model the entire process of software development is described in four phases which are repeated until the project is completed.

Those phases are as follows:-

- **Determining objectives and alternate solutions:** In the first phase, whatever requirements the customer has related to the software are collected. On the basis of which objectives are identified and analyzed and various alternative solutions are proposed.
- **Identifying and resolving risks:** In this phase, all the proposed solutions are assessed and the best solution is selected. Now that solution is analyzed and the risks related to it are identified. Now the identified risks are resolved through some best strategy.
- **Develop and test:** Now the development of software is started. In this phase various features are implemented, that is, their coding is done. Then those features are verified through testing.
- **Review and plan for the next phase:** In this phase the developed version of the software is given to the customer and he evaluates it. Gives his feedback and tells new requirements. Finally planning for the next phase (next spiral) is started.

Advantages of Spiral Model

- If we have to add additional functionality or make any changes to the software, then throug this model we can do so in the later stages also.
- Spiral model is suitable for large and complex projects.
- It is easy to estimate how much the project will cost.
- Risk analysis is done in each phase of this model.
- The customer can see the look of his software only in the early stages of the development process.
- Since continuous feedback is taken from the customer during the development process, the chances of customer satisfaction increases.

Disadvantage of Spiral Model

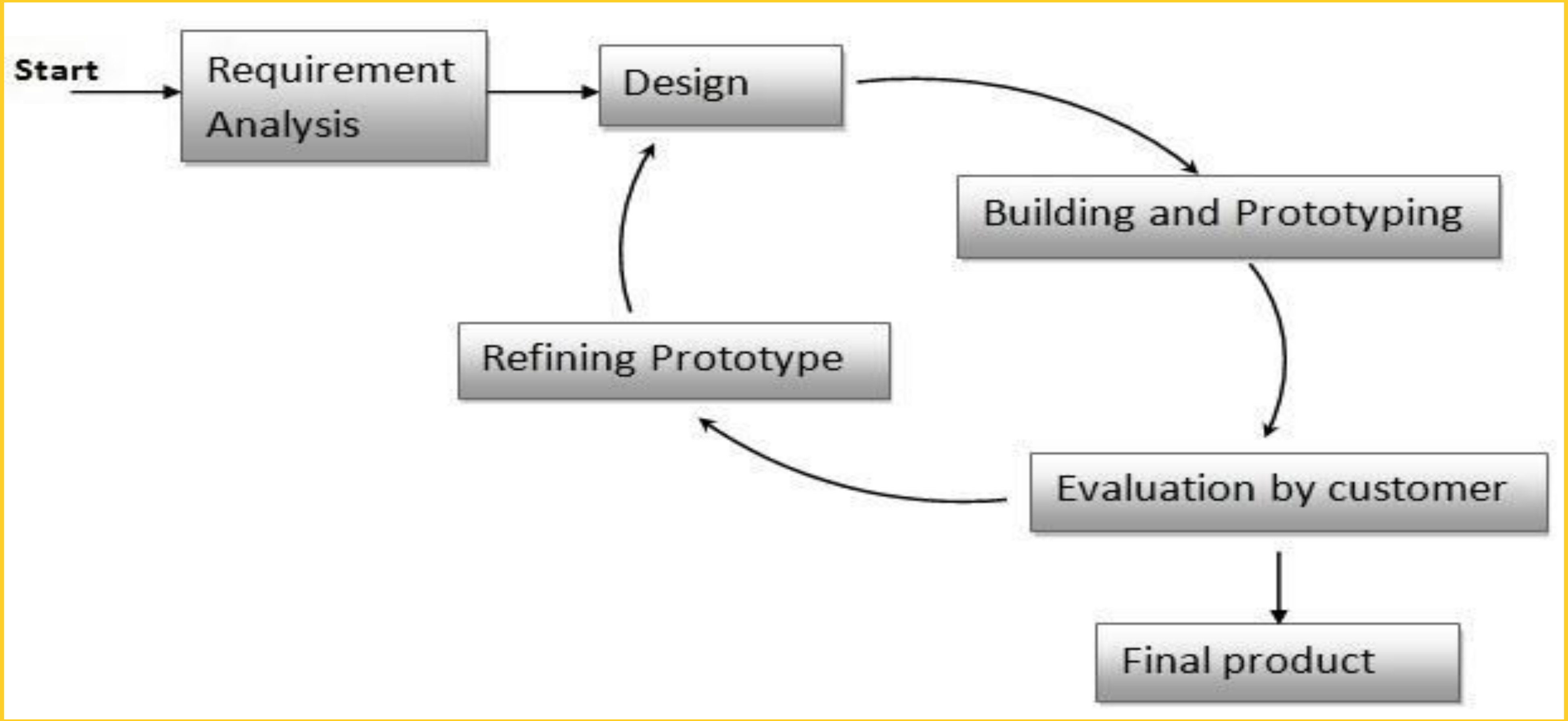
- **This is the most complex model of SDLC, due to which it is quite difficult to manage.**
- **This model is not suitable for small projects.**
- **The cost of this model is quite high.**
- **It requires more documentation than other models.**
- **Experienced experts are required to evaluate and review the project from time to time.**
- **Using this model, the success of the project depends greatly on the risk analysis phase.**

