**Chapter 1: INTRODUCTION**

**Topic – 1: Engineering**

* **Engineering:** Application of science, tools & methods to find **cost effective** solutions to problems.
* **Software engineering:** Systematic and disciplined approach to develop, operate & maintain software.

**Topic – 2: Broad Categories of Software**

* **System software**
* **Real-time software**
* **Application software**
* **Engineering & scientific software**
* **Embedded software**
* **Product-line/ business software**
* **Web applications**
* **Artificial intelligence software**

**Topic – 3: Legacy Software**

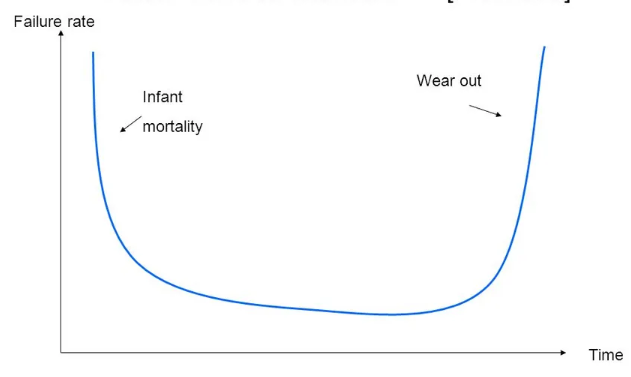
* **Old** software which are not used much anymore.
* Might have **no** **documentation**.
* Contains inextensible code.
* The test cases and results are **not** up to expectations anymore.

**Topic – 4: Software Characteristics**

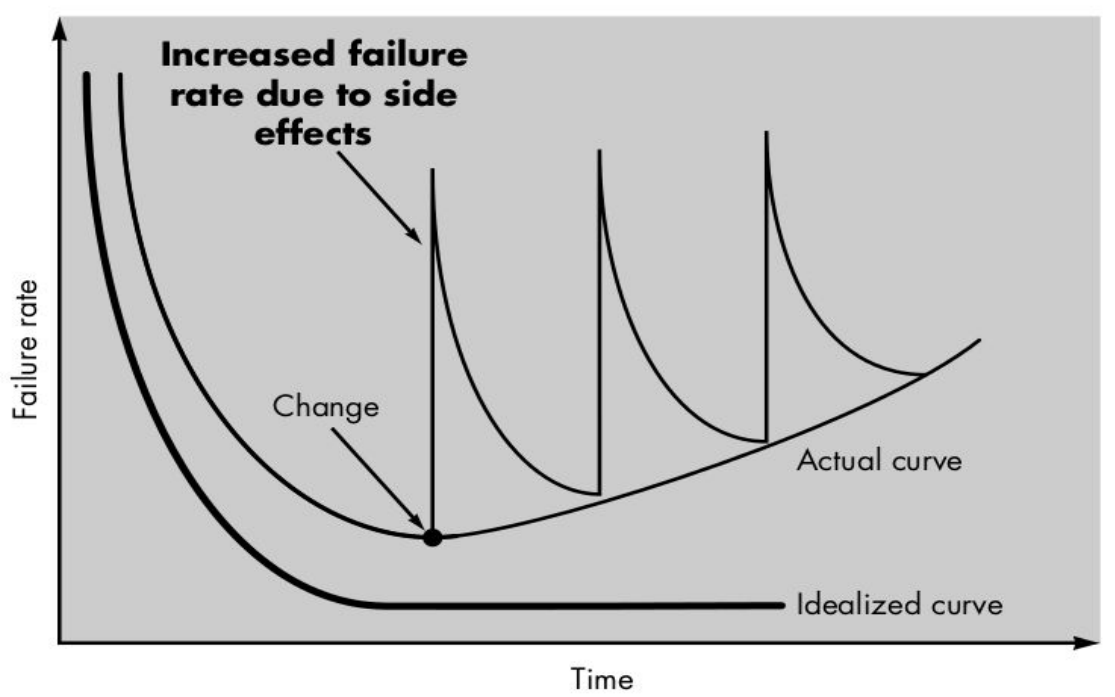
* ***“Software are developed or engineered, not manufactured.”***
* Means, software is **set of instructions** which are **not** physically moulded.

**Topic – 5: Hardware Performance**

**Hardware Failure Curve**



**Ideal Curve v/s Actual Failure Curve**



**Topic – 6: Generic Requirements of SE**

**Generic Requirements**

* System engineering
* Project planning
* Requirement analysis

**Development Phase**

* Focuses on what **set of operation** to perform.
* Also, what **way to use the data structures** in.
* And, how to **translate the design** into programming language.

