### ****PROCESS MODELS****

#### ****Need for Process Models****

A process model provides a structured approach to software development, ensuring consistency, predictability, and quality. The key reasons for adopting a process model include:

1. **Systematic Software Product Development**
   * Ensures a well-defined, step-by-step approach to building software.
   * Reduces errors and improves efficiency by following best practices.
2. **Definiteness and Discipline in the Overall Process**
   * Establishes clear guidelines and protocols for each phase of development.
   * Encourages teamwork and accountability.
3. **Definitive Transition Between Phases**
   * Each phase has a well-defined starting and ending point.
   * The software progresses systematically from one stage to the next.
   * Reduces confusion and overlaps between different stages.
4. **Monitoring Overall Progress of the Software Product**
   * Facilitates tracking of project milestones and deliverables.
   * Helps project managers ensure deadlines and quality standards are met.
   * Identifies risks early, allowing for timely corrective actions.

#### ****Selection Criteria for a Process Model****

Choosing the right process model depends on several factors:

1. **Nature of the Project and Application**
   * Large-scale vs. small-scale development.
   * If it’s complex or needs high security (like banking or healthcare systems), a strict process is better.

* If requirements keep changing, a flexible approach works best.

1. **Methods and Tools Used**
   * Programming languages, frameworks, and development environments influence the process model.
   * Automated testing, CI/CD pipelines, and DevOps practices impact model selection.
2. **Controls (Rules) and Deliverables**
   * Some projects need to follow strict rules (like government or financial software).

* Deliverables can include documents (plans, designs) and actual software features.

#### ****Key Phases in Software Development Process****

1. **Requirement Analysis, and Specification**
   * Understanding user needs and expectations.
   * Gathering functional and non-functional requirements.
   * Documenting requirements in detail to serve as a blueprint for development.
2. **System Design**
   * Architectural and detailed design of the software system.
   * Identifying components, modules, and their interactions.
   * Ensuring scalability, maintainability, and efficiency in design choices.
3. **Program Implementation (Coding & Development)**
   * Writing clean, efficient, and maintainable code.
   * Following coding standards and best practices.
   * Integrating various system components and modules.
4. **Testing**
   * Ensuring the software meets specified requirements.
   * Different levels of testing: Unit Testing, Integration Testing, System Testing, and Acceptance Testing.
   * Detecting and fixing bugs before deployment.

**PRESCRIPTIVE PROCESS MODELS**

**Process model as an abstract representation of process.**

**They define a prescribed set of process elements and a predictable process work flow**

**TheWaterfall Model**

**-Incremental Process Models**

**Incremental Model**

**RAD Model**

**-Evolutionary Process Models**

**Prototyping**

**Spiral**

**Concurrent Development**

**-The Unified Process Model**

**-Concurrent Models 1-57**

****PREDICTIVE vs ADAPTIVE****

****Predictive**(Fixed Plan Approach) : a pretty good understanding of the requirements of the**

**software or the product that you are building.**

**client or the analyst or the customer have a very high confidence of the requirements or what they're looking for,then the team goes through this design implementation and testing phase, or the process that you have**

**learned.**

**Get the product in one shot in the end.**

****Example**: Building a house—everything is planned before construction begins, and changes are difficult to make later.**

****Adaptive models (Flexible Plan approach):** The client or the customer generally has an idea of**

**what they want to build, but not quite there, right?**

**They have an idea but they are not 100% sure what they want to build.**

**The team that is working on it, they actually build something like a really**

**small version of it or like a low fidelity version of it.Then they build the next version based on the feedback.**

**Build a small or basic version of the product (prototype).**

* Show it to the client and **gather feedback**.
* Improve it in the next version, **step by step**.

**Example: Developing a mobile app—you first release a basic version, get user feedback, and keep improving it.**

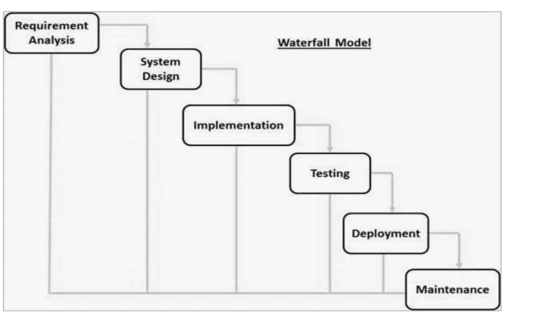
### ****Waterfall Model****

The **Waterfall Model** is one of the earliest methods used in software development. It follows a **step-by-step** approach, meaning you must complete one phase before moving to the next. That’s why it’s also called the **Linear Sequential Model**

#### ****Phases of the Waterfall Model:****

1️⃣ **Requirement Analysis** – Understanding what the software needs to do.  
2️⃣ **Design** – Creating a blueprint (structure) for the software.  
3️⃣ **Implementation** – Writing the actual code and building the software.  
4️⃣ **Verification/Testing** – Checking for errors and ensuring everything works correctly.  
5️⃣ **Maintenance** – Fixing issues, updates, and improvements after the software is delivered.