Prototype model

Prototype model is an activity in which prototypes of software applications are created. First a prototype is created and then the final product is manufactured based on that prototype.

- The prototype model was developed to overcome the shortcomings of the waterfall model.
- This model is created when we do not know the requirements well.
- The specialty of this model is that this model can be used with other models as well as alone.

One problem in this model is that if the end users are not satisfied with the prototype model, then a new prototype model is created again, due to which this model consumes a lot of money and time.



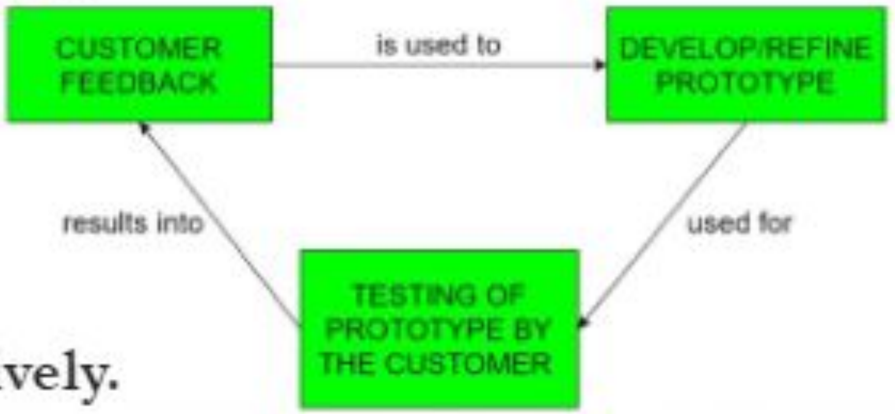
Prototype model

The prototype model has the following phases

- **Requirement gathering:** The first step of prototype model is to collect the requirements, although the customer does not know much about the requirements but the major requirements are defined in detail.
- **Build the initial prototype:** In this phase the initial prototype is built. In this some basic requirements are displayed and user interface is made available.
- **Review the prototype:** When the construction of the prototype is completed, it is presented to the end users or customer and feedback is taken from them about this prototype. This feedback is used to further improve the system and possible changes are made to the prototype.
- **Revise and improve the prototype:** When feedback is taken from end users and customers, the prototype is improved on the basis of feedback. If the customer is not satisfied with the prototype, a new prototype is created and this process continues until the customer gets the prototype as per his desire.

ADVANTAGES OF PROTOTYPING MODEL

- Prototype model need not known the detailed input, output, processes, adaptability of operating system and full machine interaction.
- Good where requirement are changing.
- Customers are actively involved in development.
- Prototypes can be changed and even discarded.
- Errors can be detected much earlier as the system is made side by side.
- Flexibility in design.
- Helps team member to communicate effectively.



Disadvantages of Prototype model

- When the first version of the prototype model is ready, the customer himself often wants small fixes and changes in it rather than rebuilding the system. Whereas if the system is redesigned then more quality will be maintained in it.
- Many compromises can be seen in the first version of the Prototype Model.
- Sometimes a software developer may make compromises in his implementation, just to get the prototype model up and running quickly, and after some time he may become comfortable with making such compromises and may forget that it is completely inappropriate to do so.

QUESTIONS

1)What is a Process Model?

2)What is a waterfall model and write the drawbacks of waterfall model?

3)When do you require incremental model?

4)When do you require RAD model?

5)Compare Incremental model with RAD model.

6)Which process model is used?

