Unit-1 Software Process Models

❖ Defining Software:

Software is the "collection of computer programs, procedures, rules, associated documents and concerned data with the operation of data processing system".

It also includes representation of pictorial, video and audio information.

Software"s are of two types.

- System Software
- Application Software

❖ System Software

- It is responsible for controlling, integrating the hardware components of a system so the software and the users can work with them.
- Example: Operating System.

❖ Application Software

- It is used to accomplish some specific task.
- It should be collection of small programs.
- Example: Microsoft Word, Excel etc.

❖ Embedded Software

- Embedded software is a piece of software that is embedded in hardware or non-PC devices.
- It is written specifically for the particular hardware that it runs on and usually has processing and memory constraints because of the device"s limited computing capabilities.
- Examples of embedded software include those found in dedicated GPS devices, factory robots, some calculators and even modern smartwatches.

❖ Web Application

- A web-application is an application program that is usually stored on a remote server, and users can access it through the use of Software known as web-browser.
- It is a type of computer program that usually runs with the help of a web browser and also uses many web technologies to perform various tasks on the internet.

❖ **Artificial intelligence Software**

- Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.
- Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.
- Artificial Intelligence (AI) Software is a computer program which mimics human behavior by learning various data patterns and insights.
- Top features of AI software include Machine Learning, Speech & Voice Recognition, Virtual Assistant etc.

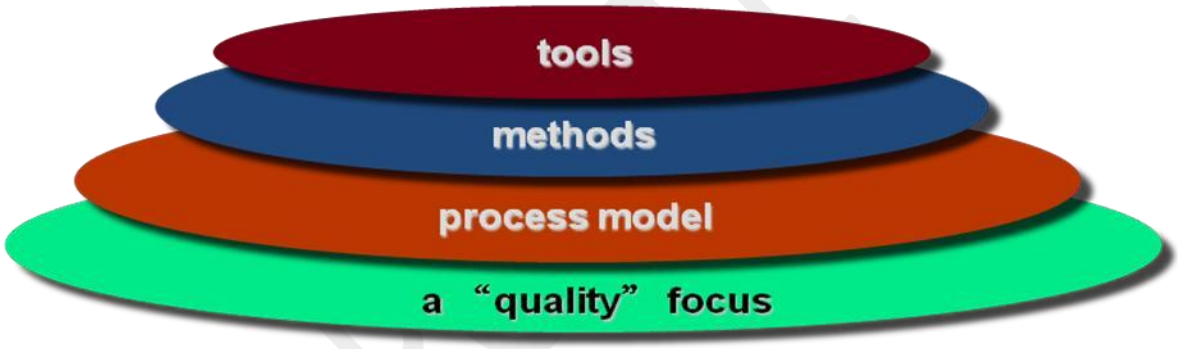
❖ Software Characteristics

- The characteristics of software decide whether the software is good or bad.
- **Understandability**
  - ✓ Software should be easy to understand
  - ✓ It should be efficient to use.
- **Cost**
  - ✓ Software should be cost effective as per its usage.
- **Maintainability**
  - ✓ Software should be easily maintainable and modifiable in future.
- **Modularity**
  - ✓ Software should have modular approach so it can be handled easily for testing.
- **Functionality**
  - ✓ Software should be functionally capable to meet user requirements.
- **Reliability**
  - ✓ It should have the capability to provide failure-free service.
- **Portability**
  - ✓ Software should have the capability to be adapted for different environments.
- **Correctness**

- ✓ Software should be correct as per its requirements.

- **Documentation**
  - ✓ Software should be properly documented so that we can re-refer it in future.
- **Reusability**
  - ✓ It should be reusable, or its code or logic should be reusable in future.
- **Interoperability**
  - ✓ Software should be able to communicate with various devices using standard bus structure and protocol.

❖ Software Engineering – A Layered Approach



- Software engineering can be viewed as a layered technology.
- It contains process, methods, and tools that enables software product to be built in a timely manner.