**Support Phase**

* Focuses on **making changes** to software.
* It can be due to **client’s requirements**, or even for **correcting errors** etc.
* Types of changes:
  + **Correction** (as per report by customers)
  + **Adaptations** (mounting it to required environment)
  + **Enhancement** (maintenance)
  + **Prevention** (security & modifying for easy changing)

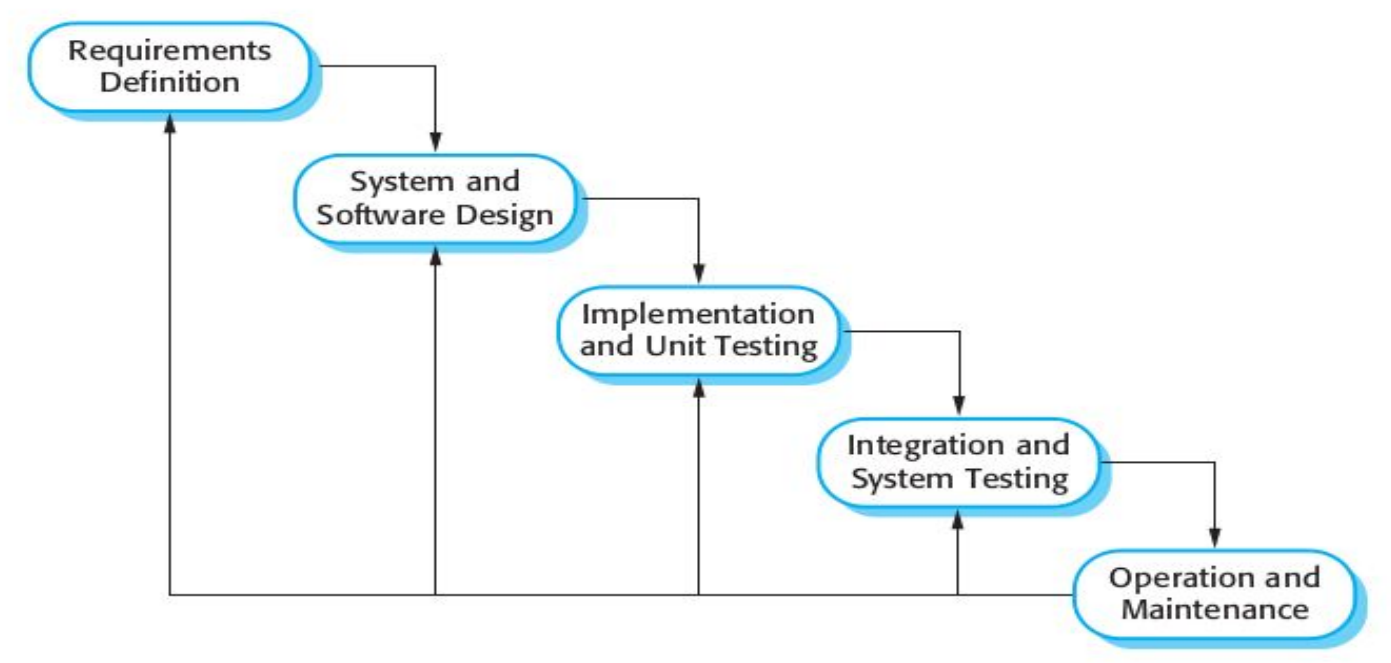
**Topic – 7: Process Model**

**About Model**

* Means the model being developed to meet requirements of the client.
* And as this model contains **sequences of processes**, so we call it ***process model***.

**Topic – 8: Waterfall Model**

**Model Diagram**



**Disadvantages**

* Difficult to make changes.
* Suitable only for making **predictable changes**.
* Suitable for **large software** only.

**Topic – 9: Evolutionary Software Process Model (ESPM)**

**Introduction**

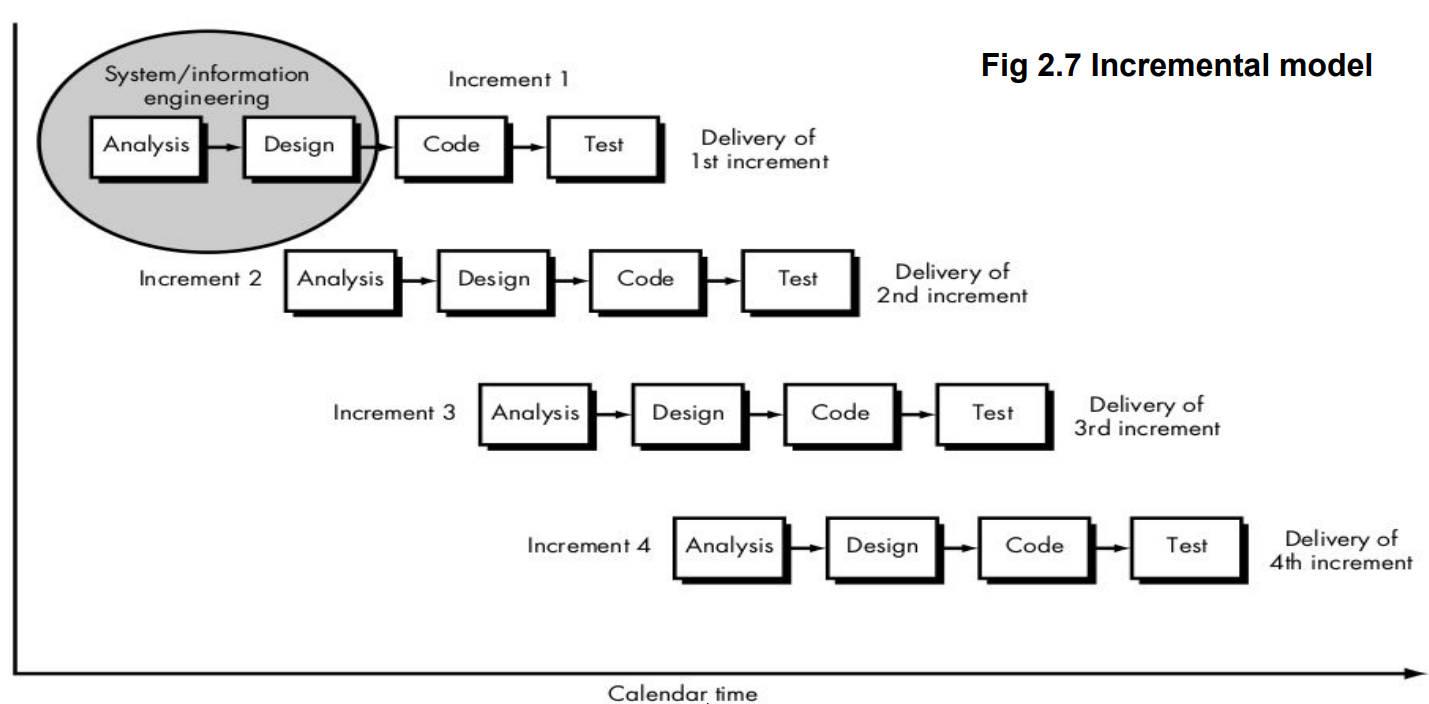
* These types of software can be **evolved** over time.
* For example, various **MS office** components.