Since changes are hard to make once a phase is completed, this model is **best suited for projects where all requirements are clear from the start** (e.g., government or defense projects).



### ****When to Use the Waterfall Model?****

✅ **When requirements are well-defined and fixed** – The project should have clear, stable requirements with no expected changes.  
✅ **For small to medium-sized projects** – Works best when the scope is limited and manageable.  
✅ **When the technology and tools are well understood** – If the development team is familiar with the technology stack, the structured approach is effective.  
✅ **For projects with strict documentation and approvals** – Industries like healthcare, aerospace, and government projects require a well-documented approach.  
✅ **When quality and reliability are more important than speed** – Since testing is done after the development phase, it ensures a highly reliable final product.  
✅ **When no early releases are needed** – If the client is okay with waiting until the entire software is complete, the Waterfall Model is a good choice.

****Advantages of Waterfall Model****

**- It is very simple to understand and easy to use.**

**- Phases of waterfall model do not overlap with each other.**

**- It is useful for small projects in which requirements are clear at the beginning.**

**- Since development is linear it is easy to manage development process.**

****Disadvantages of Waterfall Model****

**It is not useful for large projects.**

**Not suitable for projects in which requirement are not clear initially.**

**Product is available only at the end of development process.**

**It is very difficult to modify system requirement in the middle of development process.**

### ****Incremental Process Model****

The **Incremental Model** is a step-by-step approach to software development. Instead of building the entire software at once, it is divided into **smaller parts (modules)** that are developed and tested one by one.

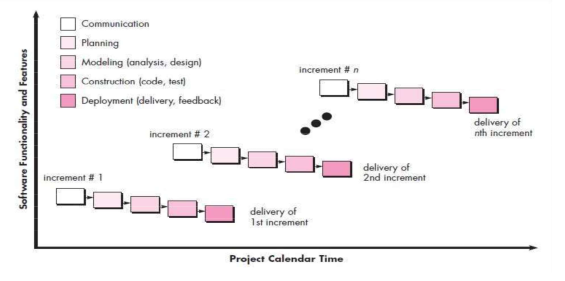
#### ****Key Features:****

✅ The software is built **in multiple stages** instead of all at once.  
✅ Each stage (module) goes through **Requirement Analysis, Design, Implementation, and Testing**.  
✅ **Every new module adds more features** to the previous version.  
✅ The process **continues until the full system is developed**.  
✅ This approach is also used in **Agile Development**.

#### ****How It Works:****

1️⃣ **First module** → Developed, tested, and delivered.  
2️⃣ **Next module** → Built on top of the previous one, adding more features.  
3️⃣ **Repeat** until the entire software is complete.

This model is useful when **requirements are not fully clear** at the start, as it allows for gradual improvements and user feedback.