- **A Quality Focus Layer**
  - ✓ Software engineering mainly focuses on quality product.
  - ✓ It checks whether the output meets with its requirement specification or not.
  - ✓ Every organization should maintain its total quality management.
- **Process Layer**
  - ✓ It is the heart of software engineering.
  - ✓ It is also work as foundation layer.

- ✓ Software process is a set of activities together if ordered and performed properly, then the desired result would be produced.
- ✓ It defines framework activities.
- ✓ The main objective of this layer is to deliver software in time.

- **Method Layer**
  - ✓ It describes „how-to" build software product.
  - ✓ It creates software engineering environment to software product using CASE tools
- **Tools Layer**
  - ✓ It provides support to below layers.
  - ✓ Due to this layer, process is executed in proper manner.

❖ Generic Process Model or Generic view of software engineering

- The work associated with software engineering can be categorized into three generic phases.
  - ✓ Definition phase
  - ✓ Development phase
  - ✓ Support phase

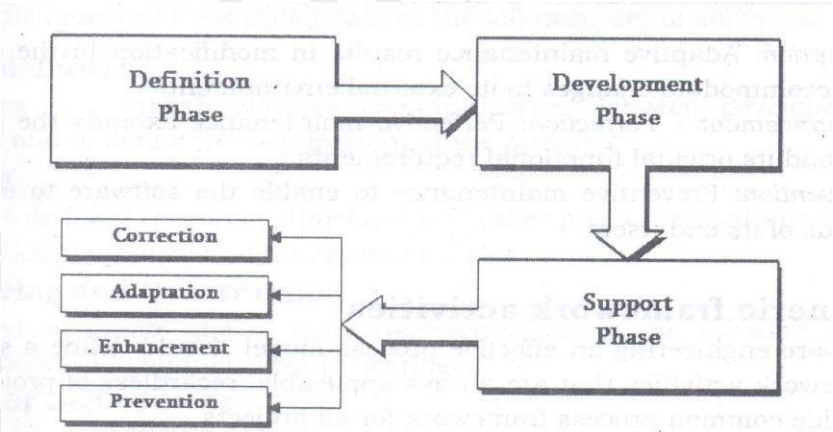


Figure 1.4 Software generic view

- **Definition phase**
  - ✓ It focuses on what part.
  - ✓ During definition phase, the software engineer attempts to identify
    - What information is to be processed?
    - What function and performance are desired?

- What system behavior can be expected?
- What interfaces are to be established?
- What design constraints exist?
- What validation criteria are required to define a successful system?

✓ Three main activities performed during this phase:

- System or information engineering
- Software project planning
- Requirement analysis

- **Development Phase**

✓ It focuses on how part of the development.

✓ During development a software engineer attempts to define

- How data are to be structured?
- How function is to be implemented within a software architecture?
- How interfaces are to be characterized?
- How design will be translated into a programming language?
- How testing will be performed?

✓ Main activities are performed under this phase are:

- Software design
- Code generation
- Software testing

- **Support Phase**

✓ The support phase focuses on change associated with error correction.

✓ Four types of change are encountered during the support phase.

✓ **Correction**: corrective maintenance changes the software to correct defects.

✓ **Adaption**: Adaptive maintenance results in modification to the software to accommodate changes to its external environment.

✓ **Enhancement / perfection**: Perfective maintenance extends the software beyond its original functional requirements.

✓ **Prevention**: Preventive maintenance to enable the software to serve the needs of its end

users.

❖ Generic Framework Activities

- **Project Definition:**
  - ✓ It requires to establish effective communication between developer and customer and to define the requirement of the project.
- **Planning:**
  - ✓ It requires defining resources, timelines and other project related information and assessing technical and management risks.
- **Engineering & Construction:**
  - ✓ It required for create one or more representations of the software and to generate code and conduct through testing.
- **Release or Deployment:**
  - ✓ It required to install the software in its target environment and to provide customer support.
- **Customer Use:**
  - ✓ It required for obtaining customer feedback based on use and evaluation of project delivered during release.