**Characteristics of ESPM (Summary)**

* Incremental model
* Spiral model
* Concurrent development model

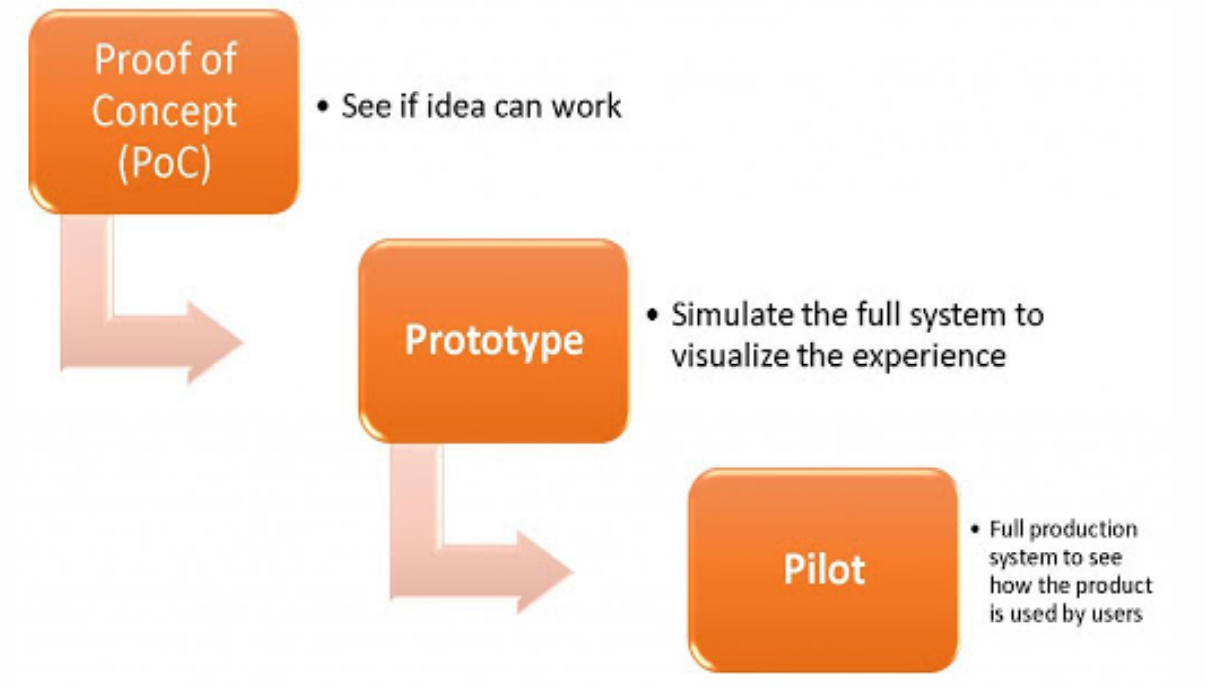
**Topic – 10: Characteristics of ESPM**

**Incremental Mode**



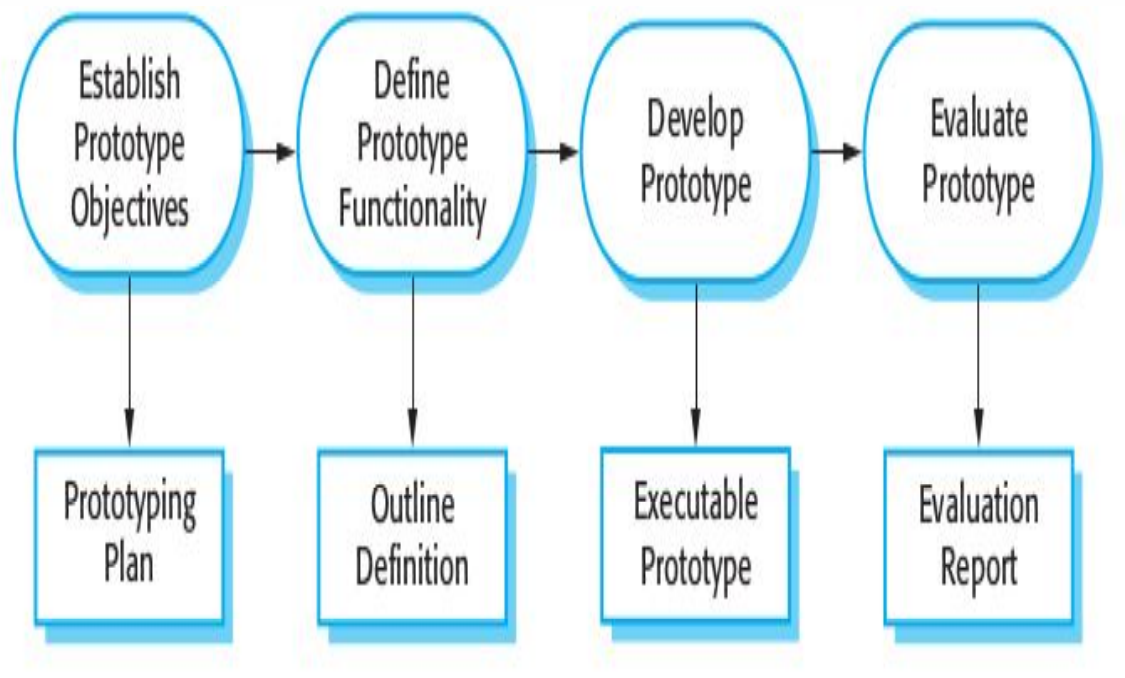
**Topic – 11: Prototype**

**Understanding Prototype**



**Prototyping**

* Open to experiments.
* Rapid development.
* Cheap development cost.

