



Unit-2

Requirement Analysis and Engineering



Requirement Determination

- ❖ In software engineering, requirement determination is the process of identifying, gathering, and defining what the software must do. This phase is crucial because it forms the foundation upon which the entire software development process is built. A well-defined requirement determination process ensures that the software meets the needs of users, is delivered on time, and within budget.

Major Activities in requirement Determination



Requirement collection

- The process of collecting requirements from stakeholders, including end users, customers, and other relevant parties.

Requirement Analysis

- Analyzing and refining the gathered requirements to ensure they are clear, complete, and feasible.

Requirement Specification

- Documenting the requirements in a clear and structured format so that they can be easily understood and followed by the development team.

System Requirement Specification or Software Requirement Specification (SRS)

- ❖ It contains a complete information description, a detailed functional description, a representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria, and other information to requirements.
- ❖ Software requirement specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it.
- ❖ The basic goal of the requirement phase is to produce the SRS, Which describes the complete behavior of the proposed software.
- ❖ SRS is also helping the clients to understand their own needs.

Fact Finding Techniques

- The specific methods for finding information of the system are termed as fact finding techniques.
 - Interview, Questionnaire, Record View and Observations are the different fact finding techniques.
1. **Interview** : This method is used to collect the information from groups or individuals. The information collected is quite accurate and reliable.
 2. **Questionnaire** : The Questionnaire consists of series of questions framed together in logical manner. The questions are simple, clear and to the point. This method is very useful for obtaining information from people who are concerned with the usage of the system.



Fact Finding Techniques

- 3. Record View :** The information related to the system is published in the sources like newspapers, magazines, journals, documents etc. This record review helps the analyst to get valuable information about the system and the organization.
- 4. Observations :** In this method the analyst himself visits the organization and observes and understand the flow of documents, working of the existing system, the users of the system etc.



Process Models

- A software process model is the mechanism of dividing software development work into distinct phases to improve design, product management, and project management.
- It is also known as a software development life cycle.
- Process modeling is the graphical representation of business processes or workflows.