❖ Umbrella Activities

- Umbrella activities are performed throughout the process.
- These activities are independent of any framework activity.
- The umbrella activities are given below:
- **Software project tracking and control:**
  - ✓ It assess progress against the plan and take actions to maintain the schedule.
- **Formal Technical Review:**
  - ✓ This includes reviewing the techniques that has been used in the project.
- **Software Quality Assurance:**
  - ✓ This is very important to ensure the quality management of each part to ensure them.

- **Document Preparation and production**
  - ✓ All the project planning and other activities should be hardly copied and the production gets started here.
- **Reusability Management**
  - ✓ This includes the backing up of each part of the software project they can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand.
- **Measurement**
  - ✓ This will include all the measurement of every aspects of the software project.
- **Risk Management**
  - ✓ Risk management is a series of steps that help a software team to understand and manage uncertainty.

❖ Software Development Life Cycle

- Every system has a life cycle.
- It begins when a problem is recognized, after then system is developed, grows until maturity and then maintenance needed due to change in the nature of the system. So, it died and new system or replacement of it taken place.
- SDLC is a framework that describes the activities performed at each stage of a software development project.

- Different Software life cycle models
  - ✓ Classical Waterfall model
  - ✓ Iterative waterfall model
  - ✓ Incremental Model
  - ✓ RAD Model
  - ✓ Spiral Model
  - ✓ Prototype Model

❖ Waterfall Model

- Waterfall model was proposed by Royce in 1970.
- It is also called as "traditional waterfall model" or "conventional waterfall model".
- This model break down the life cycle into set of phases like.
  - ✓ Feasibility study
  - ✓ Requirements analysis and specification
  - ✓ Design
  - ✓ Coding and unit testing
  - ✓ Integration and system testing
  - ✓ Maintenance.
- During each phase of life cycle, a set of well defined activities are carried out. And each phase required different amount of efforts.
- The phases between feasibility study and testing known as development phases.
- Among all life cycle phases maintenance phase consumes maximum effort.

- Feasibility Study:
  - ✓ Aim of this phase is to determine whether the system would be financially and technically feasible to develop the product.