Scenarios:

1) Projects developed by students
2) Organization head wants to develop a software for better management of the organization but exactly fails to detail his requirements.

7)Which is the most important feature of spiral model?

8)Name some evolutionary process models and explain in detail?

9)What is the significance of V-model?

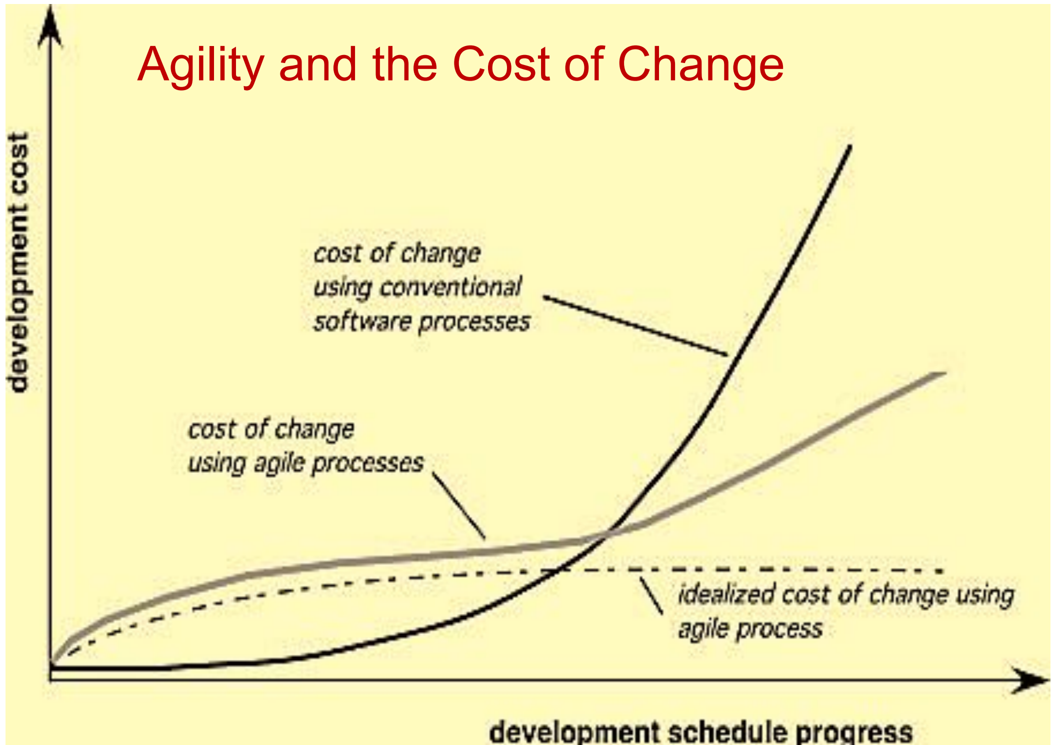
Define Change and Agility in Software Development

Change is very much inevitable in software development. Change can be due to

- changes in the requirements of the software being built,
- changes to the team members,
- changes because of new technology,

Changes of all kinds may have an impact on the software product or the project that creates the product. Changes escalates the cost quickly as the project progresses.

So, support for changes should be embraced into the development process, because it is the heart and soul of software. Agility in software development process facilitates it.



What is Agility?

❑**Agility means effective (rapid and adaptive) response to change by having effective communication among all stakeholders.**

- ▪Effective (rapid and adaptive) response to change
- ▪Effective communication among all stakeholders
- ▪Drawing the customer onto the team
- ▪Organizing a team so that it is in control of the work performed, no escalation of time and cost

