Introduction to Software Engineering and Process Models

- Software Engineering
  - Process Framework
  - Capability Maturity Model (CMM)
  - Advanced Trends in Software Engineering
- Prescriptive Process Models
  - The Waterfall Model
  - Incremental Process Model
  - Evolutionary Models
    - RAD Model
    - Spiral Model
- Agile Process Model
  - Extreme Programming (XP)
  - Scrum
  - Kanban

Software Engineering

❖ What is Software Engineering?

- Software engineering is a systematic and disciplined approach towards the development of the software operation and maintenance.

- The main purpose behind software engineering is to give a framework for building a software with best quality.

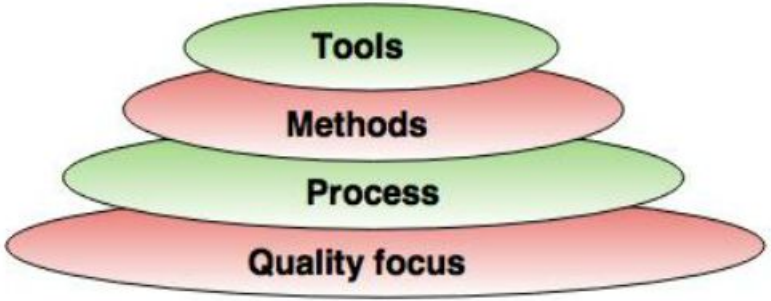
Software Engineering …..Contd.

❖ **Characteristics of a software:**

- Software should achieve a good quality in design and meet all the specifications of the customer.
- Software does not wear out i.e. it does not lose the material.
- Software should be inherently complex.
- Software must be efficient i.e. the ability of the software to use system resources in an effective and efficient manner.
- Software must be integral i.e. it must prevent from unauthorized access to the software or data.

Software Engineering – Layered Technology

- Software engineering is a fully layered technology.
- To develop a software, we need to go from one layer to another.
- All these layers are related to each other and each layer demands the fulfillment of the previous layer.
- The layered technology consists of:



Layered Technology

1. Quality focus:

The characteristics of good quality software are:

- Correctness of the functions required to be performed by the software.
- Maintainability of the software
- Integrity i.e. providing security so that the unauthorized user cannot access information or data.
- Usability i.e. the efforts required to use or operate the software.

2. Process:

- It is the base layer or foundation layer for the software engineering.
- The software process is the key to keep all levels together.
- It defines a framework that includes different activities and tasks.
- i.e. it covers all activities, actions and tasks required to be carried out for software development.

Layered Technology ….Contd.

3. Methods

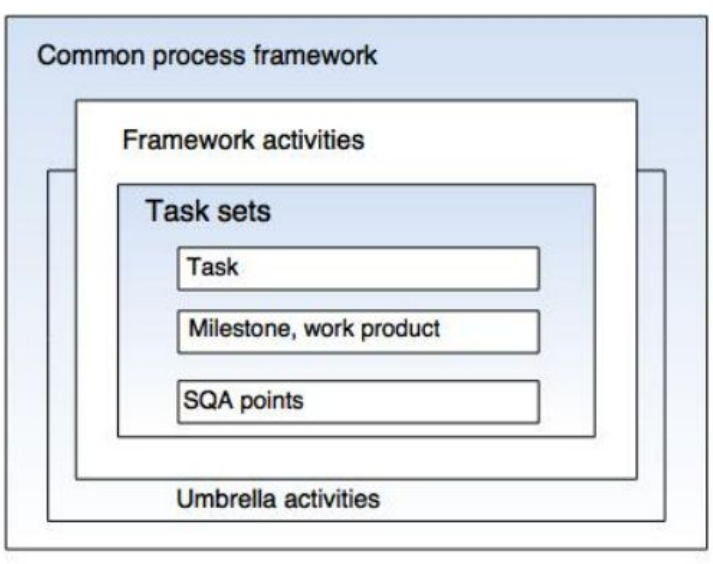
- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.

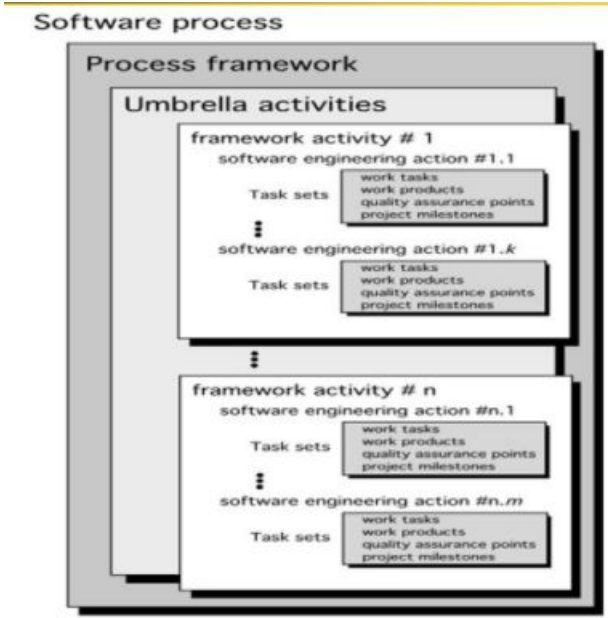
4. Tools

- The software engineering tool is an automated support for the software development.
- The tools are integrated i.e. the information created by one tool can be used by the other tool.
- **Ex:** The Microsoft publisher can be used as a web designing tool.

Software Process Framework

The process of framework defines a small set of activities that are applicable to all types of projects.





The software process framework is a collection of task sets.

Task set consists of a collection of small work tasks, project milestones, work productivity and software quality assurance points.

Umbrella Activities

Typical umbrella activities are:

1. **Software project tracking and control**

✔In this activity, the developing team accesses project plan and compares it with the predefined schedule.

✔If these project plans do not match with the predefined schedule, then the required actions are taken to maintain the schedule.

2. **Risk management**

✔Risk is an event that may or may not occur.

✔If the event occurs, then it causes some unwanted outcome. Hence, proper risk management is required.

3. **Software Quality Assurance (SQA)**

✔SQA is the planned and systematic pattern of activities which are required to give a guarantee of software quality.

✔Ex. During the software development, meetings are conducted at every stage of development to find out the defects and suggest improvements to produce good quality software.

Umbrella Activities ….Contd.

4. **Formal Technical Reviews (FTR)**

✔FTR is a meeting conducted by the technical staff.

✔The motive of the meeting is to detect quality problems and suggest improvements.

✔The technical person focuses on the quality of the software from the customer point of view.

5. **Measurement**

✔Measurement consists of the effort required to measure the software.

✔The software cannot be measured directly. It is measured by direct and indirect measures.

- Direct measures like cost, lines of code, size of software etc.
- Indirect measures such as quality of software which is measured by some other factor. Hence, it is an indirect measure of software.

Umbrella Activities ….Contd.

6. **Software Configuration Management (SCM)**

✔It manages the effect of change throughout the software process.

7. **Reusability management**

✔It defines the criteria for reuse the product.

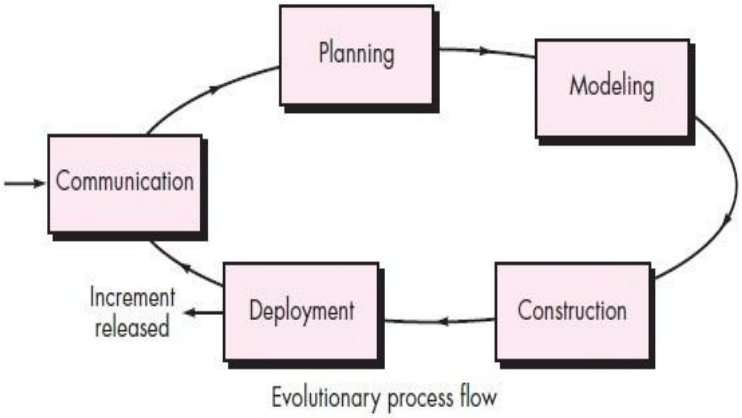
✔The quality of software is good when the components of the software are developed for certain application and are useful for developing other applications.

8. **Work product preparation and production**

✔It consists of the activities that are needed to create the documents, forms, lists, logs and user manuals for developing a software.

Generic Process Framework

There are five generic process framework activities:



1.Communication:

- The software development starts with the communication between customer and developer.
- Communicate with the customer to understand objectives and gather requirements.

Generic Process Framework .....