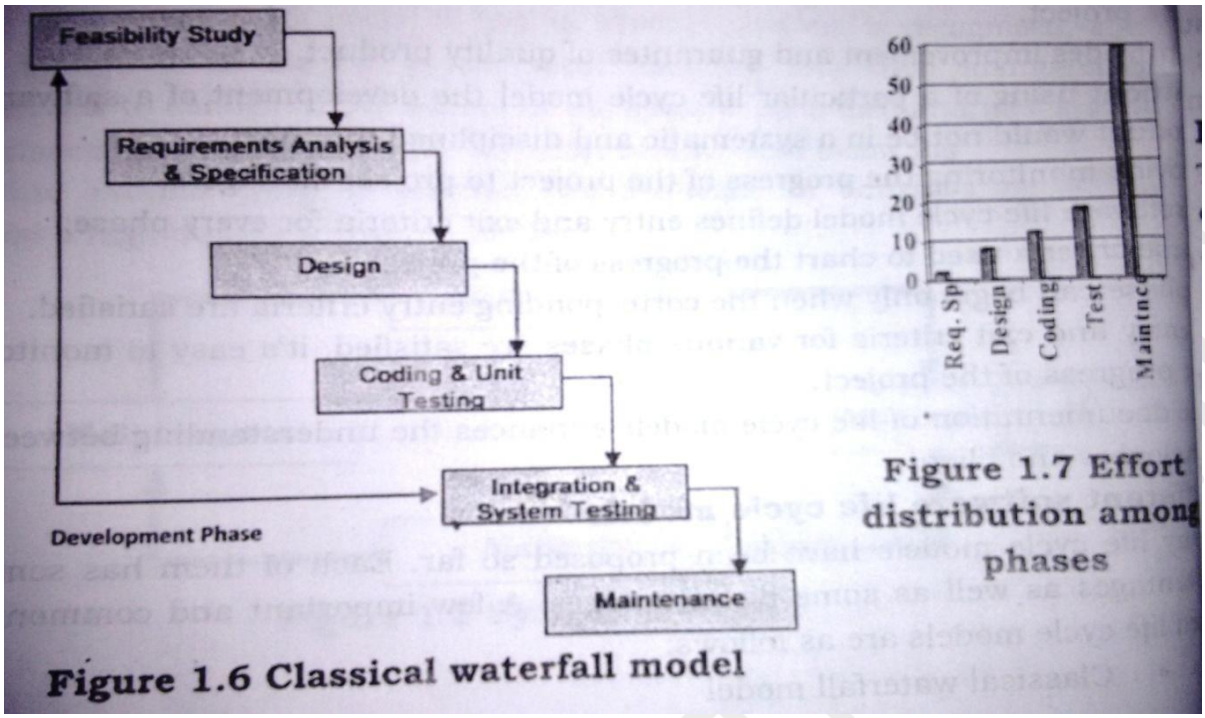


Figure 1.6 Classical waterfall model

Figure 1.7 Effort distribution among phases

- Requirement Analysis and Specification:
  - ✓ Aim of this phase is to understand the exact requirements of the customer and to document them properly.
- Design:
  - ✓ The goal of design phase is to transform the requirements specified in SRS document into a structure that is suitable for implementation in some programming language.
- Coding and Unit Testing
  - ✓ It is also called as implementation phase.
  - ✓ Aim of this phase is to translate the software design into source code and unit testing is done module wise.
- Integration and System Testing
  - ✓ Once all the modules are coded and tested individually, integration of different modules is undertaken.
  - ✓ Goal of this phase is to ensure that the developed system works well to its requirements described in the SRS document.
- Maintenance
  - ✓ It requires maximum efforts to develop software product.

- ✓ This phase is needed to keep system operational.
- ✓ General maintenance is needed due to change in the environment or the requirement of the system.

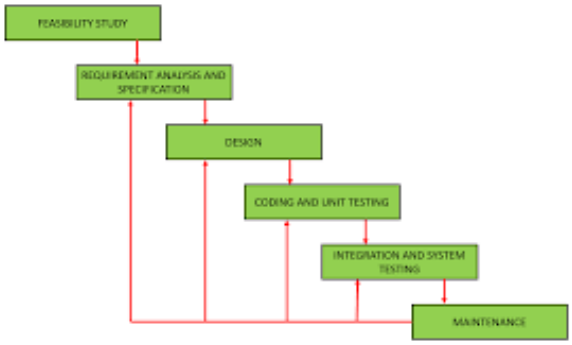
- **Waterfall Model – Advantages**
  - ✓ It is simple and easy to understand and use.
  - ✓ Each phase has well defined input and output.
  - ✓ Waterfall model works well for smaller projects where requirements are very well understood.
  - ✓ In divides complex tasks into smaller, more manageable works.

- **Waterfall Model – Disadvantages**
  - ✓ It is a theoretical model, as it is very difficult to strictly follow all the phases in all types of projects.
  - ✓ It may happen that the error may be generated at any phase and encountered in later phase. So, it is not possible to go back and solve the error in this model.
  - ✓ High amount of risk.
  - ✓ It is a document driven process that requires formal documents at the end of each phase.

- **Waterfall Model – When to use?**
  - ✓ Requirements are very well known and fixed.
  - ✓ Product definition is stable.
  - ✓ Technology is understood.
  - ✓ When the project is short.

❖ Iterative waterfall model

- Classical waterfall model is idealistic: it assumes that no defect is introduced during any development activity.
- But in practice: defects do get introduced in almost every phase of the life cycle.
- Even defects may get at much later of the life cycle.
- So, the solution of this problem is iterative waterfall model.



When to use Iterative Waterfall Model?

1. The prerequisite of being well-defined and comprehended.
2. The development team is gaining knowledge about new technologies.
3. Certain characteristics and objectives carry a significant chance of failure in the future.