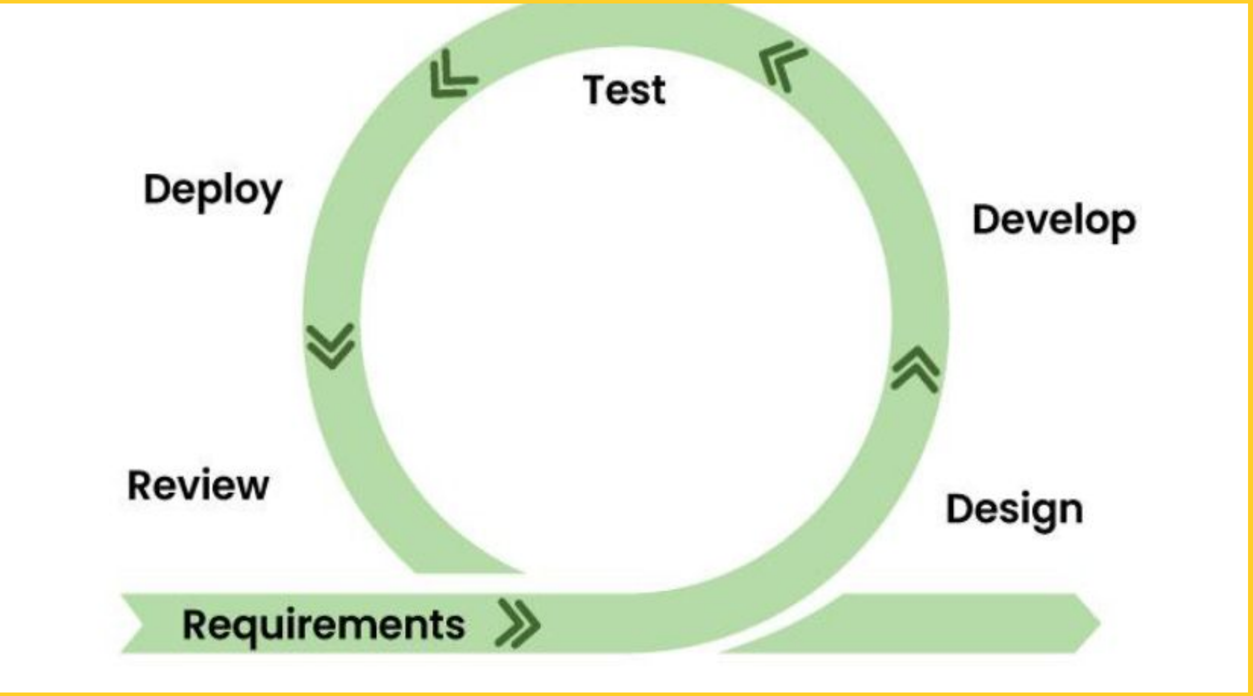
Yielding …

- ▪Rapid, incremental delivery of software

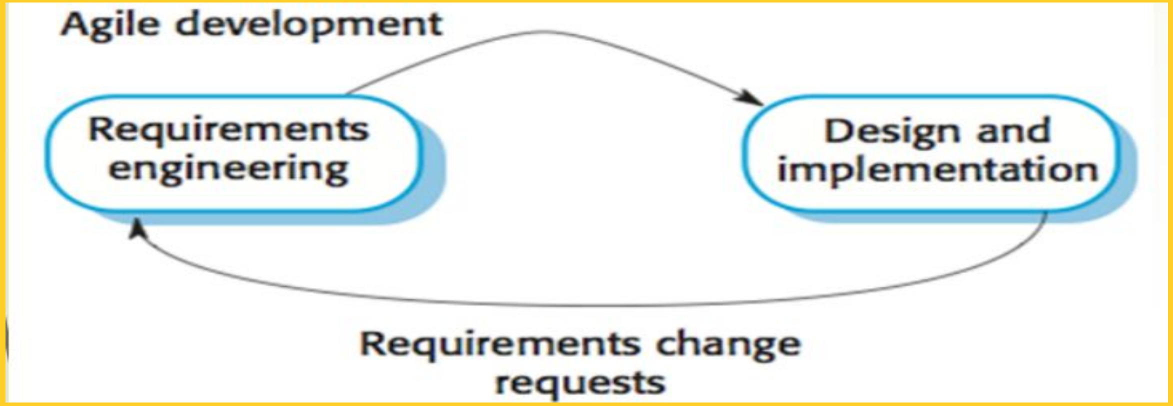
Write a note on Agile Model ?