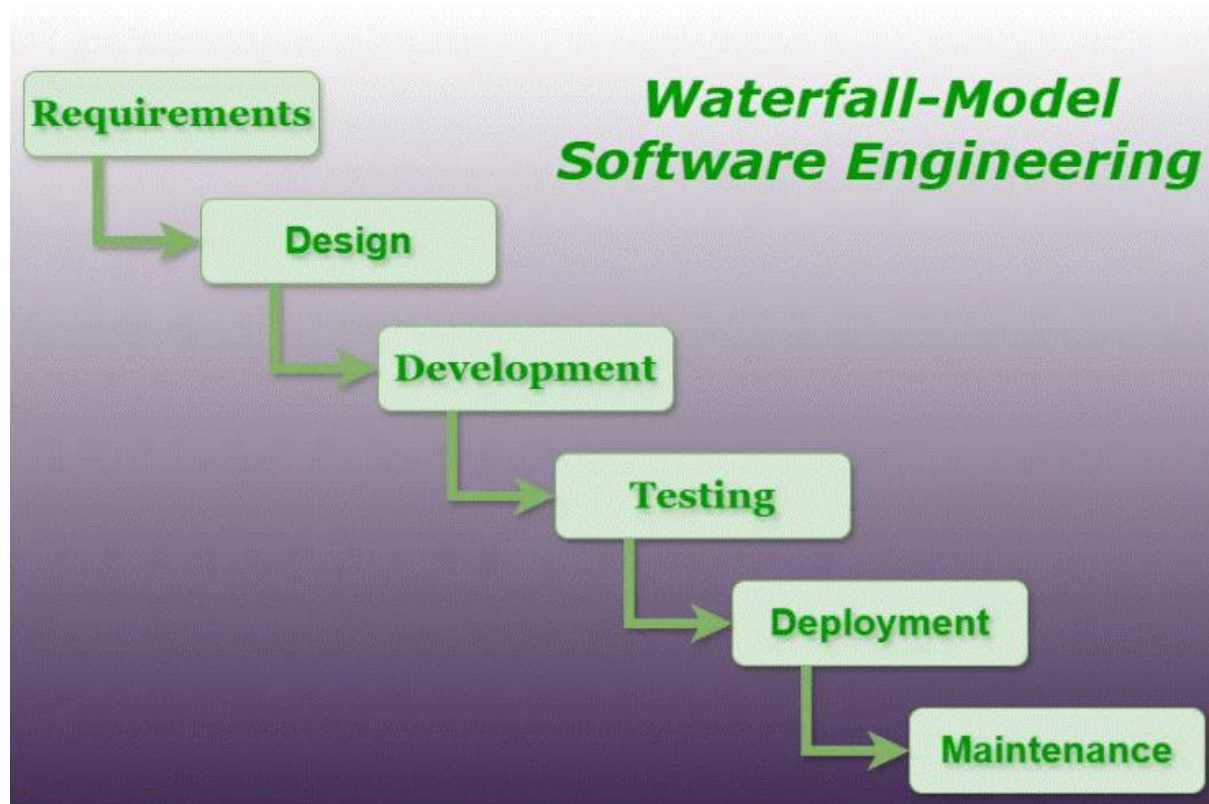


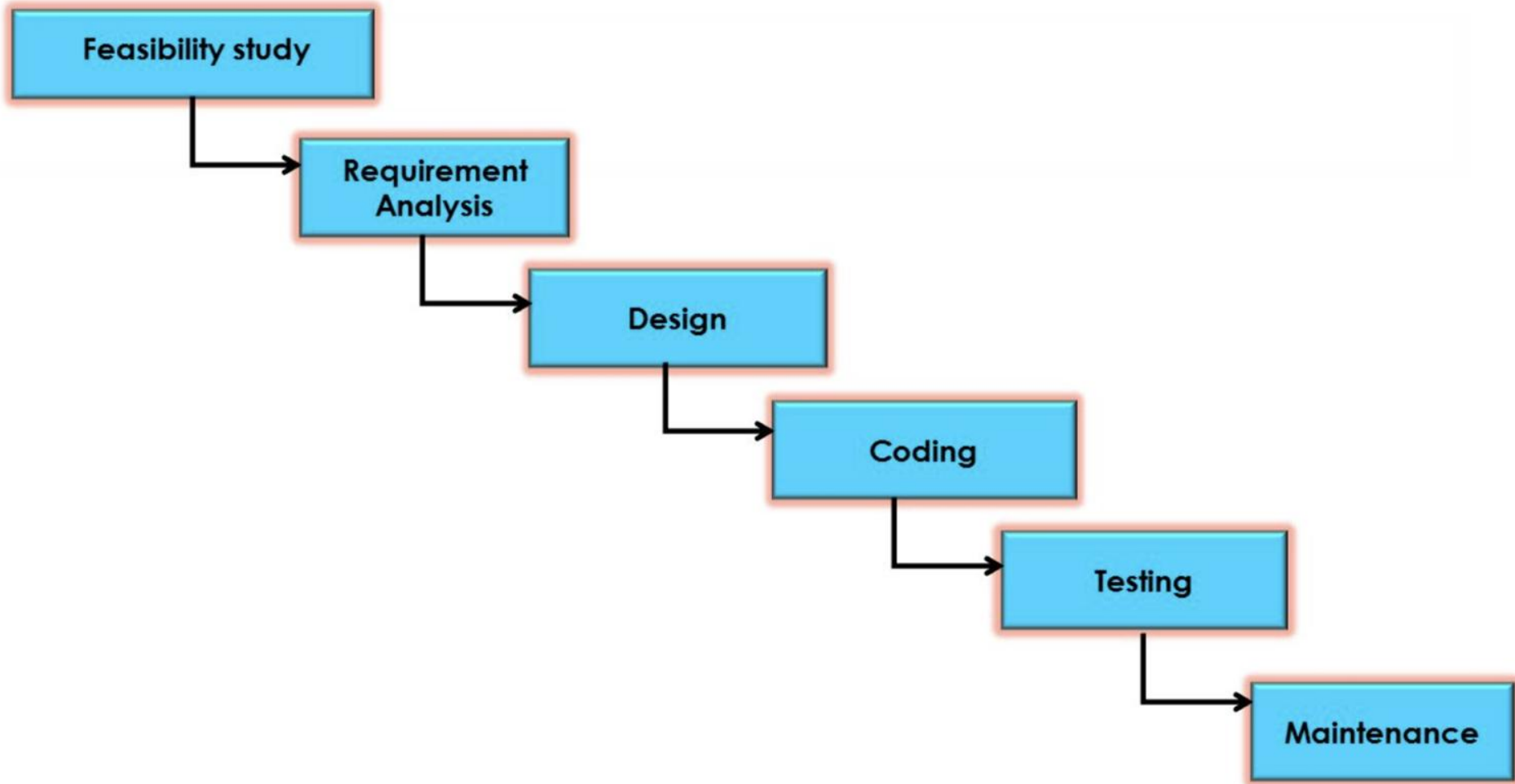
Here are some of the most commonly used software process models:

1. Waterfall Model

2. Incremental Model

Waterfall Model







What is the SDLC Waterfall Model?

The waterfall model is a software development model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to project management and software development.

- This model is used when,
 - ❖ Requirements are very well known, clear and fixed.
 - ❖ Product definition is stable.
 - ❖ Technology is understood.
 - ❖ There are no ambiguous (unclear) requirements.
 - ❖ Ample (sufficient) resources with required expertise are available freely.
 - ❖ The project is short.



Features of the SDLC Waterfall Model

1. **Sequential Approach:** The waterfall model involves a sequential approach to software development, where each phase of the project is completed before moving on to the next one.
2. **Document-Driven:** The waterfall model relies heavily on documentation to ensure that the project is well-defined and the project team is working towards a clear set of goals.
3. **Quality Control:** The waterfall model places a high emphasis on quality control and testing at each phase of the project, to ensure that the final product meets the requirements and expectations of the stakeholders.