Advantages of Iterative Waterfall Model

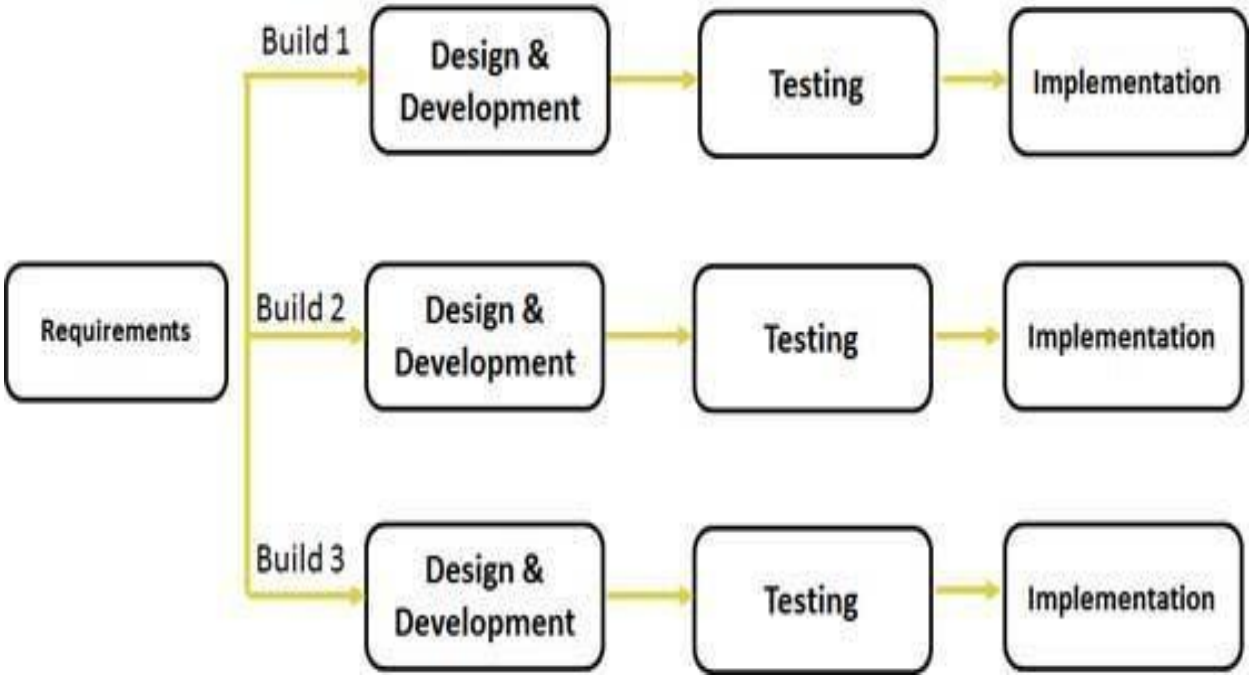
1. Phase Containment of Errors: The principle of detecting errors as close to their points of commitment as possible is known as Phase containment of errors.
2. Collaboration: Throughout each stage of the process, there is collaboration between the business owners and developers. This ensures that the website meets the needs of the business and that any issues or concerns are addressed in a timely manner.
3. Flexibility: The iterative waterfall model allows for flexibility in the development process. If changes or new requirements arise, they can be incorporated into the next iteration of the website.
4. Testing and feedback: The testing stage of the process is important for identifying any issues or bugs that need to be addressed before the website is deployed. Additionally, feedback from users or customers can be gathered and used to improve the website in subsequent iterations.
5. Scalability: The iterative waterfall model is scalable, meaning it can be used for projects of various sizes and complexities. For example, a larger business may require more iterations or more complex requirements, but the same process can still be followed.
6. Maintenance: Once the website is live, ongoing maintenance is necessary to ensure it continues to meet the needs of the business and its users. The iterative waterfall model can be used for maintenance and improvement cycles, allowing the website to evolve and stay up-to-date.
7. Easy to Manage: The iterative waterfall model is easy to manage as each phase is well-defined and has a clear set of deliverables. This makes it easier to track progress, identify issues, and manage resources.