2. **Planning:**
   - It consists of complete estimation, scheduling for project development and tracking.
   - A “map” is created which defines the work by describing the tasks, risks, resources, work products and work schedule.
3. **Modeling:**
   - It creates a “sketch”, what it looks like architecturally, how the constituent parts fit together and other characteristics.
   - Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
   - The algorithm is the step-by-step solution of the problem and the flow chart shows a complete flow diagram of a program.

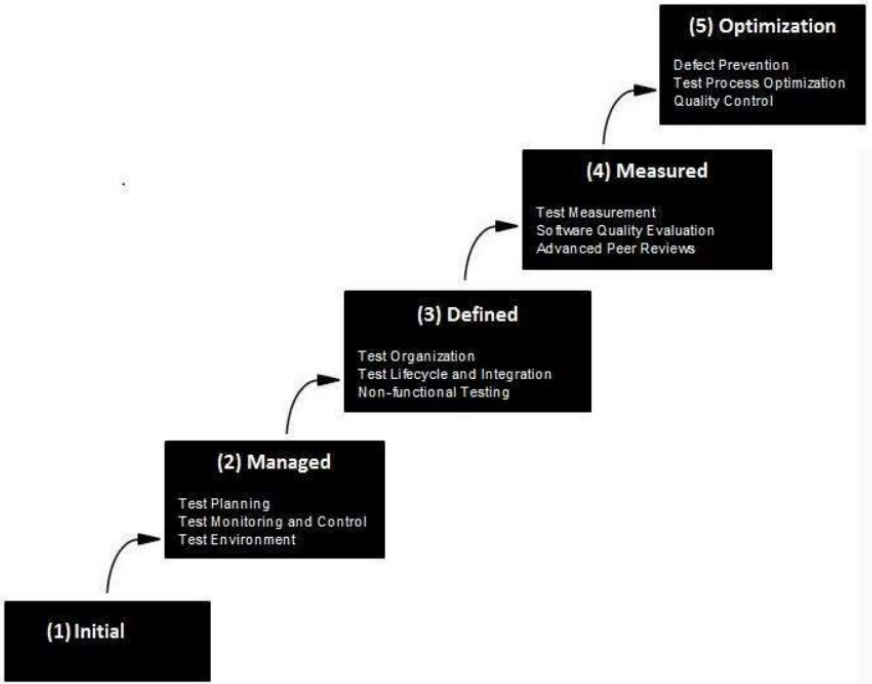
Generic Process Framework ….

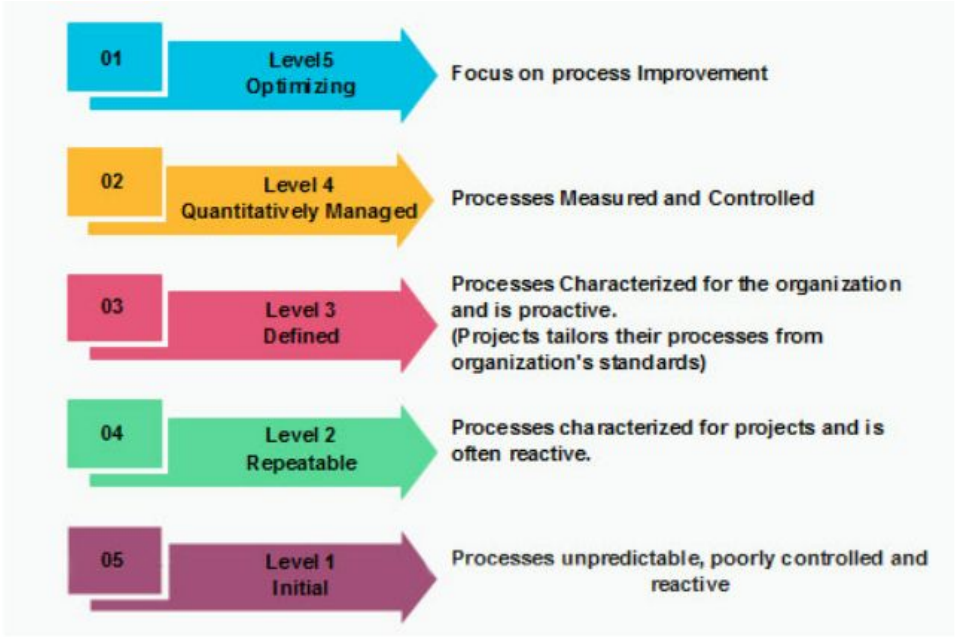
4. **Construction:**
   - Construction consists of code generation and the testing part.
   - Coding part implements the design details using an appropriate programming language.
   - Testing is to check whether the flow of coding is correct or not.
   - Testing also check that the program provides desired output.
5. **Deployment:**
   - Deployment step consists of delivering the product to the customer and taking feedback from them.
   - If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

Capability Maturity Model (CMM)

- The Capability Maturity Model (CMM) is a procedure used to develop and refine an organization's software development process.
- The model defines a five-level evolutionary stage of increasingly organized and consistently more mature processes.
- CMM was developed and is promoted by the Software Engineering Institute (SEI), a research and development center promote by the U.S. Department of Defense (DOD).
- Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.

CMM…..Contd.


(5) Optimization
Defect Prevention
Test Process Optimization
Quality Control
(4) Measured
Test Measurement
Software Quality Evaluation
Advanced Peer Reviews
(3) Defined
Test Organization
Test Lifecycle and Integration
Non-functional Testing
(2) Managed
Test Planning
Test Monitoring and Control
Test Environment
(1) Initial


01
Level 5
Optimizing
Focus on process Improvement
02
Level 4
Quantitatively Managed
Processes Measured and Controlled
03
Level 3
Defined
Processes Characterized for the organization and is proactive.
(Projects tailors their processes from organization's standards)
04
Level 2
Repeatable
Processes characterized for projects and is often reactive.
05
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

CMM…..Contd.

Level 1: Initial

- Ad hoc activities characterize a software development organization at this level.
- Very few or no processes are described and followed.
- Processes followed are adhoc and immature and are not well defined.
- Unstable environment for software development.
- No basis for predicting product quality, time for completion, etc.
- Since software production processes are not limited, different engineers follow their process and as a result, development efforts become chaotic.
- Therefore, it is also called a chaotic level.

CMM…..Contd.

Level 2: Repeatable

- At this level, the fundamental project management practices like tracking cost and schedule are established.
- Size and cost estimation methods, like function point analysis, COCOMO, etc. are used.
- Experience with earlier projects is used for managing new similar natured projects.
- Program management is a key characteristic of a level two organization.
- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.

CMM…..Contd.

Level 3: Defined

- At this level, the methods for both management and development activities are defined and documented.
- There is a common organization-wide understanding of operations, roles, and responsibilities.
  - Peer Reviews- In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
  - Intergroup Coordination- It consists of planned interactions between different development teams to ensure efficient and proper fulfilment of customer needs.
  - Training Programs- It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

CMM…..Contd.

Level 4: Managed

- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.

  - Software Quality Management- It includes the establishment of plans and strategies to develop a quantitative analysis and understanding of the product's quality.
  - Quantitative Management- It focuses on controlling the project performance in a quantitative manner.

CMM…..Contd.

Level 5: Optimizing

- This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.
- Use of new tools, techniques and evaluation of software processes is done to prevent recurrence of known defects.
  - Process Change Management- Its focus is on the continuous improvement of organization's software processes to improve productivity, quality and cycle time for the software product.
  - Technology Change Management- It consists of identification and use of new technologies to improve product quality and decrease the product development time.
  - Defect Prevention- It focuses on identification of causes of defects and to prevent them from recurring in future projects by improving project defined process.

CMM…..Contd.

- Except for CMM level 1, each maturity level is featured by several Key Process Areas (KPAs) that contains the areas an organization should focus on improving its software process to the next level.
- The focus of each level and the corresponding key process areas can be summarized as:

CMM Level	Focus	Key Process Areas
1. Initial	Competent People	NO KPA'S
2. Repeatable	Project Management	Software Project Planning software Configuration Management
3. Defined	Definition of Processes	Process definition Training Program Peer reviews
4. Managed	Product and Process quality	Quantitiative Process Metrics Software Quality Management
5. Optimizing	Continuous Process improvement	Defect Prevention Process change management Technology change management

Advanced Trends in Software Engineering

- **Internet of Things (IoT):**

  - IoT has come a long way, from ordinary mobile devices to the vehicle, everything is attached with sensors that enable an object to be monitored or data gathered.
  - It is only possible to get the best out of IoT by combining it with other revolutionary innovations such as AI and Big Data.
  - The on-going COVID pandemic has accelerated the adoption of tech-driven healthcare transformation.
  - The rise of smart city solutions is rolling out IoT projects recently.