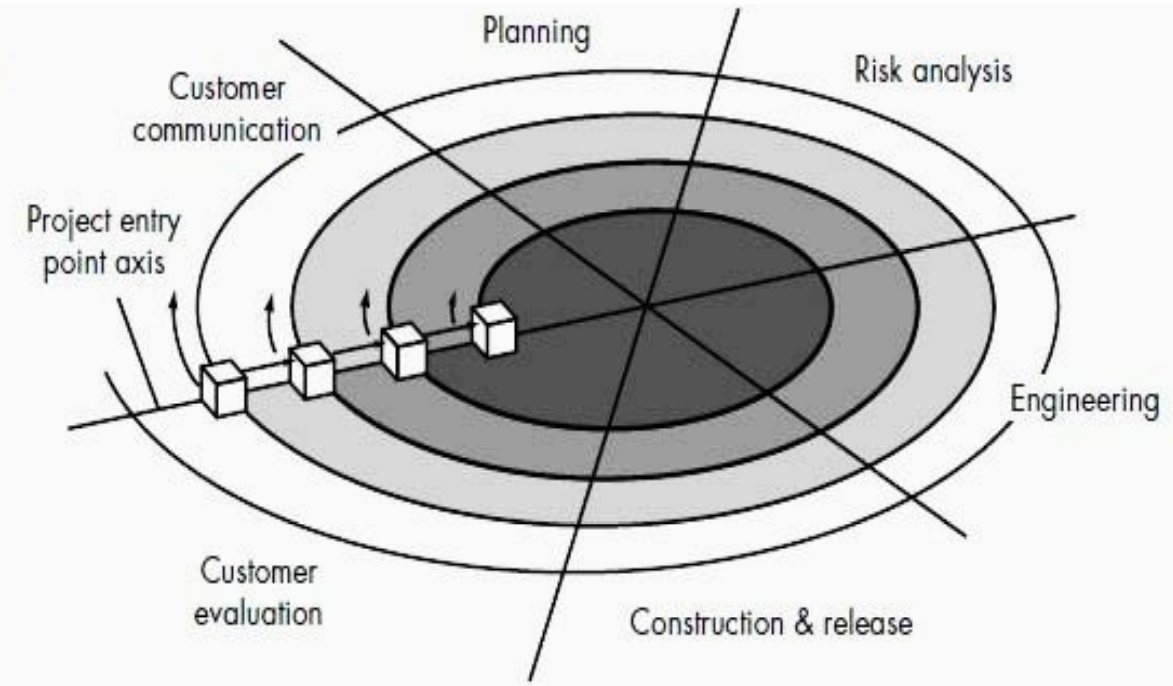


**Topic – 12: Spiral Model**

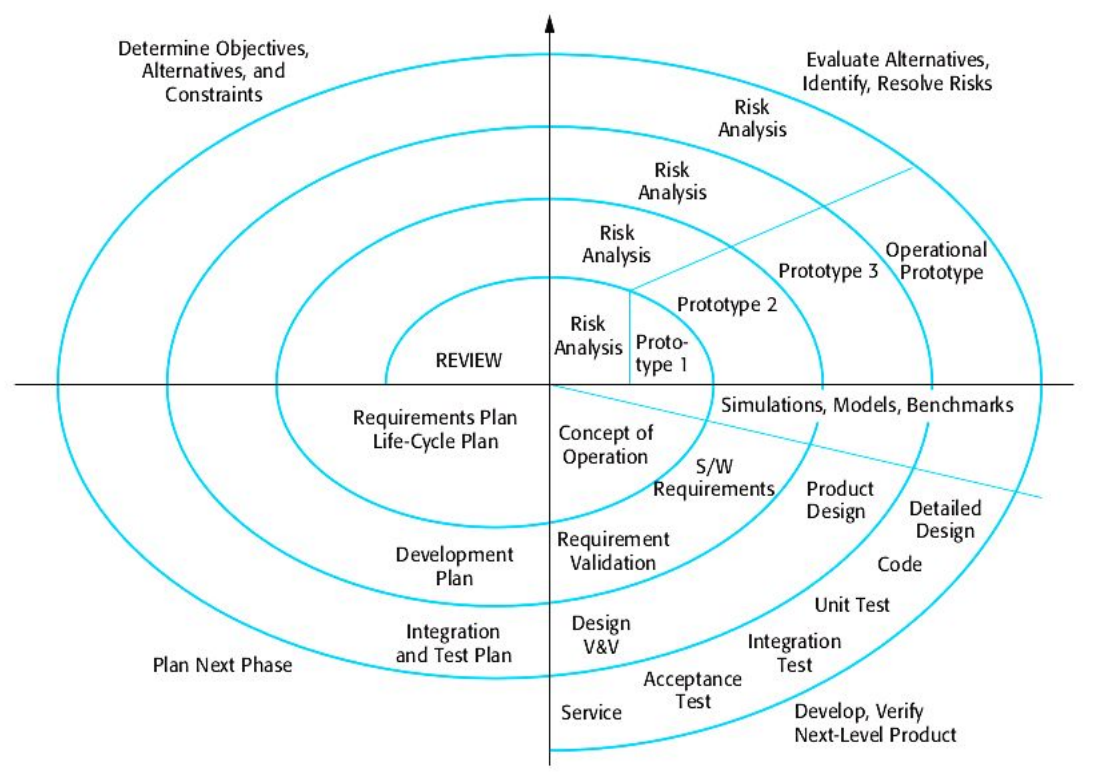
**Introduction**

* Similar to **incremental model**, first model is just a **prototype** which is incrementally improved in later versions.

**Parts Of Spiral Model**



**Brief Diagram**



**Benefits**

* Customers can **immediately react** & provide reviews.
* Stronger approach to **slowly but carefully** plan each phase.

**Topic – 13: Concurrent Process**

**Introduction**

* Performing **technical tasks** concurrently.
* All **tasks** are developed/evolved concurrently too.

**Concurrent Development Model**



**Topic – 14: Agile Model**

**Introduction**

* Name ***agile*** is derived from quality of being **agile**.
* It can respond quickly to **changing customer demands**.
* Is an **incremental approach** to develop software.
* **Documentations** are documented **concurrently** when the software is being developed.
* This model ensures that developers get idea of **processes & tools** being used by other members in the team.
* **Customer involvement** in development of the project is very important.
* **Agile** promotes **collaboration** among cross-functional team.
* Agile **isn’t** good for **system software development**, but is good for **web development**.
* Each phase is handled by a **group of specialists** for that specific phase.

**Reason For Change**

* New business or **market conditions**.
* **Modification** or addition of features.
* **Reorganizing development team** due to changes in developing company.
* **Change in budget** or **time constraints**.

**Agile Principles**

|  |  |
| --- | --- |
| **Principle** | **Description** |
| **Customer involvement** | **-** |
| **Incremental delivery** | **New feature demands by customers are added at the next iteration of model.** |
| **People, not process** | **Full creative permission must be given to talented team members.** |
| **Embrace change** | **Expect changes.** |
| **Maintain simplicity** | **Keep the process and system being developed simple & eliminate complexities.** |

**Note!**

**🡪 Agile mustn’t be used where the project plan is unlikely to change.**

**🡪 It also shouldn’t be used where there is a large collaboration of various teams.**

**Agile Process Model**

* Extreme programming (XP)