Drawbacks of Iterative Waterfall Model

- Difficult to incorporate change requests
- Incremental delivery not supported
- Overlapping of phases not supported
- Risk handling not supported
- Limited customer interactions

❖ Incremental Model

- The incremental model is also referred as the successive version of waterfall model using incremental approach.
- In this model, the system is broken down into several modules which can be incrementally implemented and delivered.
- First develop the core product of the system.
- The core product is used by customers to evaluate the system.
- The initial product skeleton is refined into increasing levels of capability: by adding new functionality in successive version.



- **Incremental Model – Advantages**
  - ✓ Each successive version performing more useful work than previous version.
  - ✓ The core modules get tested thoroughly, there by reducing chance of error in final product.

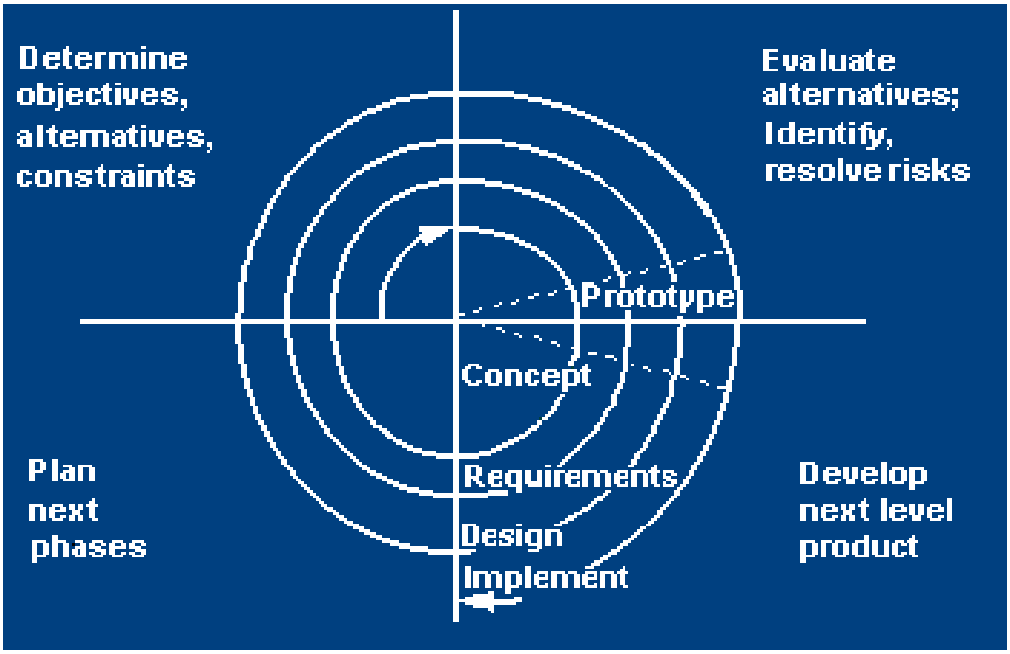
The model is more flexible and less costly to change the scope and requirements.

- ✓ User gets a chance to experiment with partially developed software.
- ✓ Feedback providing at each increment is useful for determining the better final product.

- **Incremental Model – Disadvantages.**
  - ✓ Sometimes it is difficult to subdivide problems into functional unit.
  - ✓ Model can be used for very large problems.
  - ✓ It needs good planning and design.
- **Incremental Model – When to use?**
  - ✓ The incremental model is used when the problem is very large and user requirements are not well specified at initial stage.
  - ✓ This model is used when the system is modularized and all the requirements are well defined.