Advanced Trends in Software Engineering

- **Low-Code Development:**

  - You do not need to code an application line by line for low-code development tools, you sketch it like a flowchart.
  - This makes it fast and intuitive to create powerful new applications without manual coding.
  - Companies have started to realize that using the low-code software development approach is much more fruitful in the digital transformation and to operate their activities smoothly.

Advanced Trends in Software Engineering

- **Human Augmentation (HA):**

  - The implantation of technologies into the human body and the human brain will begin to advance.
  - The physical and cognitive improvements in the human experience are actually the goal.
  - An example of human augmentation is powered exoskeletons.
    - It can be applied to the roles of firefighters and workers who constantly move heavy loads as it gives back support, sends a signal to motors, and assists humans in physical movements.

Advanced Trends in Software Engineering

- **Augmented Reality (AR):**

  - It is empowering the digital world and is applicable in almost all architecture industries like retail, navigation, and manufacturing.
  - An example of customer Augmented Reality in the eCommerce business is in the form of ads.
    - The more people are using social media like Facebook and Instagram, marketers are embracing AR as the ad format.
    - The beauty and fashion industry is using AR filters for a complete look, with virtual makeup bringing it to a whole new dimension.

Advanced Trends in Software Engineering

- **Virtual Reality (VR):**

  - It is widely used in gaming and video industry.
  - The training, and education sector has also likely to explore VR soon as schools have started making investments to bring VR into the classroom.
  - With the help of software development, employees and management can implement VR in the workforce and training sessions to improve customer service and user experience.

Advanced Trends in Software Engineering

- **Mixed Reality (MR):**

  - Smartphones gave us access, but mixed reality will give us a visualization.
  - If you are an entrepreneur with a complex operational business, you can think of AR smart glasses with VR headsets.
    - Using a mixed reality interface will direct your employees through the most complicated control operations of enterprises.
  - It will be also useful for native app development of restaurants, real-estate, etc.

Advanced Trends in Software Engineering

- **Cloud-Based Solution:**

  - In recent years, there is a growing need for business availability, data recovery, and high accessibility.
  - Many companies and entities, even governmental institutions felt the need to shift on to cloud technology.
  - With newer technology coming in like “distributed cloud”, the need for this technology is going to rise even more.
  - This would make it easier for them to more easily access vast amounts of data.

Advanced Trends in Software Engineering

- **Big Data:**

  - With big data, companies and startups around the world have begun reaping benefits like making use of data.
  - Another advantage of big data is Daas (data-as-a-service).
    - We know that the cloud allows businesses to access the infrastructure they need when they need it.
    - Similarly, Daas will allow businesses to access the exact data they want and when they want, eliminating redundancy.

Advanced Trends in Software Engineering

- **Artificial Intelligence (AI):**

  - Machines can think intelligently like humans and perform tasks that were initially restricted to humans.
  - There is a rising trend of chatbots in the development of apps and websites using powerful programming languages.
    - AI chatbots have replaced human assistance and customer service.
  - The use of AI and machine learning in native app development has begun to enhance customer experience and app usability.
  - They have reduced human involvement in almost everything and made it less prone to errors.

Advanced Trends in Software Engineering

- **Blockchain:**

  - Blockchain became the talk of the town with the introduction of digital currencies like Blockchain Bitcoin and Ethereum.
  - Blockchain is intensively used in the finance sector, that it is now also being applied in banking, finance, media, publishing, and healthcare software development.
  - Blockchain technology helps in the secured and simplified recording of transactions in a decentralized ledger.
  - This makes it strategically important for businesses in all industrial verticals.

Prescriptive Process Models

The name 'prescriptive' is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.

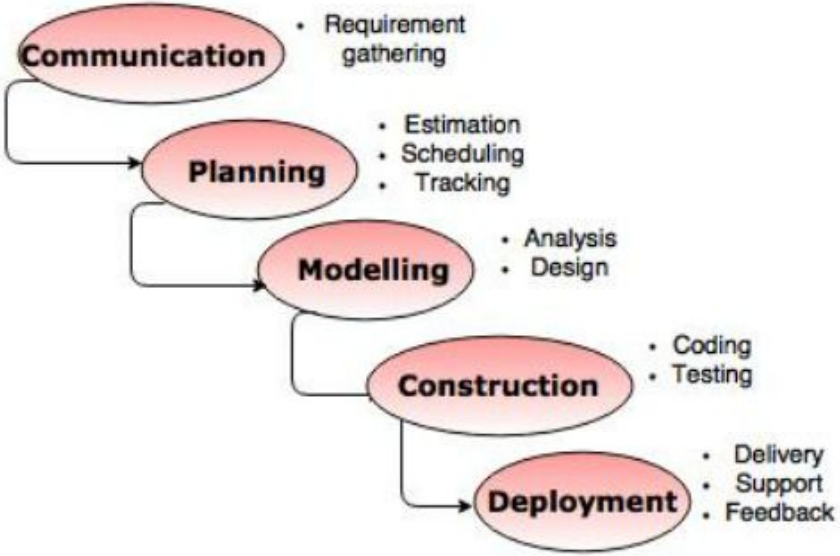
There are three types of prescriptive process models:

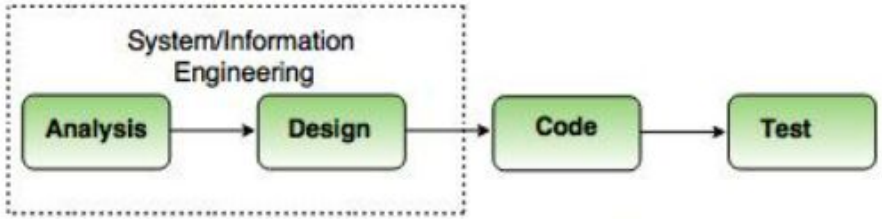
1. The Waterfall Model
2. Incremental Process model
3. Rapid Application Development (RAD) model

Waterfall Model

- The Waterfall model was the first Process Model to be introduced.
- It is also referred to as a linear-sequential life cycle model.
- It is very simple to understand and use.
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- Typically, the outcome of one phase acts as the input for the next phase sequentially.
- In this model, feedback is taken after each phase to ensure that the project is on the right path.
- Testing part starts only after the development is complete
- The description of the phases of the waterfall model is same as that of the process model.
- It does have an alternative design.

Waterfall Model …..Contd.





Phases in Waterfall Model

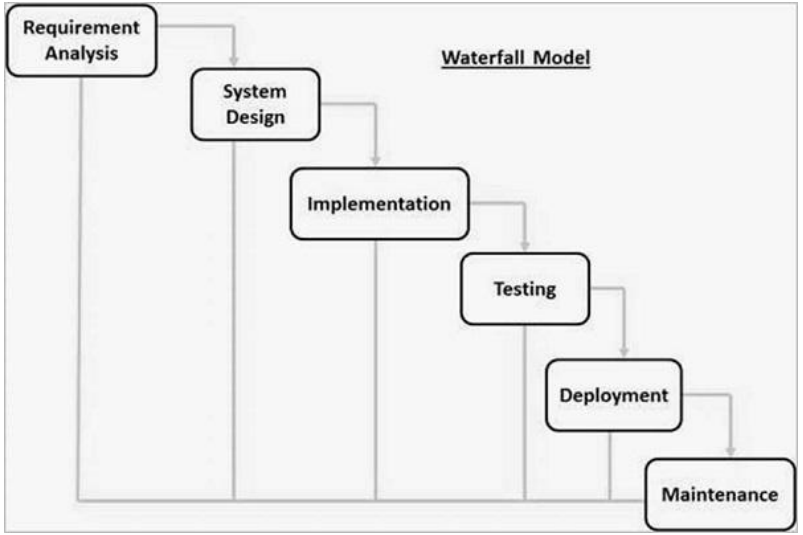
- **Requirement Gathering and analysis −**
  - All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
  - Both the customer and the software developer work together so as to document all the functions, performance, and interfacing requirement of the software.
  - It describes the "what" of the system to be produced and not "how."
- **System Design −**
  - The requirement specifications from first phase are studied in this phase and the system design is prepared.
  - This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Phases in Waterfall Model ….Contd.

- **Implementation −**
  - With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase.
  - Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing −**
  - All the units developed in the implementation phase are integrated into a system after testing of each unit.
  - Post integration the entire system is tested for any faults and failures.
- **Deployment of system −**
  - Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Phases in Waterfall Model ….Contd.