Phases of SDLC Waterfall Model – Design

1. Feasibility Study:

The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software. The feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem. These different identified solutions are analyzed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.



Phases of SDLC Waterfall Model – Design

2. Requirements Analysis and Specification: The requirement analysis and specification phase aims to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.

- **Requirement gathering and analysis:** Firstly all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (an inconsistent requirement is one in which some part of the requirement contradicts some other part).
- **Requirement specification:** These analyzed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document.



Phases of SDLC Waterfall Model – Design

3. Design:

The goal of this phase is to convert the requirements acquired in the SRS into a format that can be coded in a programming language. It includes high-level and detailed design as well as the overall software architecture. A Software Design Document is used to document all of this effort (SDD).



Phases of SDLC Waterfall Model – Design

4. Coding and Unit Testing:

In the coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The unit testing phase aims to check whether each module is working properly or not.



Phases of SDLC Waterfall Model – Design

5. Integration and System testing: Integration of different modules is undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over several steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested, the full working system is obtained and system testing is carried out on this.

System testing consists of three different kinds of testing activities as described below.

- **Alpha testing:** Alpha testing is the system testing performed by the development team.
- **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.
- **Acceptance testing:** After the software has been delivered, the customer performs acceptance testing to determine whether to accept the delivered software or reject it.



Phases of SDLC Waterfall Model – Design

5. Deployment:

Once the software has been tested and approved, it is deployed to the production environment.



Phases of SDLC Waterfall Model – Design

6. Maintenance: Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are three types of maintenance.

- **Corrective Maintenance:** This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
- **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
- **Adaptive Maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment such as working on a new computer platform or with a new operating system.



Importance of SDLC Waterfall Model

1. **Clarity and Simplicity:** The linear form of the Waterfall Model offers a simple and unambiguous foundation for project development.
2. **Clearly Defined Phases:** The Waterfall Model phases each have unique inputs and outputs, guaranteeing a planned development with obvious checkpoints.
3. **Documentation:** A focus on thorough documentation helps with software comprehension, upkeep, and future growth.
4. **Stability in Requirements:** Suitable for projects when the requirements are clear and steady, reducing modifications as the project progresses.
5. **Resource Optimization:** It encourages effective task-focused work without continuously changing contexts by allocating resources according to project phases.
6. **Relevance for Small Projects:** Economical for modest projects with simple specifications and minimal complexity.

Advantages of the SDLC Waterfall Model

- **Easy to Understand:** The Classical Waterfall Model is very simple and easy to understand.
- **Individual Processing:** Phases in the Classical Waterfall model are processed one at a time.
- **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
- **Clear Milestones:** The classical Waterfall model has very clear and well-understood milestones.
- **Properly Documented:** Processes, actions, and results are very well documented.
- **Reinforces Good Habits:** The Classical Waterfall Model reinforces good habits like define-before-design and design-before-code.
- **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.

Disadvantages of the SDLC Waterfall Model

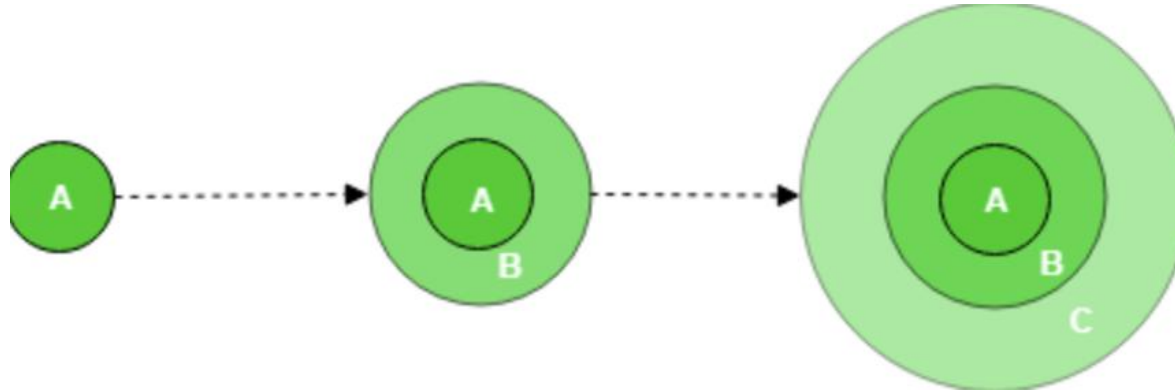
- **No Feedback Path:** In the classical waterfall model evolution of software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phase. Therefore, it does not incorporate any mechanism for error correction.
- **Difficult to accommodate Change Requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but the customer's requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
- **No Overlapping of Phases:** This model recommends that a new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase efficiency and reduce cost, phases may overlap.