- The meaning of Agile is swift or versatile.
- **Agile development is a software development methodology that emphasizes iterative and incremental development, frequent delivery of working software, and close collaboration between the development team and stakeholders.**

- "**Agile process model**" refers to a software development approach based on **iterative development.**
- Agile methods **break tasks** into **smaller iterations**, or parts, do not directly involve long term planning.
- The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of **each iteration are clearly defined in advance.**
- **Each iteration** is considered as a short time "frame" in the Agile process model, which typically lasts from **one to four weeks**. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



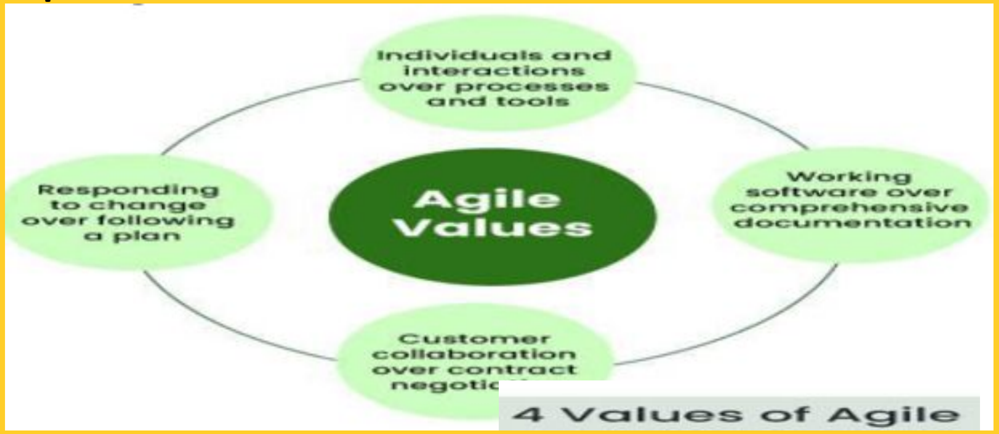
Agile Model



- The incremental parts of a project are Carried out in <u>short-term </u>development cycles.
- The agile approach is based on the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change over processes and tools, comprehensive documentation, contract negotiation, and following a plan.
- Though originally created for software development, the Agile approach is now widely used to execute many different types of projects and run organisations.

Write about the Values and Principles of the Agile Manifesto?

• In 2001, values and principles were introduced by Agile Alliance as part of Agile Manifesto that are considered as the guiding principles for Agile development.


Individuals and interactions over processes and tools
Responding to change over following a plan
Agile Values
Working software over comprehensive documentation
Customer collaboration over contract negotic
4 Values of Agile

Values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The principles of agile methods: There are 12 Principles summarized as:

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.

What are the key practices(characteristics) of Agile development method ?

Agility is achieved through several key practices:

Iterative and incremental development: Agile development is based on iterative and incremental development, which means that the software is developed in small increments or iterations. Each iteration delivers a working piece of software that can be reviewed and tested by stakeholders. This approach enables changes to be made easily and quickly, without incurring significant costs.

Prioritization and flexibility: Agile development emphasizes prioritization and flexibility. This means that requirements are prioritized based on their importance and value to the business, and the development team is able to respond quickly to changing priorities.

Continuous collaboration: Agile development emphasizes continuous collaboration between the development team and stakeholders. This means that stakeholders are involved in the development process, providing feedback and input on the software as it is developed. This enables changes to be made quickly and easily, without significant rework.

Continuous Improvement: Agile development emphasizes continuous improvement. This means that the development team regularly reviews its processes and practices, and makes changes to improve the quality and efficiency of the software development process.

- **By adopting these practices, agile development enables organizations to respond quickly and effectively to changing business needs, without incurring significant costs.**
- **This is because the software is developed in small increments, with each iteration delivering a working piece of software that can be reviewed and tested by stakeholders. Any changes or modifications can be made easily and quickly, without requiring significant rework or delays**

Write the Advantages and Disadvantages of Agile model ?