- **Maintenance** –
  - There are some issues which come up in the client environment. To fix those issues, patches are released.
  - Also to enhance the product some better versions are released.
  - Maintenance is done to deliver these changes in the customer environment.



Advantages of Waterfall Model

- The waterfall model is simple and easy to understand, implement, and use.
- All the requirements are known at the beginning of the project, hence it is easy to manage.
- It avoids overlapping of phases because each phase is completed at once.
- This model works for small projects because the requirements are understood very well.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.
- This model is preferred for those projects where the quality is more important as compared to the cost of the project.

Disadvantages of Waterfall Model

- No working software is produced until late during the life cycle.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- The problems with this model are uncovered, until the software testing.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a “big-bang” at the very end, which does not allow identifying any technological or business bottleneck or challenges early.

Applications of Waterfall Model

- Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors.

- Some situations where the use of Waterfall model is most appropriate are:-
  - ✔ Requirements are very well documented, clear and fixed.
  - ✔ Product definition is stable.
  - ✔ Technology is understood and is not dynamic.
  - ✔ There are no ambiguous requirements.
  - ✔ Ample resources with required expertise are available to support the product.
  - ✔ The project is short.

Incremental Process Model

- Incremental model combines the elements of waterfall model and they are applied in an iterative manner.
- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for any suggested modifications.
- The next increment implements on the customer's suggestions and add additional requirements in the previous increment.
- This process is repeated until the product is finished.
- Ex. the word-processing software is developed using the incremental model.

Incremental Process Model ….

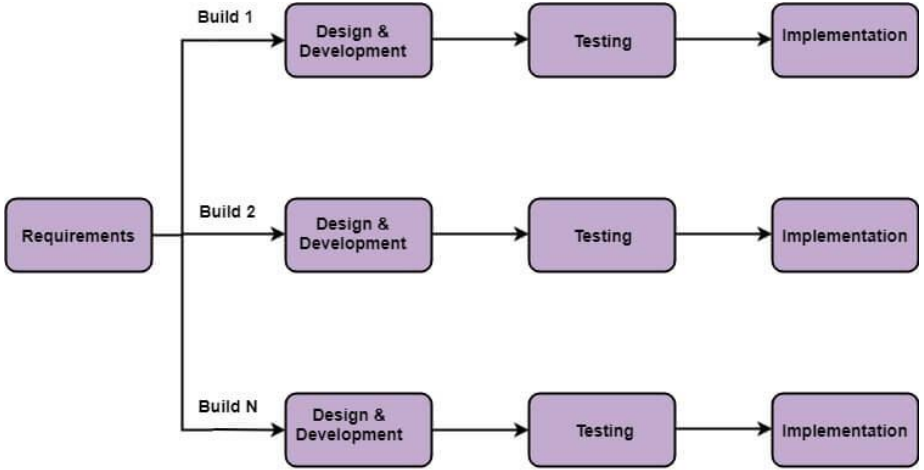


Fig: Incremental Model

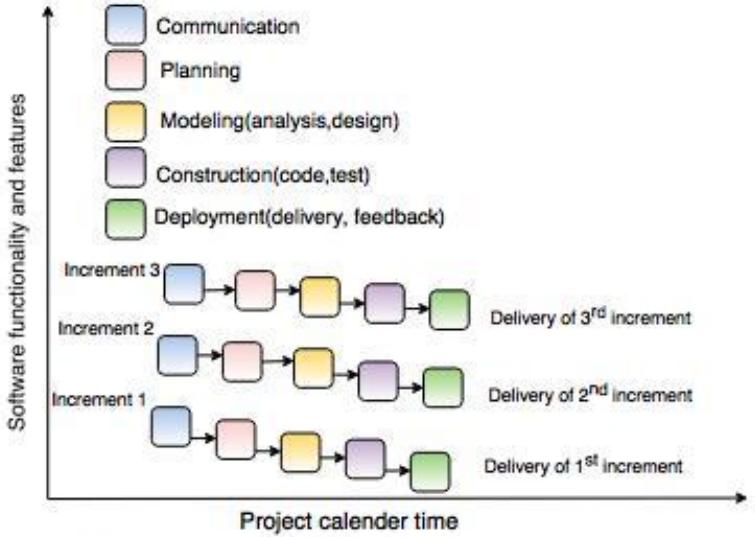


Fig. - Incremental Process Model

Phases in Incremental Process Model

1. **Requirement analysis:**
   - In the first phase of the incremental model, the product analysis expertise identifies the requirements.
   - The system functional requirements are understood by the requirement analysis team.
   - To develop the software under the incremental model, this phase performs a crucial role.
2. **Design and Development:**
   - In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success.
   - When software develops new practicality, the incremental model uses style and development phase.

Phases in Incremental Model ….

3. **Testing:**
   - In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality.
   - In the testing phase, the various methods are used to test the behavior of each task.
4. **Implementation:**
   - Implementation phase enables the coding phase of the development system.
   - It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase.
   - After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Advantages of Incremental Model

- This model is flexible because the cost of development is low and initial product delivery is faster.
- Errors are easy to be recognized.
- It is easier to test and debug during the smaller iteration.
- Simple to manage risk because it handled during its iteration.
- The working software generates quickly and early during the software life cycle.
- The customers can respond to its functionalities after every increment.
- The Client gets important functionality early.
- Parallel development can be planned.
- Progress can be measured.
- It supports changing requirements.
- Better suited for large and mission-critical projects.

Disadvantages of Incremental Model

- The cost of the final product may cross the cost estimated initially.
- This model requires a very clear and complete planning.
- The planning of design is required before the whole system is broken into small increments.
- The demands of customer for the additional functionalities after every increment causes problem during the system architecture.
- More as well as highly skilled resources are required for risk analysis.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Projects progress is highly dependent upon the risk analysis phase.

Applications of Incremental Model

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.
- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

Evolutionary Models

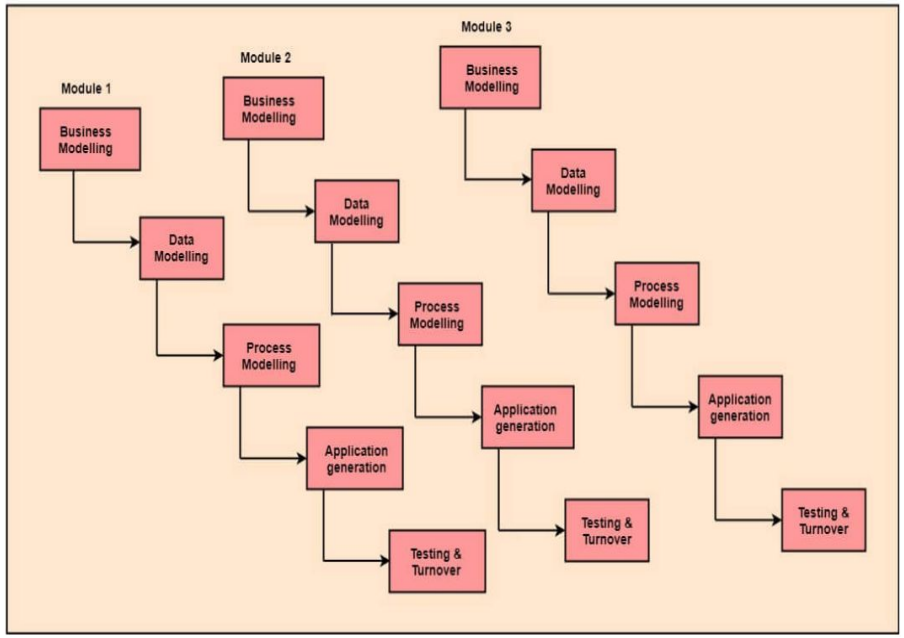
- Evolutionary model is a combination of Iterative and Incremental model of software development life cycle.
- The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle.
- Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process.
- Therefore, the software product evolves with time.