* Adaptive software development (ASD)
* Dynamic system development method (DSDM)
* Scrum
* Feature driven development (FDD)
* Crystal
* Agile modelling (AM)

**Topic – 15: Extreme Programming (XP)**

**Procedure**

* **Test cases** are developed for each requirement.
* Requirement is broken into **small tasks** & **multiple programmers** work on each task.
* Then the requirements are **directly implemented** into code.
* Codes must **satisfy** the test cases in order to be deployed.
* Customers will submit **stories** to give developers idea about how they are planning to use the software in real-life.
* There is even a role called ***tracker*** required in **XP**.
* A **tracker** keeps **track record** of the time, resources & test data etc.



**XP Principles**

|  |  |
| --- | --- |
| **Principle/Practice** | **Description** |
| **Incremental planning** | **Requirements are broken into tasks.** |
| **Small releases** | **Important requirements are developed first, rest are incrementally added.** |
| **Simple design** | **-** |
| **Test-first development** | **-** |
| **Refactoring** | **Modifying codes immediately when the requirements come.** |
| **Pair programming** | **Programmers work in pair to complete tasks & support.** |
| **Collective ownership** | **Everyone maintain each type of code, there is no separation among the developers as per their expertise.** |
| **Continuous integration** | **New functionalities are integrated & tested.** |
| **Sustainable pace** | **Overtime on employees aren’t imposed on employees as it could reduce quality of program in exchange of short-term productivity.** |
| **On-site customers** | **Customers must be available to the development team for full time, as customer is a member of development team too.** |

**XP Testing Features**

* High user involvement in **testing process**.
* Uses **automated** **testing frameworks**.

**Pair Programming**

* In this pair, one is an **observer** while another one is a **programmer**.
* They discuss & start together so that the second person **doesn’t** has to spend time knowing about the project separately.
* They have **collective ownership** on the project.
* They share **informal reviews** to other team members.