Disadvantages of the SDLC Waterfall Model

- **Limited Flexibility:** The Waterfall Model is a rigid and linear approach to software development, which means that it is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed, it is difficult to make changes or go back to a previous phase.
- **Limited Stakeholder Involvement:** The Waterfall Model is a structured and sequential approach, which means that stakeholders are typically involved in the early phases of the project (requirements gathering and analysis) but may not be involved in the later phases (implementation, testing, and deployment).
- **Late Defect Detection:** In the Waterfall Model, testing is typically done toward the end of the development process. This means that defects may not be discovered until late in the development process, which can be expensive and time-consuming to fix.

What is the Incremental Process Model?

First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.





Phases of Incremental model

Requirement
Analysis

Design
& Development

Testing

Implementation



Phases of incremental model

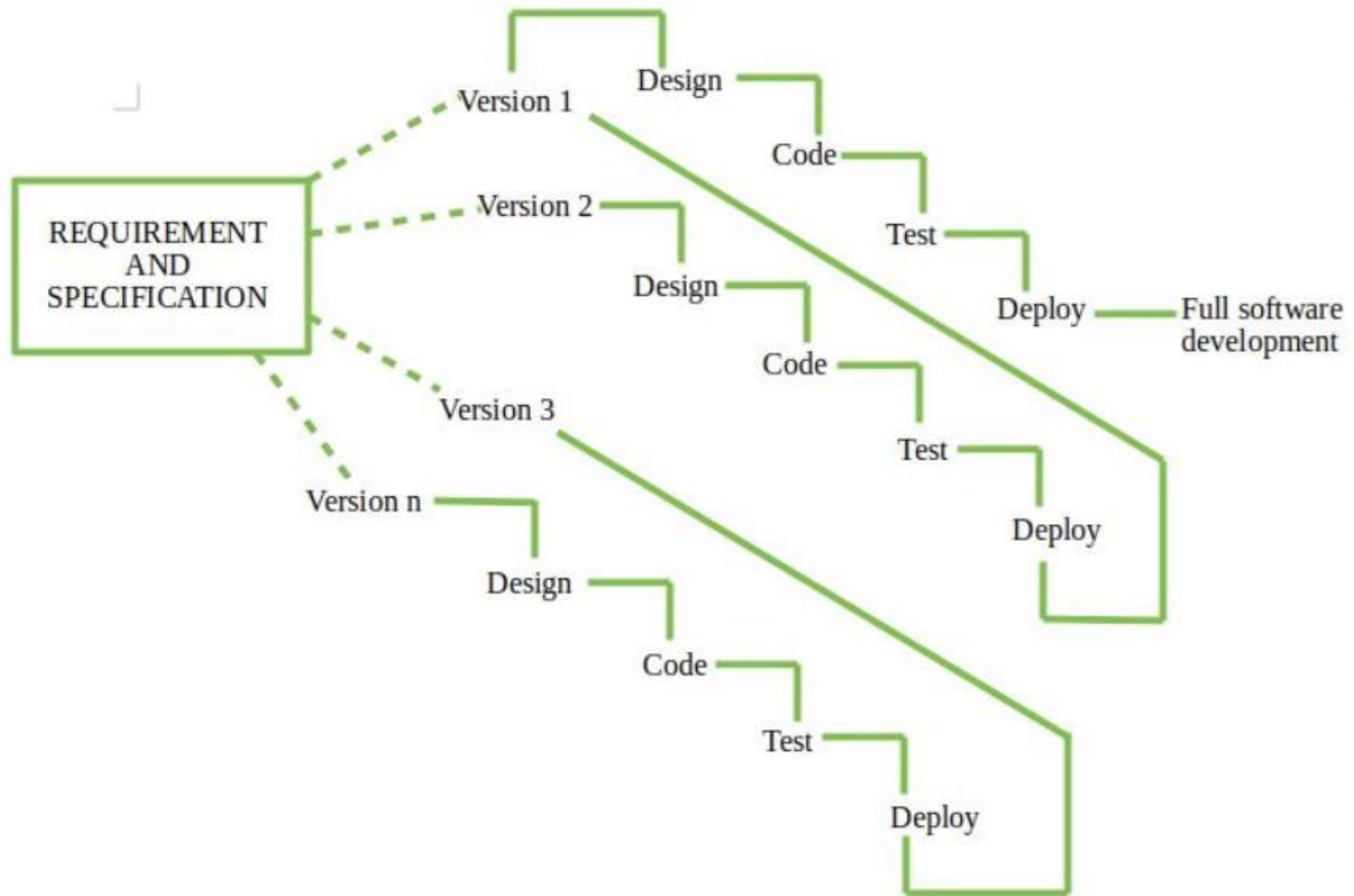
1. **Requirement analysis:** In Requirement Analysis At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer.
2. **Design & Development:** At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the needs of the customer. The Development Team first undertakes to develop core features (these do not need services from other features) of the system. Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.



Phases of incremental model

3. Deployment and Testing: After Requirements gathering and specification, requirements are then split into several different versions starting with version 1, in each successive increment, the next version is constructed and then deployed at the customer site. in development and Testing the product is checked and tested for the actual process of the model.