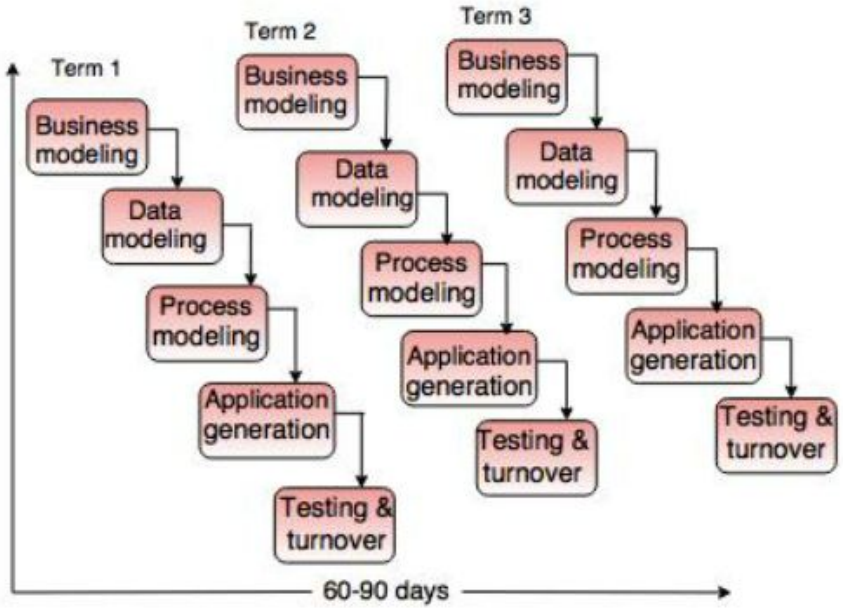
- Types of Evolutionary Models:
  1) Rapid Application Development (RAD) Model
  2) Spiral Model

Rapid Application Development Model

- Requirements of the complete system are clearly defined and understood.
- Using the RAD model, software product is developed in a short period of time.
- The initial activity starts with the communication between customer and developer.
- Planning depends upon the initial requirements and then the requirements are divided into groups.
- Planning is more important to work together on different modules.
- The products can be developed faster and of higher quality through:
  - Gathering requirements using workshops or focus groups
  - Prototyping and early, reiterative user testing of designs
  - The re-use of software components
  - A rigidly paced schedule that refers design improvements to the next product version
  - Less formality in reviews and other team communication.

RAD Model


Module 1
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 2
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 3
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover


Term 1
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Term 2
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
Term 3
Business modeling
Data modeling
Process modeling
Application generation
Testing & turnover
60-90 days

Phases in RAD Model

1.Business Modelling:

- The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.
- A complete business analysis should be performed to get the essential business information.

2. Data Modelling:

- The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business.
- The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

Phases in RAD Model ….Contd.

3. Process Modelling:

- The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function.
- Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation:

- Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

Phases in RAD Model ….Contd.

5. Testing and Turnover:

- Many of the programming components have already been tested since RAD emphasis reuse.
- This reduces the overall testing time.
- But the new part must be tested, and all interfaces must be fully exercised.

- The critical feature of this model is the use of powerful development tools and techniques.
- Another striking feature of this model is a short time span i.e the time frame for delivery(time-box) is generally 60-90 days.

Advantages of RAD Model

- Use of reusable components helps to reduce the cycle time of the project.
- Feedback from the customer is available at initial stages.
- Reduced costs as fewer developers are required.
- Use of powerful development tools results in better quality products in comparatively shorter time spans.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.
- Each phase in RAD brings highest priority functionality to the customer.
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD Model

- The use of powerful and efficient tools requires highly skilled professionals.
- The absence of reusable components can lead to failure of the project.
- The team leader must work closely with the developers and customers to close the project in time.
- The systems which cannot be modularized suitably cannot use this model (i.e. Suitable for systems that are component based and scalable).
- Customer involvement is required throughout the life cycle.
- It is not meant for small scale projects as for such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.
- All application is not compatible with RAD.
- On the high technical risk, it is not suitable.
- Dependency on technically strong team members for identifying business requirements.
- High dependency on Modelling skills.
- Management complexity is more.
- Suitable for project requiring shorter development times.

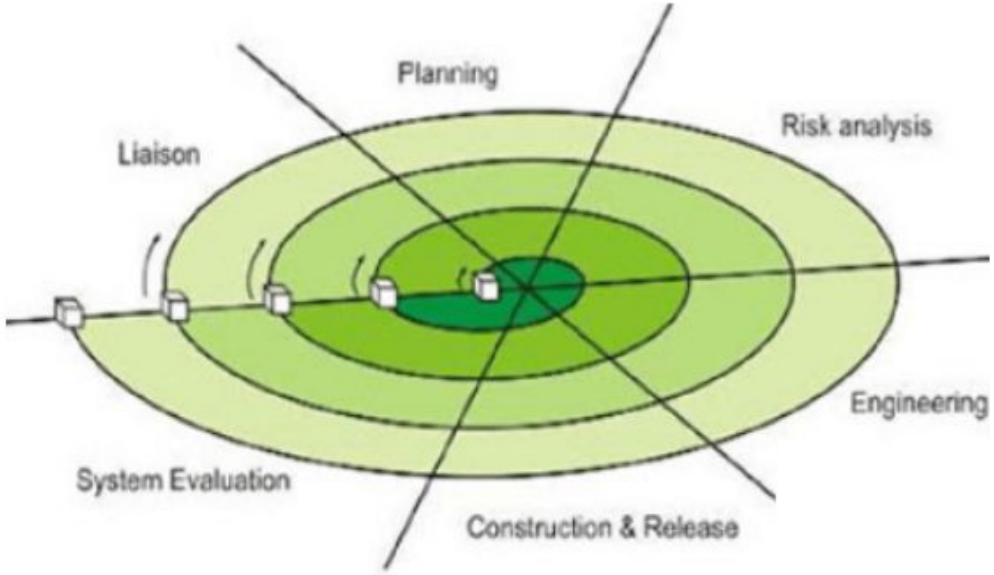
Applications of RAD Model

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- It should be used only if the budget allows the use of automatic code generating tools.
- It should be used if there is a high availability of designers for Modelling.
- This model should be chosen only if domain experts are available with relevant business knowledge.

Spiral Model

- The spiral model in software engineering was first mentioned by Barry Boehm.
- It is a risk-driven software development process model.
- It is a combination of waterfall model and iterative model.
- Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.
- It allows incremental releases of the product or incremental refinement through each iteration around the spiral.
- Each phase of spiral model in software engineering begins with a design goal and ends with the client reviewing the progress.
- The development process in Spiral model, starts with a small set of requirement and goes through each development phase for those set of requirements.
- The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.

Spiral Model ….Contd.



Phases in Spiral Model

1. **Communication/ Liaison:**
   - The members involved communicate/ interact with the customer to gather the requirements.
   - They understand the specifications for the system.

2. **Planning:**
   - It includes estimating the cost, schedule and resources for the iteration.
   - It also involves understanding the system requirements for continuous communication between the system analyst and the customer.

3. **Risk Analysis**
   - It is an important phase of the Spiral model.
   - It involves Risk analyst.
   - Identification of potential risk is done while risk mitigation strategy is planned and finalized.

Phases in Spiral Model ….Contd.

4. **Engineering**
   - It includes designing/ modeling of the system.
   - The design phase starts with the conceptual design in the baseline spiral.
   - It involves architectural design, logical design of modules, physical product design and the final design in subsequent spirals.

5. **Construction and Release**
   - In this phase, the programs are developed and integrated to form a software or a prototype.
   - In the early cycles, the product of this phase would be a prototype and in subsequent cycles, the product of this stage is a developed software.
   - It also includes testing and deploying software at the customer site.

6. **Evaluation/ Feedback**
   - It involves evaluation of the software by the customer.
   - It also includes identifying and monitoring risks such as schedule slippage and cost overrun.

Advantages of Spiral Model

- This model focuses on reducing the risks involved in the project.
- It produces a prototype which helps in evaluating the cost and time to develop the software.
- This model is suitable for long term project where the customer requirements may change over the period and it can be implemented in the product.
- In the spiral model, evaluation is performed at the end of every cycle which helps in tracking the progress of the project.
- In the spiral model, the risk is analyzed at the starting of each cycle which helps in planning the measures to reduce the risk in subsequent stages.
- There is always space for customer feedback.
- It is recommended to use the Spiral Model in large and complex projects.
- Customer can see the development of the product at the early phase of the software development and thus, they get habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model

- The spiral model is expensive and, therefore, is not suitable for small projects.
- 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.
- As the number of phases is unknown at the start of the project, so time estimation is very difficult.
- The spiral model is more complex than other SDLC options. For it to operate efficiently, protocols must be followed closely. Furthermore, there is increased documentation since the model involves intermediate phases.

Applications of Spiral Model

- It should be used in projects where high-level risks are involved.
- It is used where developing the prototype of a product is possible.
- It can be used when the client is not able to specify all the requirements explicitly and the requirement of the client may change in between the project.
- It is suitable for long term projects.
- It can be used when the requirements of the project are complex and there is a need for frequent evaluation.
- It must be used when frequent customer feedback is required.