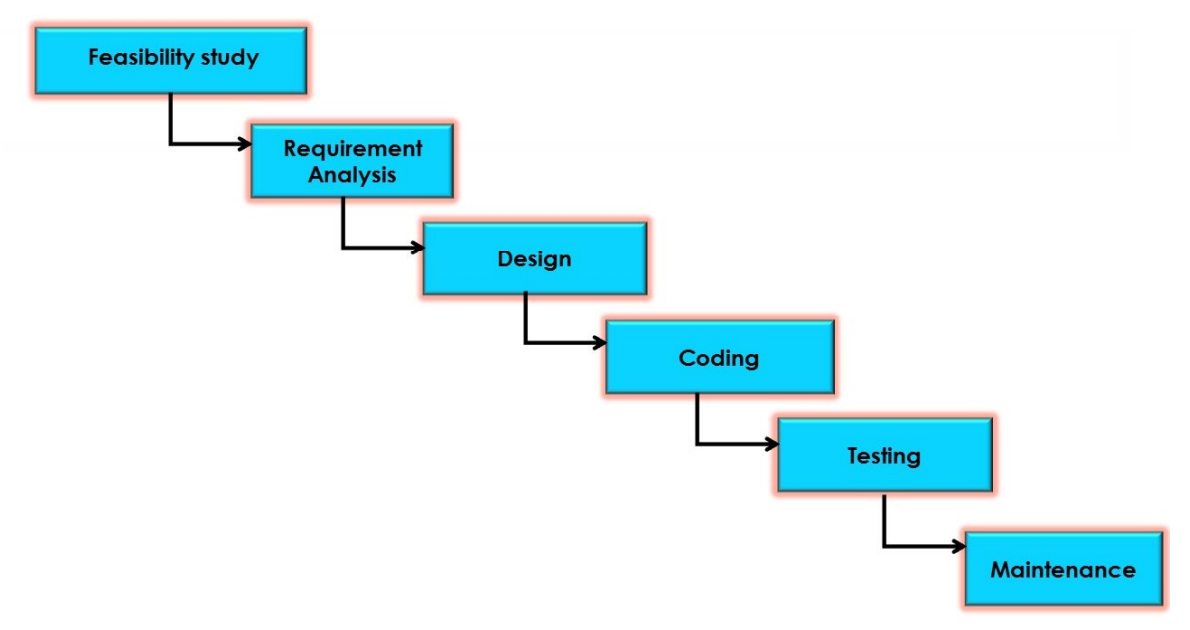
**Refactoring**

* This modification has to do more with **improving structure** of code.
* It includes **cleansing** & **improving parts** that may produce bugs etc.

**Topic – 16: Waterfall Model**

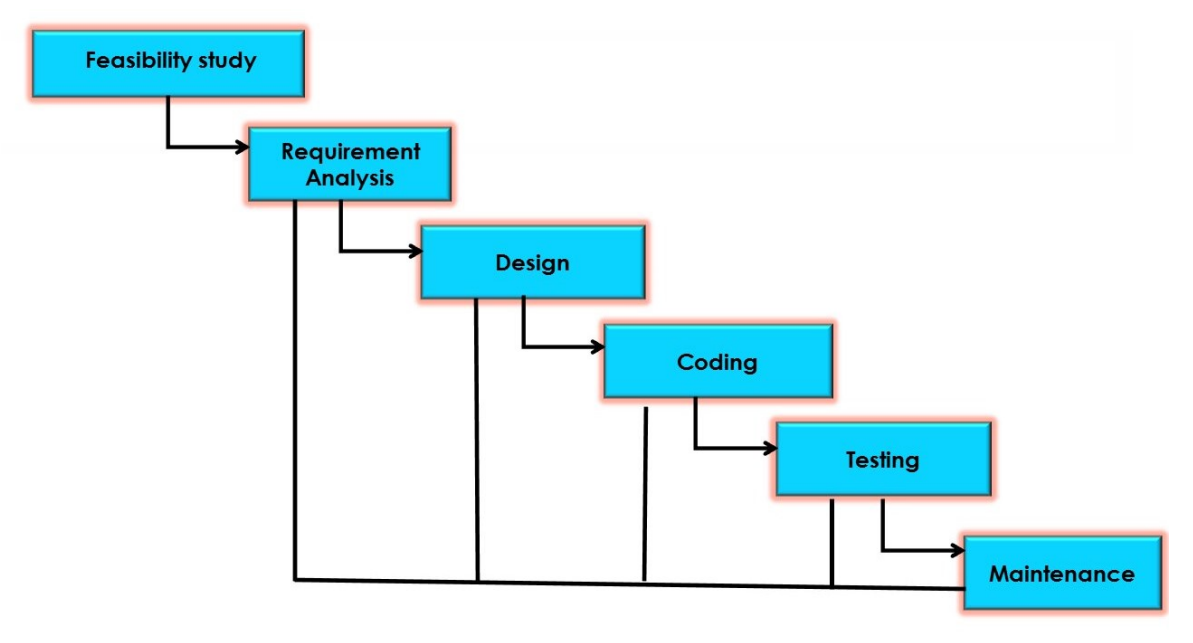
**Introduction**

* This **classical model** was used earlier but now **not** used much.
* Many popular SDLC models are **derived** from **Waterfall** model.