4. Implementation: In implementation After the last version (version n), it is now deployed at the client site.



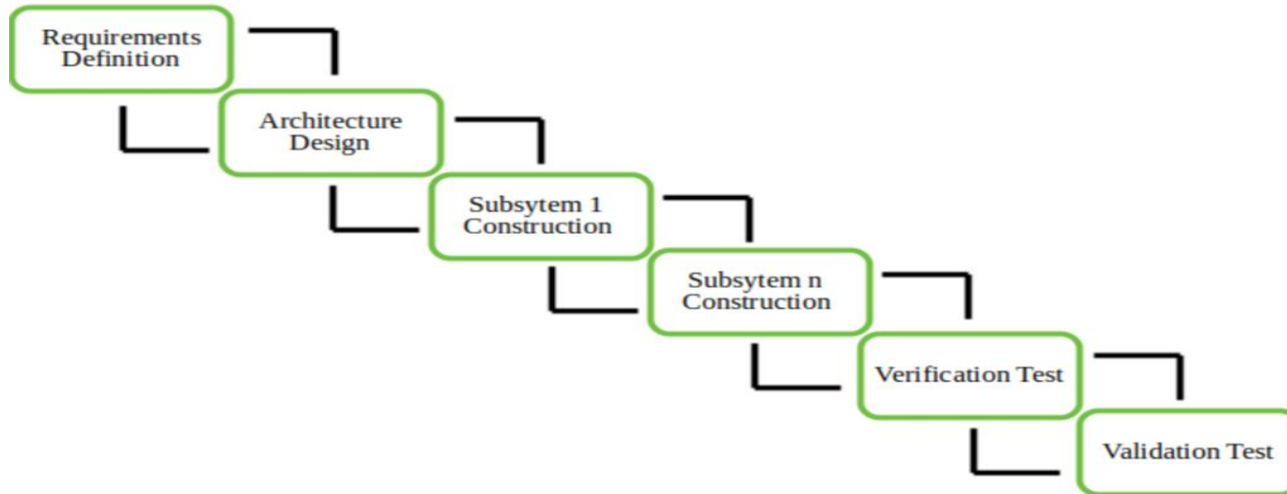


Types of Incremental Model

1. Staged Delivery Model
2. Parallel Development Model

1. Staged Delivery Model

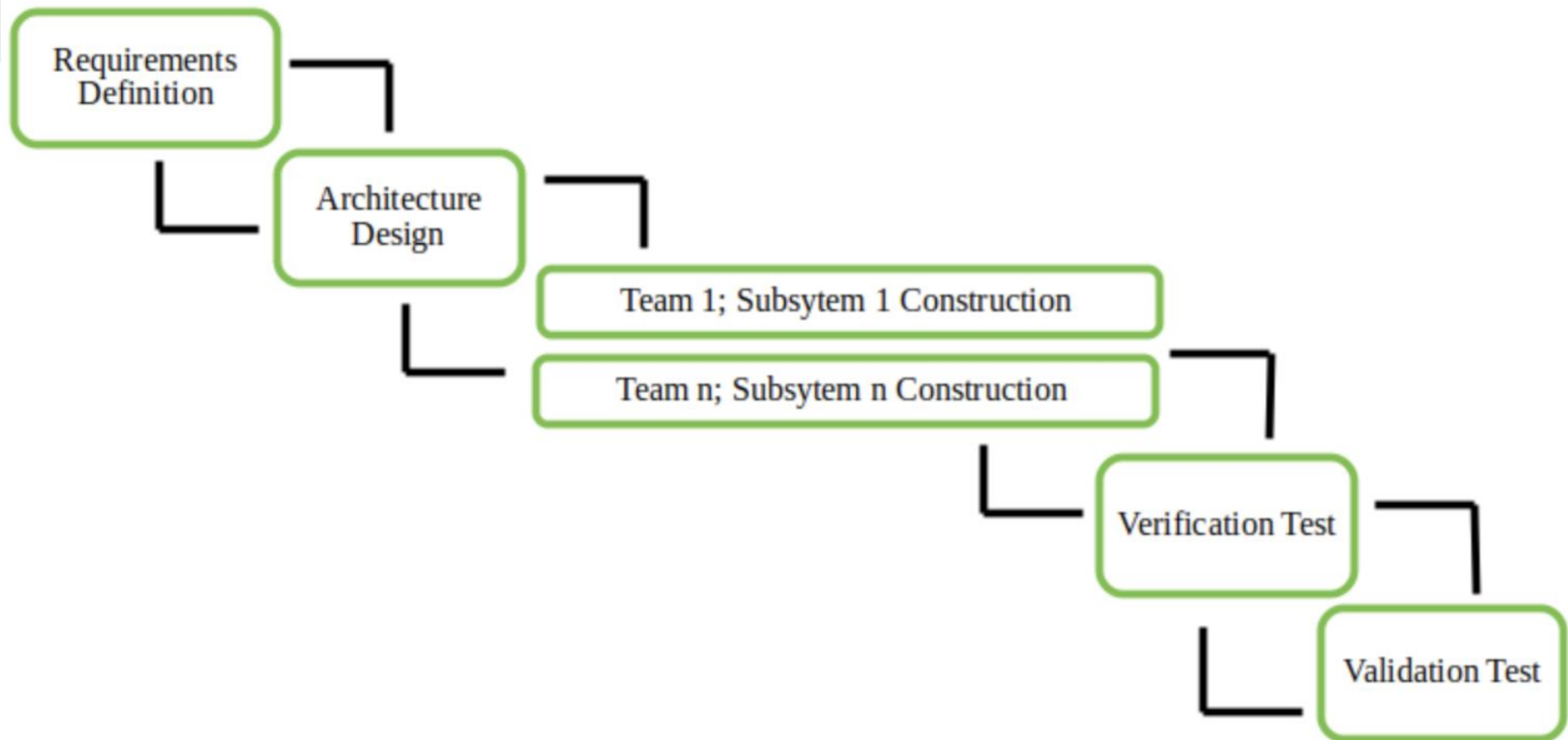
Construction of only one part of the project at a time