❖ Spiral Model

- This model is proposed by Boehm in 1986.
- In application development, spiral model uses fourth generation (4GL) languages and development tools.
- In pictorial view, this model appears like a spiral with many loops.
- Each loop of the spiral represents a phase of the software process.
  - ✓ The innermost loop might be concerned with system feasibility
  - ✓ The next loop with system requirement definition.
  - ✓ The next one with system design and so on.
  - ✓ Each loop in the spiral split into four sectors. (quadrants)

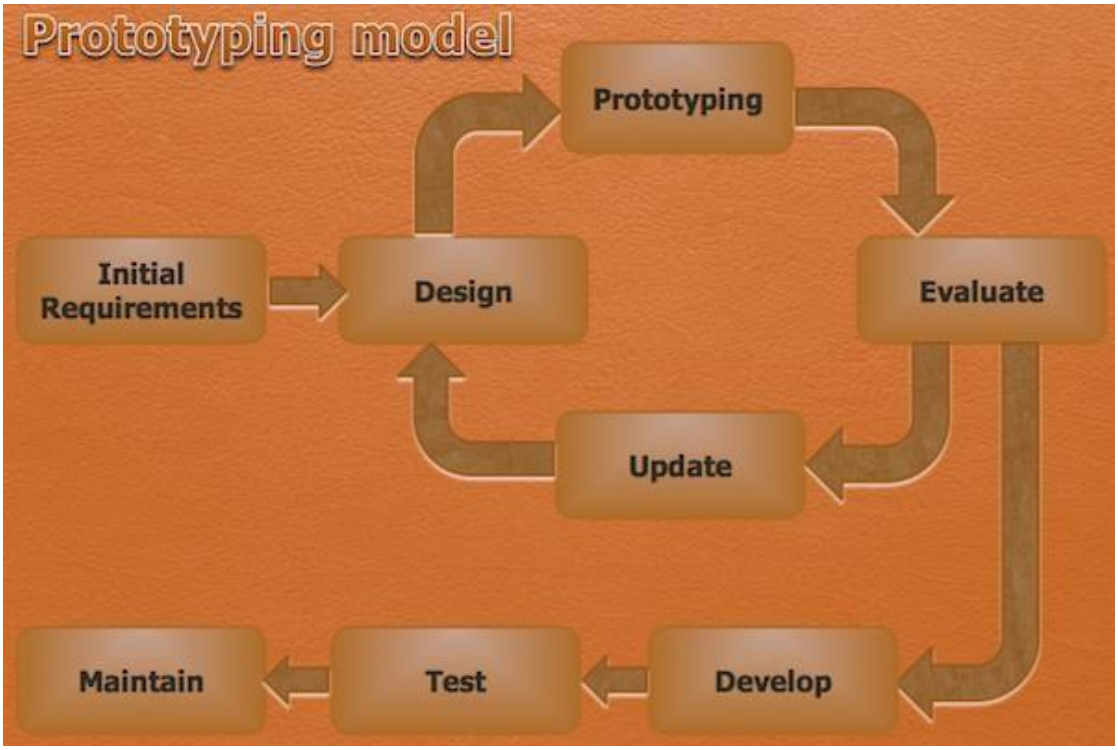


- 1st Quadrant: Determine Objectives
- 2nd Quadrant: Identify and resolve risks
- 3rd Quadrant: Develop next level product
- 4th Quadrant: Review and planning
- In spiral model, at any point, Radius represents: cost and Angular dimension represent: progress of the current phase.
- **Spiral Model – Advantages**
  - ✓ It is more flexible, as we can easily deal with changes.
  - ✓ Due to user involvement, user satisfaction is improved.
  - ✓ New idea and functionality can be easily added at later stage.
- **Spiral Model – Disadvantages**
  - ✓ It is applicable for large problem only.
  - ✓ It can be more costly to use.
  - ✓ It is more complex to understand.
  - ✓ More number of documents are needed as more number of spirals.
- **Spiral Model – When to Use?**
  - ✓ Used when medium to high risk projects.

- ✓ When user are unsure for their needs.
- ✓ When requirement are complex.

❖ Prototype Model

- Prototype is a working physical system or subsystem.
- Prototype is nothing but a tip implementation of a system.
- In this model, before starting actual development, a working prototype of the system should be built first.
- A prototype is actually a partial developed product.
- Compared to the actual software, a prototype usually have,
  - ✓ Limited functional capabilities
  - ✓ Low reliability
  - ✓ Inefficient performance
- Prototype usually built using several shortcuts, and these shortcuts might be inefficient, inaccurate or dummy functions.
- Prototype model is very useful in developing GUI part of system.