Advantages of Agile Model

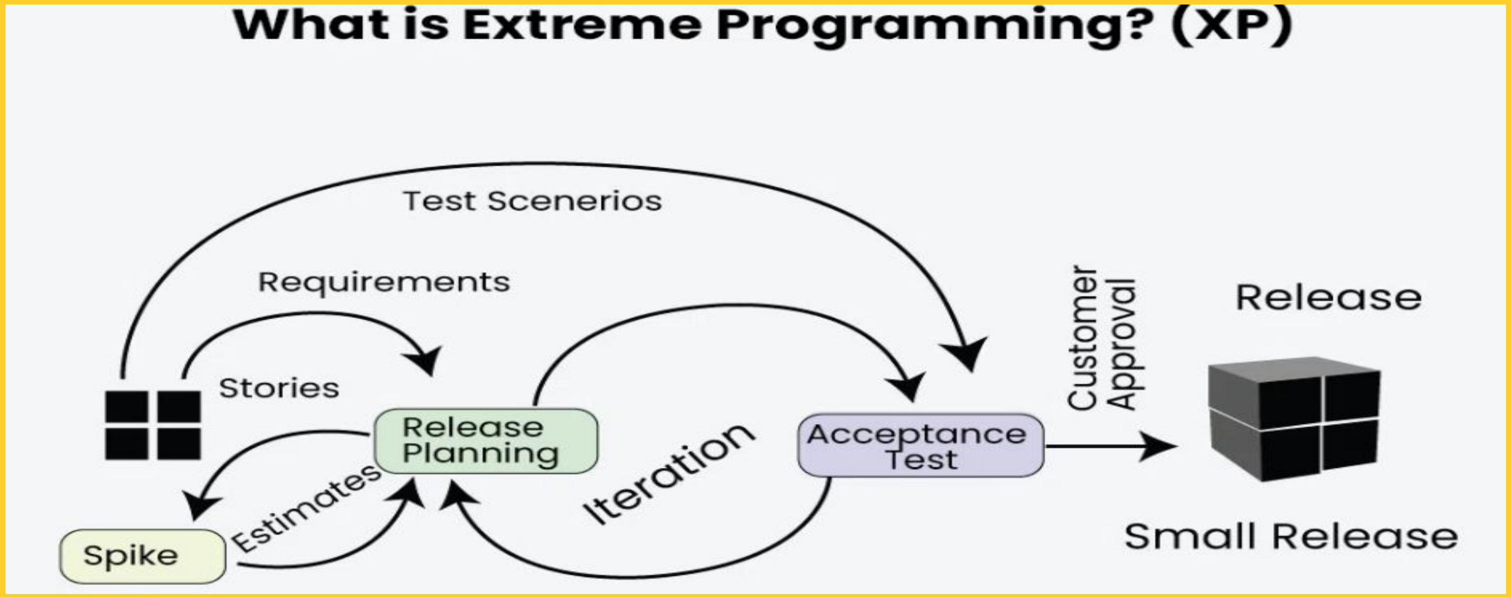
- In this, two programmers work together due to which the code is error free and there are very few mistakes in it.
- In this the software project is completed in a very short time.
- In this the customer representative has an idea of each iteration so that he can easily change the requirement.
- This is a very realistic approach to software development.
- In this, focus is given on teamwork.
- There are very few rules in this and documentation is also negligible.
- There is no need for planning in this.
- It can be managed easily.
- It provides flexibility to developers.

Disadvantages of Agile Model

- It cannot handle complex dependencies.
- Due to lack of formal documentation in this, there is confusion in development.
- It mostly depends on the customer representative, if the customer representative gives any wrong information then the software can become wrong.
- Only experienced programmers can take any decision in this. New programmers cannot take any decision.
- In the beginning of software development, it is not known how much effort and time will be required to create the software.

Write a short note on Extreme Programming ?

- Extreme Programming (XP) is a software development methodology that emphasizes customer satisfaction, teamwork, and the ability to adapt to changing requirements.
- It is an Agile methodology that focuses on delivering high-quality software quickly and efficiently.
- New versions may be built several times per day;
- Increments are delivered to customers every 2 weeks;
- All tests must be run for every build and the build is only accepted if tests run successfully.



The process and roles of extreme programming(Xp process framework Activies)