2. Parallel Development Model

Different subsystems are developed at the same time. It can decrease the calendar time needed for the development, i.e. TTM (Time to Market) if enough resources are available.





Evolutionary Process Model

- When a set of core product or system requirements is well understood but the details of product or system extensions have yet to be defined.
- In this situation there is a need of process model which specially designed to accommodate product that evolve with time.
- Evolutionary Process Models are specially meant for that which produce an increasingly more complete version of the software with each iteration.



Evolutionary Process Model

- Evolutionary Models are iterative.
- They are characterized in a manner that enable to develop increasingly more complete versions of the software.
- Evolutionary Models are :
 1. Prototyping Model
 2. Spiral Model
 3. Concurrent Model



1. Prototyping Model

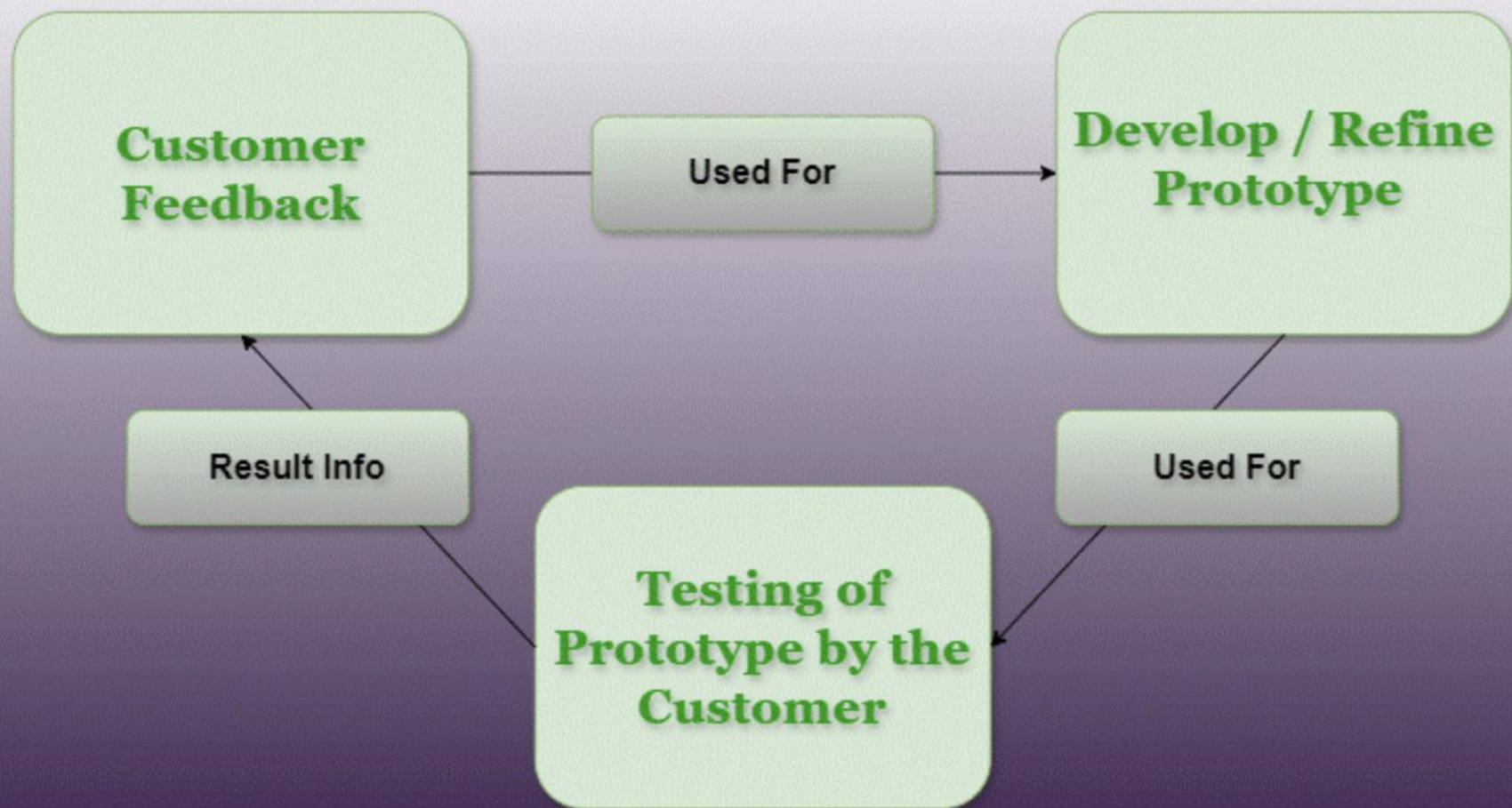
The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models). This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.