Waterfall Vs Spiral Model

Waterfall Model	Spiral Model
Waterfall model works in sequential manner.	Spiral model works in evolutionary manner.
In waterfall model errors or risks are identified and rectified after the completion of stages.	In spiral model errors or risks are identified and rectified earlier (at every stage/ phase).
Waterfall model is adopted by customers.	Spiral model is adopted by developers.
Waterfall model is applicable for small project.	Spiral model is used for large project.
In waterfall model requirements and early stage planning is necessary.	In Spiral model requirements and early stage planning is not necessary, until 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.	Spiral model is very expensive.

Waterfall Vs Incremental Vs RAD Vs Spiral Model

Parameter	Waterfall	Incremental	RAD	Spiral
Planning in early stage	Yes	Yes	Yes	No
Going back to previous phase	No	Yes	Yes	Yes
Handle Large-Project	Not Appropriate	Not Appropriate	Appropriate	Not Appropriate
Detailed Documentation	Necessary	Yes but not much	Yes	Limited
Cost	Low	Low	Expensive	Low
Requirement Specifications	Beginning	Beginning	Beginning	Time boxed release
Flexibility to change	Difficult	Easy	Easy	Easy
User Involvement	Only at beginning	Intermediate	High	Intermediate

Waterfall Vs Incremental Vs RAD Vs Spiral Model

Parameter	Waterfall	Incremental	RAD	Spiral
Risk Involvement	High	Low	Medium to high risk	Low
Framework Type	Linear	Linear + Iterative	Linear + Iterative	Linear
Testing	After completion of coding phase	After every iteration	At the end of the engineering phase	After completion of coding at every spiral
Overlapping Phases	No	Yes (As parallel development is there)	No	Yes
Maintenance	Least Maintainable	Maintainable	Yes	Easily Maintainable
Re-usability	Least possible	To some extent	To some extent	Yes
Time-Frame	Very Long	Long	Long	Short
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration	At the end of the life cycle

Waterfall Vs Incremental Vs RAD Vs Spiral Model

Parameter	Waterfall	Incremental	RAD	Spiral
Objective	High Assurance	High Assurance	Rapid Development	High Assurance
Team size	Large Team	Not Large Team	Large Team	Small Team
Customer control over administrator	Very Low	Yes	Yes	Yes

Agile Process Model

- Agile process model is based on iterative development.
- It is versatile and effective (rapid and adaptive) response to change.
- An agile process
  - ✔ Is driven by customer descriptions of what is required (scenarios).
  - ✔ Recognizes that plans are short-lived
  - ✔ Develops software iteratively with a heavy emphasis on construction activities.
  - ✔ Delivers multiple ‘software increments’.
  - ✔ Adapts as change occurs
- Project scope and requirements are laid down at the beginning of the development process.
- Entire project is divided into smaller parts. This helps to minimize the project risk and to reduce the overall project delivery time requirements.
- 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", which typically lasts from one to four weeks.
- Each iteration involves a team working through a full software development life cycle before a working product is demonstrated to the client.

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.

Agility Principles ….Contd.

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.

Key Components for Agile Team

1. **Communication**:
   - One of the most fundamental requirements of a high-performing agile team is effective communication between team members as well as between the teams and management.
   - When teams take the time to reflect on how they can communicate more effectively and adjust accordingly, they can be more successful.
2. **Collaboration**:
   - Agile teams are required to work in short iterations, or sprints, that require a higher level of collaboration with other teams and business partners.
   - Agile teams are those that realize the entire team is solving a program, and not the individuals.
3. **Self-Organizing and Self-Sufficient**:
   - Each team member must be capable of determining their next tasks, and teams must be able to organize meetings between themselves, or with external collaborators or stakeholders, without much guidance from higher-level executives.
   - A self-sufficient agile team requires specific tools to support the way they work and may benefit from using project management tools to plan their springs and manage

Key Components for Agile Team

tasks independently without the need for a traditional project manager.

4. **Metrics-Driven:**
   - Agile teams need to be result-driven and there are plenty of analytic tools to measure the success of a project or final product.
   - However, it can be difficult to measure just how healthy an agile team is in order to uncover areas for improvement.
   - Tracking and sharing sound agile metrics about the team itself can help everyone track their progress as well as inspect and adapt to how they work together.
5. **Mutual respect:**
   - Agile team only works when it consists of individuals that have both shared and complementary skill sets.
   - This allows the team to help each other on specific features or entire projects without resulting in external resources.
   - Workloads are better balanced, and team members tend to appreciate and respect each other's contributions to the overall project much more.

Extreme Programming (XP)

- e**X**treme **P**rogramming (XP) is one of the Agile software development methodologies.
- XP has a set of rules and practices which are used to guide the team behavior.
- It was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements.
- XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop a software.
- It is used to improve software quality and responsive to customer requirements.
- The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
- It uses the concept of object-oriented programming.
- A developer focuses on the framework activities like planning, design, coding and testing.

Values for Extreme Programming (XP)

I. **Communication:**
- Building software development process needs communication between the developer and the customer.
- It is important for requirement gathering and discussing the concept.

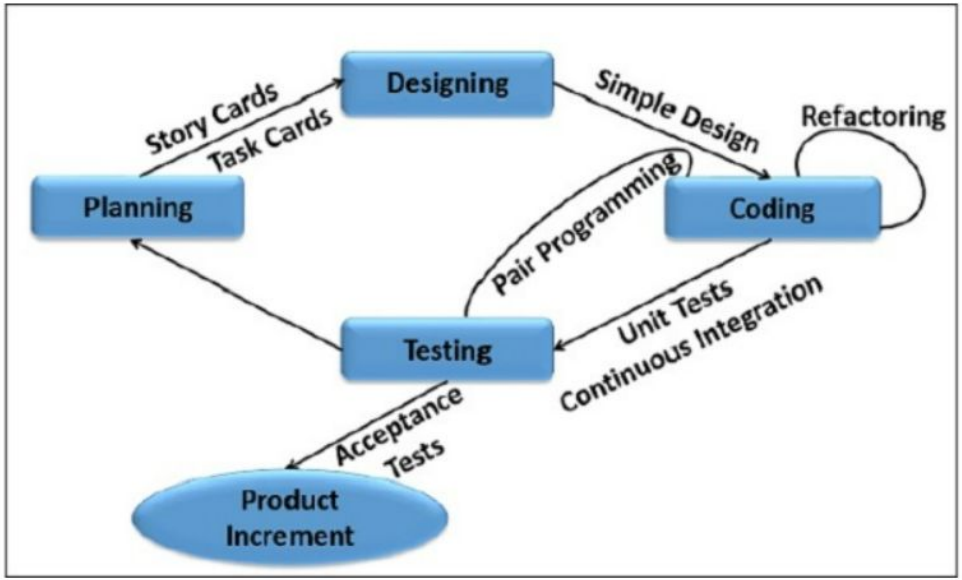
II. **Simplicity:**
- Simple design is easy to implement in code.

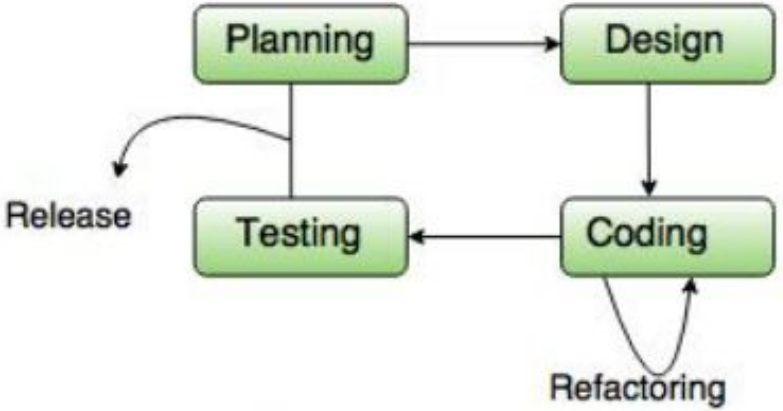
III. **Feedback:**
- Feedback guides the development process in the right direction.

IV. **Courage**
- In every development process there will always be a pressure situation.
- The courage or the discipline to deal with it surely makes the task easy**.**

V. **Respect:**
- Agile process should inculcate the habit to respect all team members, other stake holders and customer.