**Iterative Waterfall Model**

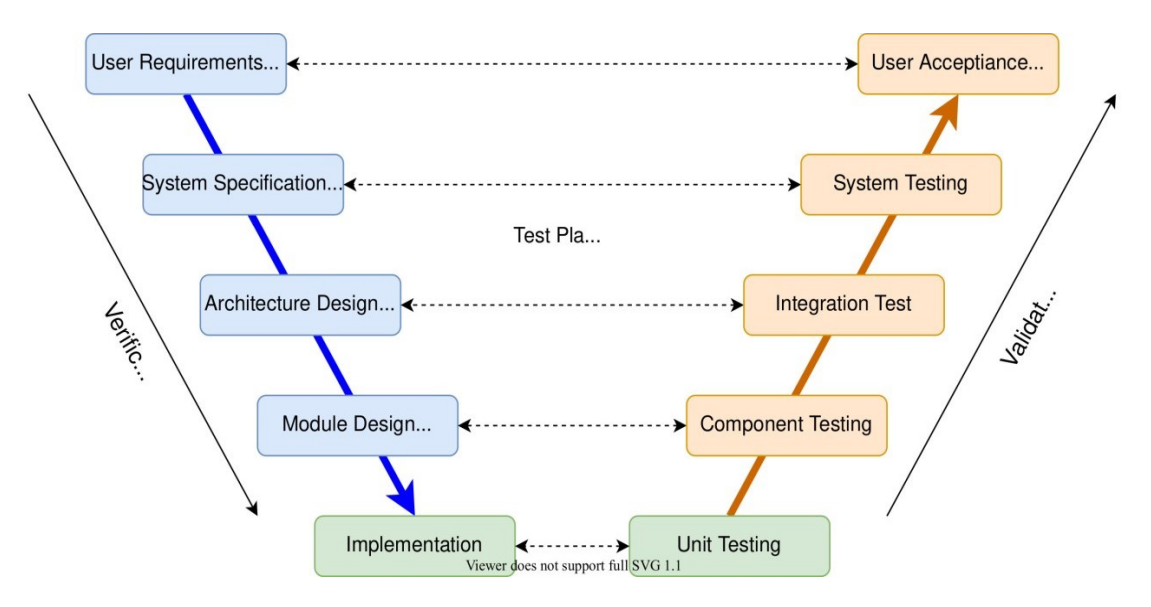
* Same traditional feature but with an extra feature of **iterative design**.
* Every stage provides feedback to its previous phase.



**Topic – 17: V-Model**

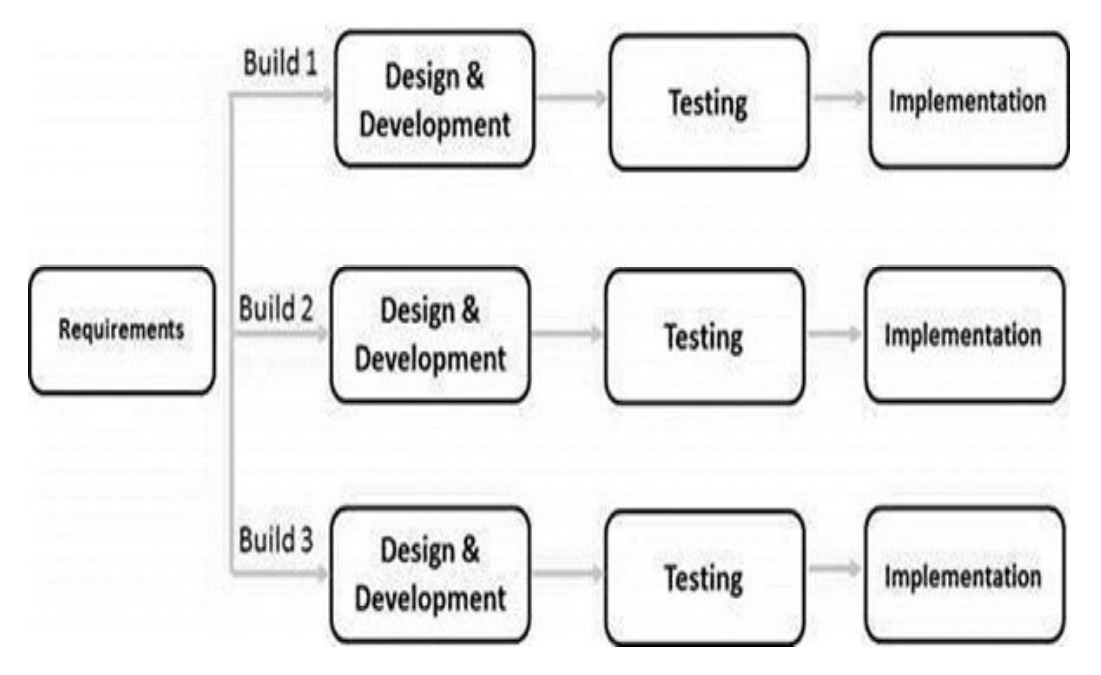
**Introduction**

* **V-Model** stands for **verification & validation** model.
* Widely used these days.
* It is a **sequential model**, meaning that each phase starts only after the previous phase has ended.
* In this model, **test** is conducted **parallel** to and for each of the phase.