****When we use the 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.**

****Advantage of Incremental Model****

**-Errors are easy to be recognized.**

**-Easier to test and debug**

**-More flexible.**

**-Simple to manage risk because it handled during its iteration.**

**-The Client gets important functionality early.**

****Disadvantage of Incremental Model****

**-Need for good planning**

**-Total Cost is high.**

**-Well defined module interfaces are needed.**

### ****Evolutionary Process Models****

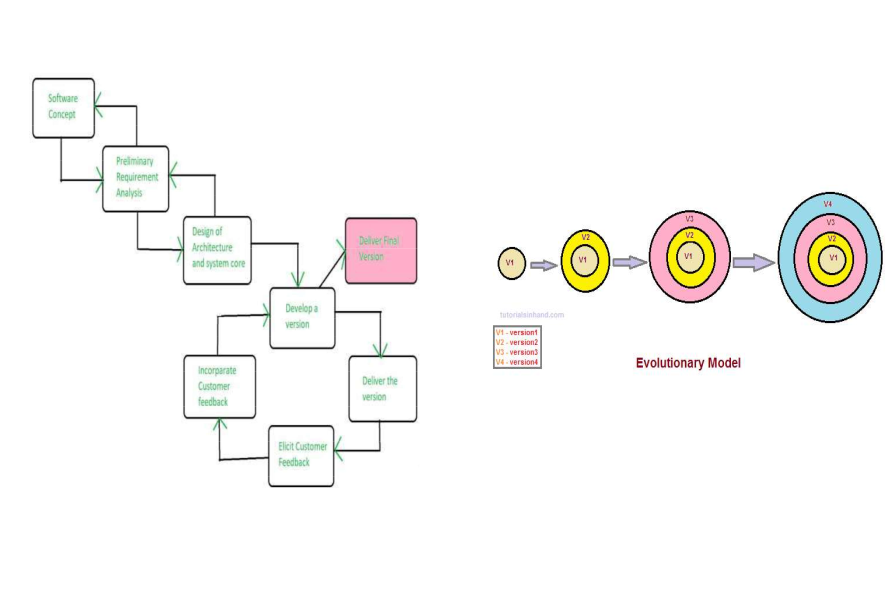
The **Evolutionary Model** is a software development approach where the product is built **step by step** and improved over time. It allows for **continuous updates and changes** based on business needs and market demands.

#### ****Key Features:****

✅ **Iterative and Incremental** – The software is developed in small parts, tested, and improved repeatedly.  
✅ **Flexible to Changes** – Since businesses and technology evolve, this model allows for **continuous improvements**.  
✅ **Customer Feedback** – Each module is delivered to the customer, who gives feedback to improve the next version.  
✅ **Gradual Development** – Instead of building everything at once, the software is developed **module by module**.

#### ****Types of Evolutionary Models:****

1️⃣ **Prototyping Model** – A basic version (**prototype**) is created first, tested, and refined based on customer feedback before making the final product.  
2️⃣ **Spiral Model** – Combines planning, risk analysis, development, and testing in **multiple cycles**, ensuring continuous refinement and improvement.



****When to use evolutionary process model****

**This is commonly used when the customer wants to starts using the core features instead of waiting for the full software**

**It is used in large projects because of step by step development**

**Customer requirement are not fixed. But clear concept.**

**Small changes required in separate modules.**

**Useful in object oriented software development because all the development is divided into different units.**

****Advantages of Evolutionary model****

**It is useful for very large project.**

**User has checked every stage during the development and it is helpful in achieving customer confidence.**

**User gets a chance to experiment with a partially developed software much before the complete version of the system is released.**

**It reduces the error because the core modules get tested thoroughly**

**It reduces the cost of development**

****Disadvantages of Evolutionary model****

**The delivery of full software can be late due to different changes by**

**customers during development.**

**Constant and clear interaction between the development team**

**members is required.**

### ****Spiral Model (Simplified Explanation)****

The **Spiral Model**, introduced by **Boehm in 1986**, is a software development approach that focuses on **risk management**. It combines features of the **Waterfall, Iterative, and Prototyping models** to develop software in multiple cycles (spirals).

#### ****Key Features:****

✅ **Risk-Driven** – Identifies and reduces risks at each stage.  
✅ **Iterative Process** – Software is built in steps, improving with each cycle.  
✅ **Prototyping Approach** – A working model is created and refined based on feedback.  
✅ **Combines Multiple Models** – Uses the best parts of **Waterfall, Iterative, and Prototyping models**.

#### ****How It Works:****

1️⃣ **Planning** – Define objectives, risks, and resources.  
2️⃣ **Risk Analysis** – Identify potential risks and find ways to solve them.  
3️⃣ **Development & Testing** – Build a version of the software and test it.  
4️⃣ **Evaluation & Refinement** – Gather feedback and improve in the next cycle.

🔄 **Repeat the process** until the final product is complete.

#### ****Where It’s Used?****

* **Microsoft** – Software like Windows undergoes multiple iterations.
* **Gaming Industry** – Games are built and tested in multiple stages before final release.

Because of its flexibility and risk management approach, the Spiral Model is also called a **Meta Model**.

****When to use Spiral model?****

**When the project is large and high budget.**

**When requirement are unclear and complex.**

**When changes may require at any time**

**When the software needs continuous risk evaluation.**

****Advantages of Spiral Model****

**High amount of risk analysis.**

**Risky part can be developed earlier which help in better risk management.**

**Useful for large and mission-critical project.**

**Allows extensive use of prototype(solve all error in prototype).**

**There is always a space for customer feedback.**

**Changing the requirement can be accommodated.**

**Development is fast**

****Disadvantages of Spiral Model****

**Risk analysis needed highly particular expertise.**

**Can be a costly model to use.**

**Doesn t work for smaller projects.**

**Spiral process is complex sometimes because spiral may be infinitely.**

**Large number of spiral stages required excessive documentation.**