Phases in Extreme Programming (XP)





Phases in Extreme Programming (XP)

1. **Planning:**
   - The first phase of Extreme Programming life cycle is planning, where customers or users meet with the development team to create ‘user stories’ or requirements.
   - The development team converts user stories into iterations that cover a small part of the functionality or features required.
   - A combination of iterations provides the customer with the final fully functional product.
   - The programming team prepares the plan, time, and costs of carrying out the iterations, and individual developers sign up for iterations.
   - One planning approach is the critical path method, grouping iterations essential for project progress in a linear fashion, and arranging for completion of other iterations parallel to the critical path.

Phases in XP …. Contd.

2. Designing:

- The XP design follows the 'keep it simple' principle.
- A simple design always prefers the more difficult representation.
- An iteration of XP programming starts with designing.
- The guiding principles of this stage are:
  - Thrust on simplicity by expressing a thing only once and not adding functionality in anticipation.
  - Using systems metaphor or standards on names, class names and methods, and agreeing on uniform styles and formats to ensure compatibility among the work of different team members.
  - Using Software Class Responsibilities and Collaboration (CRC) Cards that allow for a departure from the traditional procedural mindset and make possible object oriented technology.
  - Such cards allow all members of the project team to contribute ideas, and collate the best ideas into the design
  - Creating spike solutions or simple programs that explore potential solutions for a specific problem, ignoring all other concerns, to mitigate risk.

Phases in XP …. Contd.

3. **Coding:**
   - Coding constitutes the most important phase in the Extreme Programming life cycle.
   - XP programming gives priority to the actual coding over all other tasks such as documentation to ensure that the customer receives something substantial in value at the end of the day.
   - Standards related to coding include:
     - Developing the code based on the agreed metaphors and standards, and adopting a policy of collective code ownership.
     - Pair programming or developing code by two programmers working together on a single machine, aimed at producing higher quality code at the same or less cost.
     - Strict adherence to 40-hour work weeks with no overtime. This ensures the developers work in the peak of their mental and physical faculties.
     - Frequent integration of the code to the dedicated repository, with only one pair integrating at a time to prevent conflicts, and optimization at the end.

Phases in XP …. Contd.

4. **Testing:**
   - Extreme programming integrates testing with the development phase rather than at the end of the development phase.
   - All codes have unit tests to eliminate bugs, and the code passes all such unit tests before release.
   - Validation testing of the system occurs on a daily basis.
   - It gives the XP team a regular indication of the progress.
   - Another key test is customer acceptance test, based on the customer specifications. 'XP acceptance tests' are known as the customer test.
   - Acceptance test run at the completion of the coding, and the developers provide the customer with the results of the acceptance tests along with demonstrations.

Advantages of XP

- It creates a close contact with the customer.
- No unnecessary programming work is required.
- It generates a stable software through continuous testing.
- It avoids error through pair programming.
- No overtime, teams work at their own pace.
- Changes can be made at a short notice.
- Code is clear and comprehensible at all times.
- It can be used where environments that are highly uncertain.

Disadvantages of XP

- Customer must participate in the process.
- The customer is at a distance or unavailable.
- Relatively large time is invested.
- Cost is relatively high.
- Requires to practice self-discipline.
- Requires version management.
- Additional work.
- The environments are large and complex.

Applications of XP

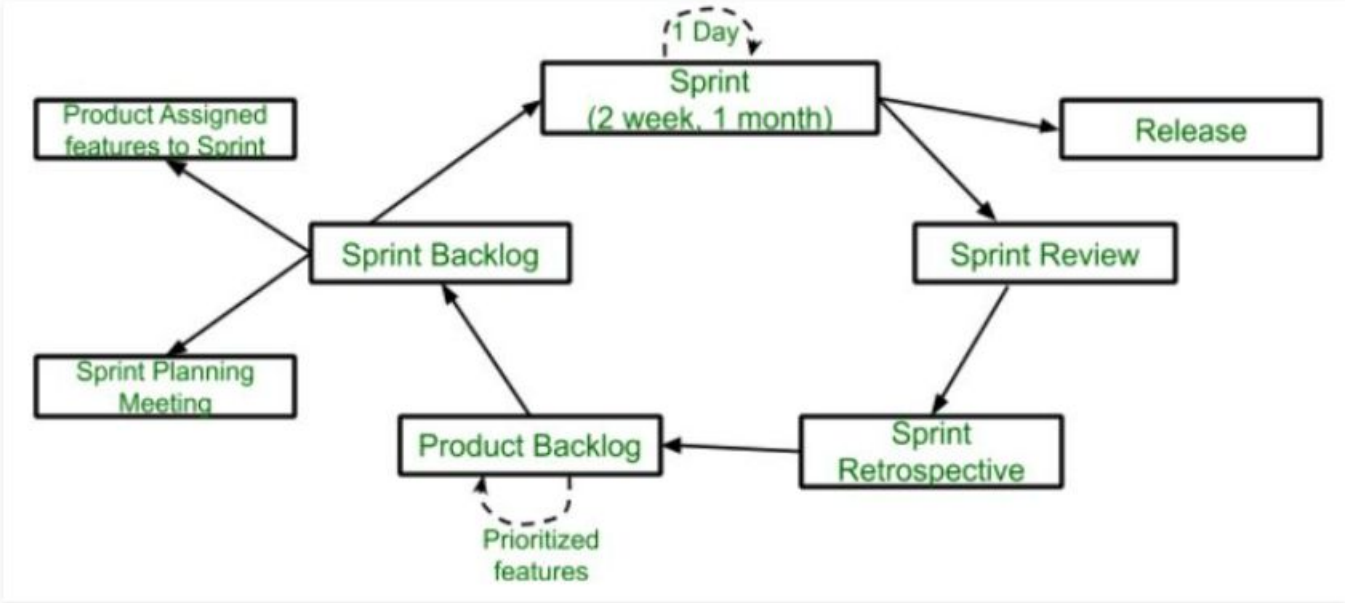
- **Small projects:**
  - XP model is very useful in small projects consisting of small teams as face to face meeting is easier to achieve.

- **Projects involving new technology or Research projects:**
  - This type of projects face changing of requirements rapidly and technical problems.
  - So XP model is used to complete this type of projects.

Scrum

- Scrum is an agile software development method.
- Scrum principles are consistent with the agile platform that are used to guide development activities within a process.
- It is a light-weighted framework.
- Scrum emphasizes self-organization.
- It is simple to understand.
- It includes the framework activities like requirement, analysis, design, evolution and delivery.
- Work tasks occur within a process pattern in each framework activity called as 'sprint'.
- Scrum highlights the use of a set of software process pattern that are effective for the projects with tight timelines, changing requirements and business criticality.
- Scrum framework help the team to work together.

Lifecycle of Scrum



Lifecycle of Scrum

1. **Backlog:**
   - A prioritized list of project requirements or features that provide business value for the customer.
   - Items can be added to the backlog at any time.
   - The product manager accesses the backlog and updates priorities, as required.

2. **Product Backlog**:
   - According to the prioritize features the product is organized.

3. **Sprint Backlog**:
   - Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

Lifecycle of Scrum ….Contd.

4. **Sprints:**
   - A Sprint is a time-box of one month or less.
   - A new Sprint starts immediately after the completion of the previous Sprint.
   - It consists of work units that are required to achieve a requirement defined in the backlog.
   - Changes are not introduced during the sprints.
   - It allows team members to work in short-term but in the stable environment.

5. **Scrum meeting:**
   - The short meetings are held daily by the scrum team.
   - The key questions are asked and answered by all team members.
   - Daily meetings guide to 'knowledge socialization' and that encourages a self-organizing team structure.

Lifecycle of Scrum ….Contd.

6. **Sprint Review**:
   - If the product still have some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.

7. **Sprint Retrospective**:
   - In this stage quality or status of the product is checked.

8. **Release**:
   - When the product is completed then it goes to the Release stage.

9. **Demos:**
   - Deliver the software increment to the customer.
   - Using which the customer evaluates and demonstrates the functionalities.

Advantages of Scrum

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

Disadvantages of Scrum

- Scrum framework does not allow changes into their sprint.
- The framework is not fully described model.
  - ✔ To adopt it, one has need to fill in the framework with his/her own details like Extreme Programming(XP), Kanban, DSDM, etc.
- 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.