In working of the prototype model, product development starts with initial requirements gathering phase.

- Then, quick design is carried out and prototype is built.
- The developed prototype is then submitted to the customer for his evaluation.
- Based on customer feedback, the requirements are refined and prototype is modified.
- This cycle of obtaining customer feedback and modifying the prototype continues till the customers approve the prototype.
- The actual system is developed using different phases of iterative waterfall model.

- **Prototype Model – Advantages**

  - ✓ A partial product is built at initial stage, so customers can get a chance to have a look of the product.
  - ✓ New requirements can be accommodate easily.
  - ✓ Quicker user feedback is available for better solution.
  - ✓ As the partial product is evaluated by the end users, more chance of user satisfaction.

- **Prototype Model – Disadvantages**
  - ✓ The code for prototype model is usually thrown way. So wasting of time is there.
  - ✓ The construction cost of developing the prototype is very high.
  - ✓ If the end user is not satisfied with the initial prototype, then he/she may loose interest in the final product.

- **Prototype Model – When to Use?**
  - ✓ This model is used when the desired system needs to have a lot of interactions with end users.
  - ✓ This type of model generally used in GUI type of development.

- **Prototype Model – Disadvantages**
  - ✓ The code for prototype model is usually thrown way. So wasting of time is there.
  - ✓ The construction cost of developing the prototype is very high.
  - ✓ If the end user is not satisfied with the initial prototype, then he/she may loose interest in the final product.

- **Prototype Model – When to Use?**
  - ✓ This model is used when the desired system needs to have a lot of interactions with end users.
  - ✓ This type of model generally used in GUI type of development.

❖ Agile Model

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

Fig. Agile Model

Phases of Agile Model:

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

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

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

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

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

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

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

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

When to use the Agile Model?

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

Advantage(Pros) of Agile Method:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

Disadvantages (Cons) of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

❖ **Give the difference between Agile model and iterative model**

Agile Model	Iterative Model
The Agile Model of software development is a type of model in which specifications and solutions enhance through the continuous collaboration of functional teams.	The Iterative Model of software development is a type of model in which implementation starts with small elements, and iteratively evolves to the final solution through a collaboration of functional teams.
The process of development in this model is called Sprint.	The process of development in this model is called an Iteration.
With the completion of a sprint, a meeting is carried out.	With the completion of an iteration, a meeting is carried out.
The preceding sprint affects the subsequent sprint.	The preceding iteration affects the subsequent iteration.
Collaborating teams can review products during a Sprint.	Collaborating teams can review products on the baseline of iteration.
There are two main roles in this model Scrum Master and Team member.	There are two roles in this model Project Manager and Team Member.
Scrum Master is responsible for the facility and team members do the estimation.	The Project Manager is responsible for the estimation and completion of each iteration.

❖ Types of widely used Agile Models

Scrum

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

Scrum is, undoubtedly, the most used of the many frameworks underpinning Agile methodology. Scrum is characterized by cycles or stages of development, known as sprints, and by the maximization of development time for a software product towards a goal, the Product Goal. This Product Goal is a larger value objective, in which sprints bring the scrum team product a step closer.

It is usually used in the management of the development of software products but can be used successfully in a business-related context.

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

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

eXtreme Programming(XP)

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

Extreme Programming (XP) is a methodology that emphasizes teamwork, communication, and feedback. It focuses on constant development and customer satisfaction. Similar to scrum, this method also uses sprints or short development cycles. This is developed by a team to create a productive and highly efficient environment.

The extreme Programming technique is very supportive in a situation of constant and varying demands from the customers. It motivates the developers to accept changes in the customer‟s demands, even if they pop up in an advanced phase of the development process.

In Extreme Programming, the project is tested from the initial stages by collecting feedback that progresses the output of the system. This also presents a spot check to implement easily any customer requirements.