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

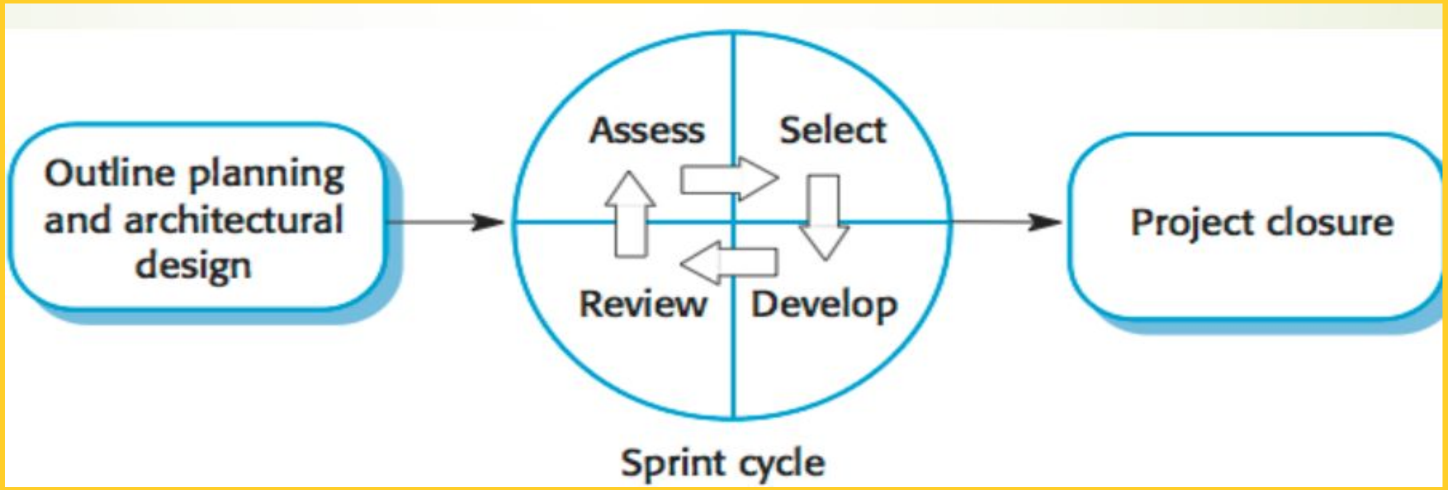
- **Planning,** the first stage, is when the customer meets the development team and presents the requirements in the form of user stories to describe the desired result. The team then estimates the stories and creates a release plan broken down into iterations needed to cover the required functionality part after part. If one or more of the stories can't be estimated, so-called *spikes* can be introduced which means that further research is needed.
- **Designing** is actually a part of the planning process, but can be set apart to emphasize its importance. It's related to one of the main XP values that we'll discuss below -- simplicity. A good design brings logic and structure to the system and allows to avoid unnecessary complexities and redundancies.
- **Coding** is the phase during which the actual code is created by implementing specific XP practices such as coding standards, pair programming, continuous integration, and collective code ownership (the entire list is described below).
- **Testing** is the core of extreme programming. It is the regular activity that involves both unit tests (automated testing to determine if the developed feature works properly) and acceptance tests (customer testing to verify that the overall system is created according to the initial requirements).
- **Listening** is all about constant communication and feedback. The customers and project managers are involved to describe the business logic and value that is expected.

Discuss SCRUM model in detail?

The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.

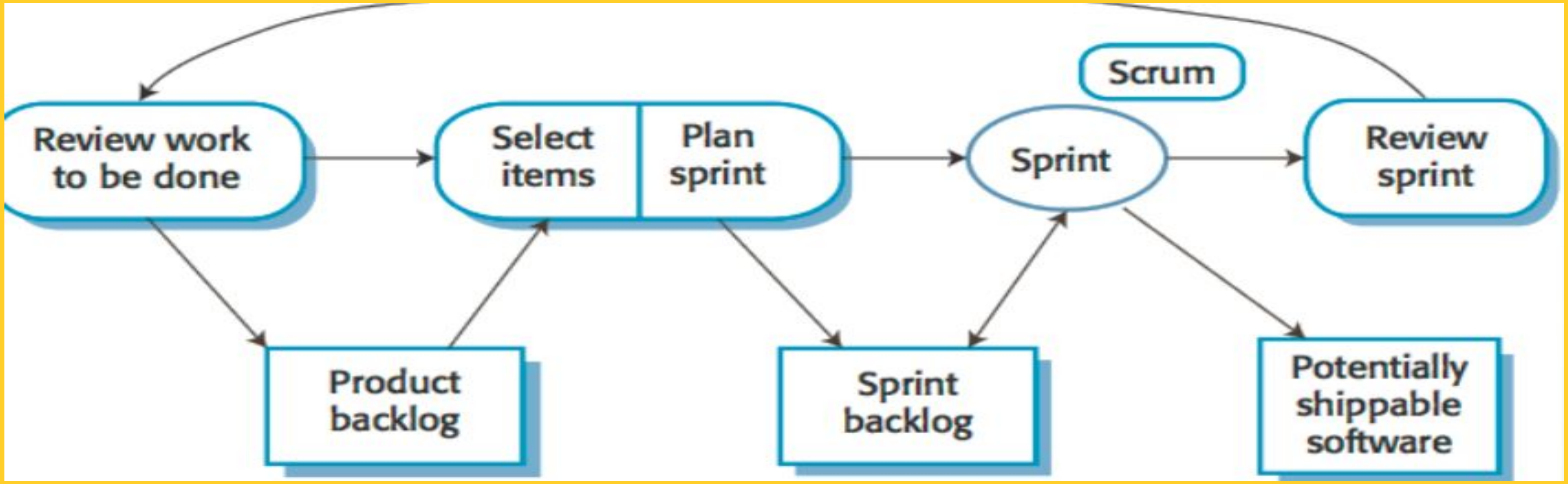
There are three phases in Scrum:

1. The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
2. This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
3. The project closure phase wraps up the project, completes required documentation such as system help menu and user manuals and assesses the lessons learned from the project.



Sprint of Scrum

- Sprints are fixed length, normally 2-4 weeks. The starting point for planning is the product backlog, which is the list of work to be done on the project. The selection phase involves all of the project team who work with the customer (product owner) to select the features and functionality to be developed during the sprint.
- Once these are agreed, the team organize themselves to develop the software. During this stage the team is relatively isolated from the product owner and the organization, with all communications channeled through the ScrumMaster. The role of the ScrumMaster is to arrange daily meetings (daily scrums) tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with the product owner and management outside of the team.
- At the end of the sprint, the work done is reviewed and presented to stakeholders (including the product owner). Velocity (i.e. amount of work) is calculated during the sprint review; it provides an estimate of how much product backlog the team can cover in a single sprint. Understanding the team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring and improving performance. The next sprint cycle then begins.



Advantages:

- ✔ The product is broken down into a set of manageable and understandable chunks. Unstable requirements do not hold up progress.
- ✔ The whole team have visibility of everything and consequently team communication is improved.
- ✔ Customers see on-time delivery of increments and gain feedback on how the product works.
- ✔ Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed

Characteristics of a Software Engineer

There are seven traits that can make an individual software engineer exhibits “super professional” behavior.

Sense of individual responsibility:

- This implies a drive to deliver on his promises to peers, stakeholders, and his management. It implies that he will do what needs to be done, when it needs to be done in an overriding effort to achieve a successful outcome.

Acutely aware of the needs of team members and stakeholders:

- It enables him to work in groups and adapt his behavior when needed.

Brutally honest about design flaws and offers constructive criticism

- Should not distort facts on schedule, feature, performance, or other characteristics of the product, project. Should be trustful.

Resilient under pressure

- Pressure comes in many forms—changes in requirements and priorities, demanding stakeholders or peers, an unrealistic or overbearing manager. He should be able to manage the pressure so that his performance does not suffer.

Heightened sense of fairness

- Gladly share credit with colleagues. Try to avoid conflicts of interest and never acts to sabotage the work of others.

Attention to detail

- carefully considers (or understand) the technical decisions (or works) fully in detail and achieve perfection.

Pragmatic

- software engineering is not a religion in which dogmatic rules must be followed, but rather a discipline that can be adapted based on the circumstances at hand.

Effective Software Team Attributes

A jelled team is a group of people so strongly knit that the whole is greater than the sum of the parts . . . .

Once a team begins to jell, the probability of success goes way up. The team can become unstoppable, a juggernaut for success . . . . They don't need to be managed in the traditional way, and they certainly don't need to be motivated. They've got momentum.

There is no foolproof method for creating a jelled team. But there are attributes that are normally found in effective software teams.

Sense of purpose

- All team members should strongly agree with the goal of the team.

Sense of involvement

- Every member should feel that his skill set and contributions are valued, and contribute fully.

Sense of trust

- Software engineers on the team should trust the skills and competence of their peers and their managers.

Sense of improvement

- Team members should periodically improve its approach to software engineering and looking for ways to improve their work

Diversity of team member skill sets

- The most effective software teams are diverse in skill sets.

Quiz Questions

1) What is agile software development?
2) What is agile manifesto?
3) What are principles of agile development?
4) What is extreme programming?
5) Explain Xp process framework activites in detail? Or what is XP process?
6) What is scrum?
7) What is a sprint in scrum?
8) What are the advantages of agile development?
9) **What are the Characteristics of a Software Engineer?**
10) **What are Effective Software Team Attributes?**