Applications of Scrum

- When requirements are not clearly defined, it is practically impossible to make an estimation of the time and costs that would allow to approach the project as a Fixed Price. Also, it is expected that, there will be a large number of changes during the development, which would make the use of a more traditional methodology more difficult.
  - ✔ Scrum is inevitable and more than convenient because of its flexible and adaptable nature throughout the entire process.
- When the probability of changes during the development is high.
- When there is a need to test the solution.
  - ✔ Using Scrum, feedback is obtained early to correct what is necessary and to guarantee that the final product is as expected.
- When the Product Owner (PO) is fully available.
- When the team has self-management skills.
- When contracting is time and materials.
- When the client's culture is open to innovation and adapts to change.

Extreme Programming Vs Scrum

Extreme Programming (XP)	Scrum
In Extreme Programming(XP), teamwork for 1-2 weeks only.	In Scrum framework, team work in iterations called Sprint which are 1-2 month long.
Extreme Programming allow changes in their set timelines.	Scrum model do not allow changes in their timeline or their guidelines.
Extreme Programming emphasizes strong engineering practices	Scrum emphasizes self-organization.
In Extreme Programming, team have to follow a strict priority order or pre-determined priority order.	In Scrum framework, team determines the sequence in which the product will be developed.
Extreme Programming(XP) can be directly applied to a team. Extreme Programming is also known for its **Ready-to-apply** features.	Scrum framework is not fully described. If you want to adopt it, then you need to fill the framework with your own framework methods like XP, DSDM or Kanban.

Kanban

- Kanban term came into existence using the flavors of “visual card,” “signboard,” or “billboard”, “signaling system” to indicate a workflow that limits Work In Progress (WIP).
- It is a system to improve and keep up a high level of production.
- Kanban is one method through which Just-In-Time (JIT), the strategy the organizations employ to control the inventory expenses, is achieved.
- Kanban became an effective tool in support of running a production system as a whole, and it proved to be an excellent way for promoting improvement.
- Its scheduling system tells us what to produce, when to produce it, and how much to produce.
- It smooths out flow, if sized properly.
- It tells us when and where there is a problem in the process.
- Assure there is always enough material on hand to make what is needed.

Core of Kanban

- Kanban development is an approach to incremental, evolutionary process and systems change for organizations.
- Teams practicing other methodologies can use Kanban to improve their existing processes.
- Kanban has three core concepts:

a) **Visualize the workflow**

  - Split the entire work into defined segments or states, visualized as named columns on a wall.
  - Write each item on a card and put in a column to indicate where the item is in the workflow.
  - Kanban teams use Kanban Boards to represent the work and workflow.
  - After visualizing the work, the stakeholders will be able to monitor the flow of work.
  - The bottlenecks in the flow are realized by the stakeholders.

Core of Kanban …. Contd.

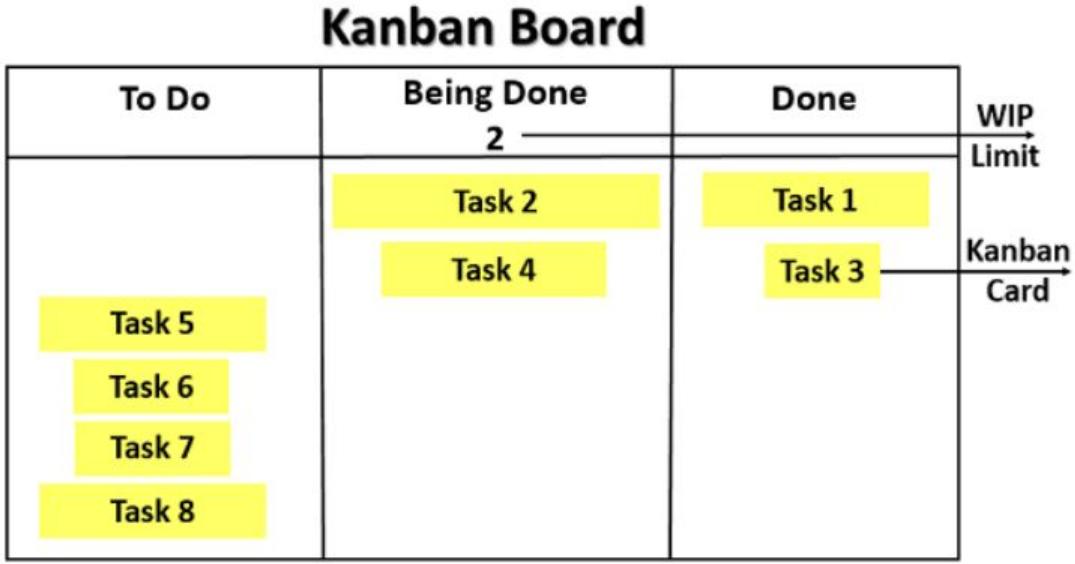
b) Limiting Work-in-Progress (WIP)

- Assign explicit limits to how many items can be in progress at each workflow segment / state. i.e., Work in Progress (WIP) is limited in each workflow state.
- Kanban limits WIP to realize the bottlenecks, to stimulate collaboration, and to continuously improve the system.
- By focusing on flow, Kanban emphasizes finishing work flow over starting new work.
- Limiting the amount of WIP prevents over production.
- When there is too much unfinished work, stakeholders redirect their attention to finishing and collaborate for unfinished work.

Core of Kanban …. Contd.

c) Measure the Lead Time

- Lead Time, also known as cycle time is the average time to complete one item.
- Measure the Lead Time and optimize the process to make the Lead Time as small and predictable as possible.



Advantages of Kanban

- It is very simple and easy to understand.
- Bottlenecks become clearly visible in real-time. This leads people to collaborate to optimize the whole value chain rather than just their part.
- Tends to spread throughout the organization naturally, including sales and management. This increases visibility of everything that is going on at the company.
- Reduces inventory in the range of 25%-75%, thereby reducing company costs.
- Since all segments/states in the workflow are visually organized, the required items, reducing the wait times and ensuring speed, continually support all the tasks in the workflow.
- Overproduction of inventory is avoided, thereby saving resources and time as well. This is termed as eliminating waste.
- Delivers features faster by shortening cycle times.
- Responsive to change.

Disadvantages of Kanban

- Kanban cannot be used as an independent tool. It is not a methodology that could be applied solely rather it can be merged with other processes and systems of a company like JIT, make to order and scrum etc. making these systems more visible.
- As the tasks are continuously shifted between the columns of Kanban board, the prediction of specific timelines for completion of tasks or activities becomes difficult. This is because Kanban acts only as a signaling port in a pull production system.
- Kanban is not suitable for the environments that are dynamic in nature. Because a Kanban system assumes the plans that are stable and consistent to a certain extent, it may become ineffective in industries where the activities are not static.
- Kanban will become very difficult to apply if too much activities or tasks are interrelated in a system. This is because such systems enhance the possibility of transfers of goods and expertise amongst different tasks too often and increases difficulty to keep the pace of all these activities.
- The implementation of the system may result in poor quality outputs. Kanban acts like a monitoring structure that makes the flow of tasks smoother. If any work done is unsatisfactory for the costumer or the company, it would require a rework that could worsen the situation as it will require more time and resources to get completed.

Applications of Kanban

- Useful for situations where operations and support teams have a high rate of uncertainty and variability.
- Kanban is ideal where priorities change very frequently.
- Useful where less wait time is required.

XP Vs Scrum Vs Kanban

Parameters	XP	Scrum	Kanban
Values/ Principles	Communication, simplicity, feedback, courage, respect	Commitment, respect, focus, openness, courage	Kanban Principles and Practices
Design Principle	Simplification of code and accommodation of unexpected changes through refactoring	Complex Design	Limits the amount of Work-In-Progress and ensures waste reduction
Design Complexity	Simple design	Complex design	Simple visual design
Requirements Management	Managed in the form of Story Cards	Requirements managed in the form of artifacts through Sprint Backlog and Product Backlog	Managed using Kanban Boards
Accommodation of Changes	Amenable to change even in later stages of development	Changes not allowed in Sprints	Changes allowed at any time
Work flow approach	No iterations, task flow development	Iterations (Sprints)	Short Iterations

XP Vs Scrum Vs Kanban ….Contd.

Parameters	XP	Scrum	Kanban
Iteration	1 week	Sprint: max 30 days	Delivery cadence (network of meetings)
Team Collaboration	Self organizing teams	Cross functional teams	Team comprises of specialized resources
Coding Standards	Coding standards are used	No coding standards are used	No coding standards are used
Testing approach	Test driven development, including acceptance testing	No formal approach used for testing	Testing done after implementation of each work product
Product delivery	Continuous delivery	Delivery as per Time boxed sprints	Continuous delivery