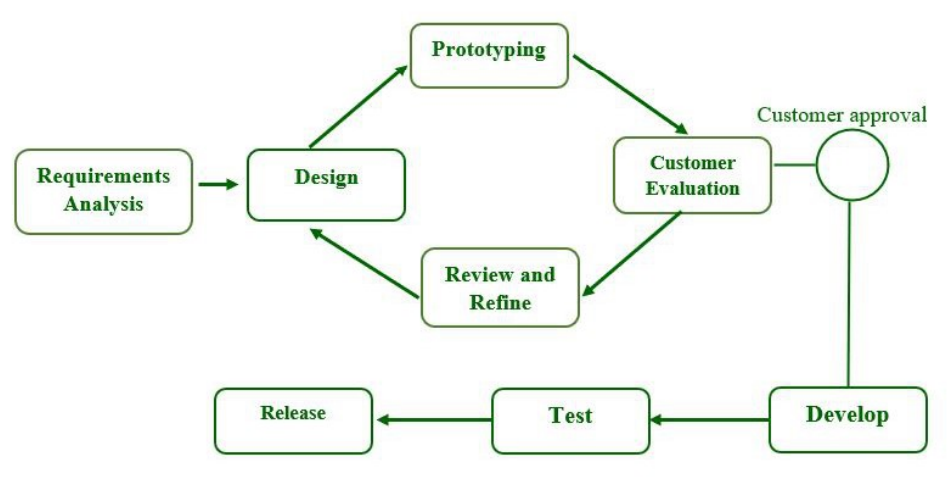
**More About V-Model**

* Instead of making whole software at once, we make rather build a **small portion** of it.
* And then we deliver that portion to customers & ask for their feedback.
* Then we again follow the model & **add features** or **remove shortcomings** as per customer review.
* Then we **again** ask for **feedback** with new build of software.
* It is repeated until the customers find the product meeting their requirements perfectly.



**Topic – 18: Prototype Model**

* Prototype & prototype model are **not** the same thing.
* In this model, first **prototypes** are developed & then final products are manufactured based on that prototype.
* It was primarily developed to **overcome shortcomings** of **waterfall model**.
* We use it when the customer requirements **aren’t** clear to us.
* One speciality about prototype model is that it can be **integrated** into **other models** as well.
* Problem is that it may require **re-development** of model if customers **aren’t** satisfied with a design.

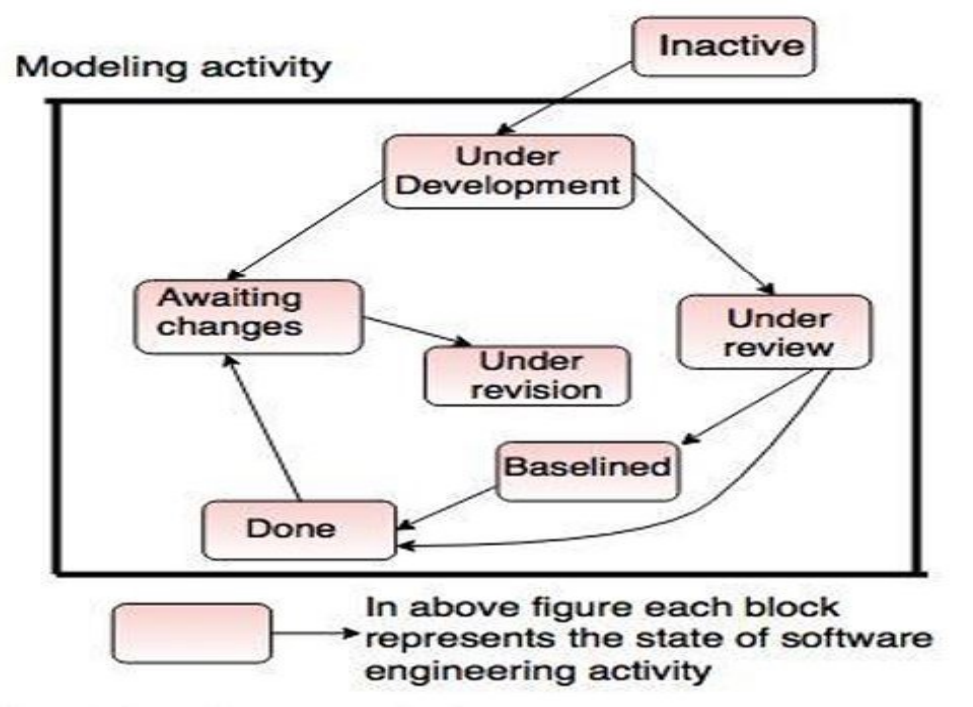


**Topic – 19: Concurrent Model**

**Introduction**

* Also known as ***concurrent development model*** & ***concurrent engineering model***.

**Understanding Model**



* These blocks with tags on them are referring to **various concurrent parts** of this model.
* SDLC activities above includes **requirement**, **design**, **code**, **test** & **deploy**.

**Topic – 20: Scrum Model (Agile)**

* **Scrum** is one of the Agile’s model.
* It introduces roles like **scrum master** & **product owner**, who organise and manage the team of developers working at various phases.
* Speciality about **Scrum** is that its concept is divided into **3 categories**.
* These categories are **roles**, **artifacts** & **time boxes**.
* **Sprint:** Time-boxed periods where one iteration of development is performed.
* Sprint is of around **2 weeks** in general cases.