When to use Prototyping Model

- It is used when customers have general objectives of software but do not have detailed requirements for functions and features.
- Also is used when developers are not sure about efficiency of an algorithm and technical feasibilities.
- It serves as a mechanism for identifying software requirements.

Prototyping Model-Concept






Steps of Prototyping Model

Step 1: Requirement Gathering and Analysis: This is the initial step in designing a prototype model. In this phase, users are asked about what they expect or what they want from the system.

Step 2: Quick Design: This is the second step in the Prototyping Model. This model covers the basic design of the requirement through which a quick overview can be easily described.

Step 3: Build a Prototype: This step helps in building an actual prototype from the knowledge gained from prototype design.

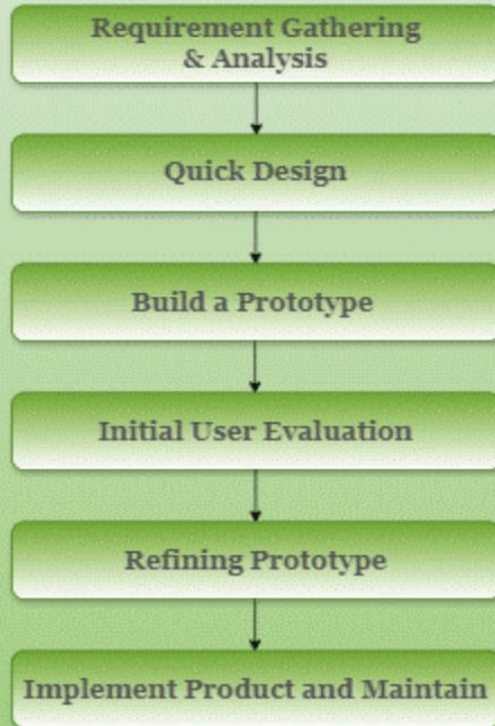


Step 4: Initial User Evaluation: This step describes the pre testing where the investigation of the performance model occurs, as the customer will tell the strengths and weaknesses of the design, which was sent to the developer.

Step 5: Refining Prototype: If any feedback is given by the user, then improving the client's response to feedback and suggestions, the final system is approved.

Step 6: Implement Product and Maintain: This is the final step in the phase of the Prototyping Model where the final system is tested and distributed to production, here the program is run regularly to prevent failures.

Prototyping Model





Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily added as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.

Disadvantages of the Prototyping Model



- Costly concerning time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.



Spiral Model

The spiral model is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model.

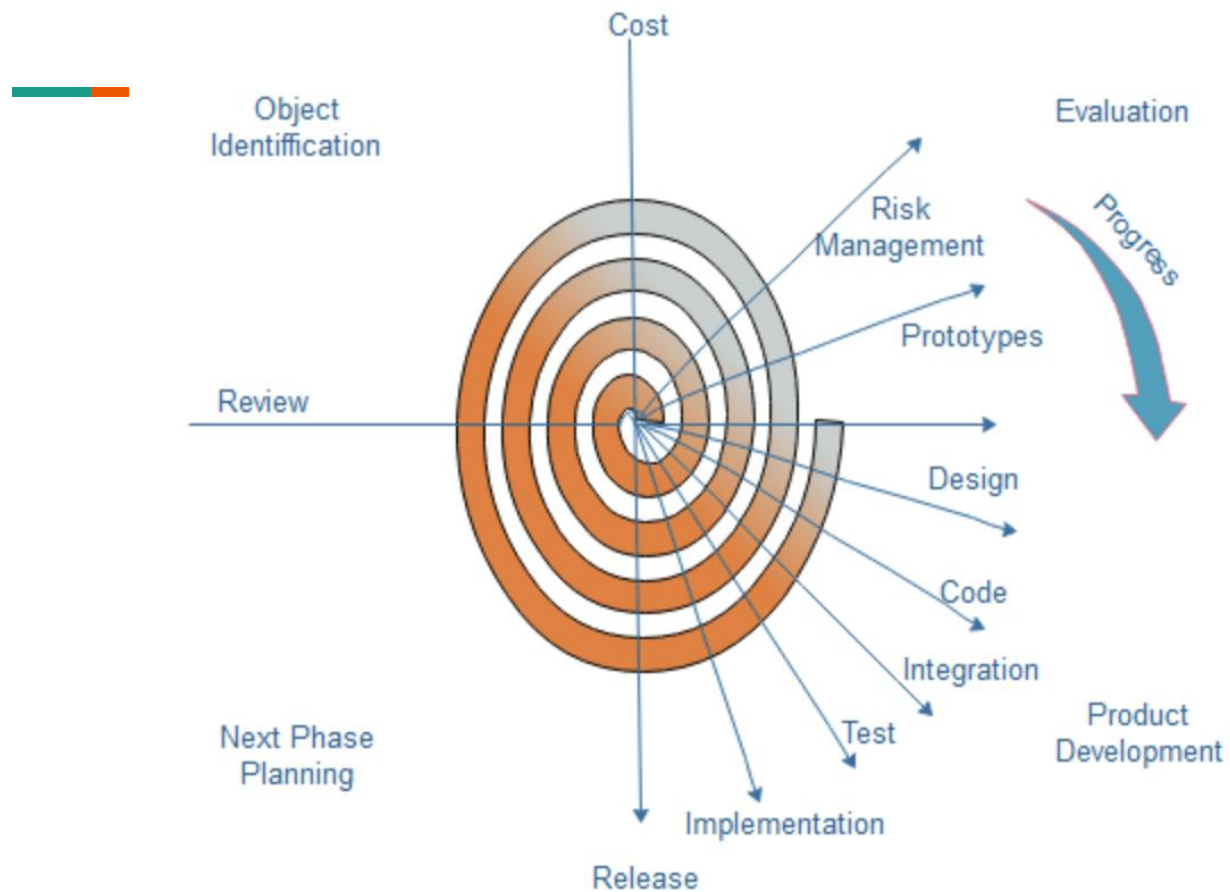
It implements the potential for rapid development of new versions of the software.

Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.



Spiral Model

- It can be used when;
 - For the development of large scale / high-risk projects.
 - When costs and risk evaluation is important.
 - Users are unsure of their needs.
 - Requirements are complex.
 - New product line.
 - Significant (considerable) changes are expected.






Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategy that resolve risks. This process may include activity such as paper model and prototyping.



Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.



When to use Spiral Model?

- When delivery is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects



Advantages

- High amount of risk analysis
- Useful for large and mission-critical projects.



Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.

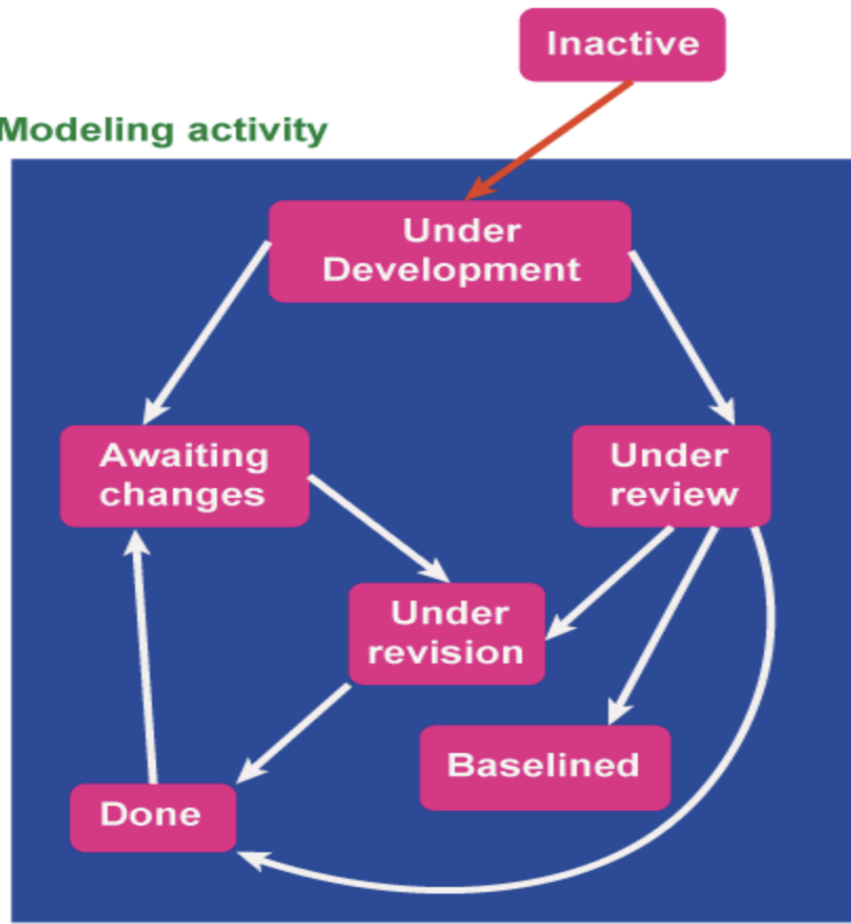
Aspect	Prototype Model	Spiral Model
Purpose	To create a working model that users can interact with to refine requirements and features.	To manage risk and ensure a systematic development process through iterative refinement.
Approach	Iterative development with an emphasis on user feedback through a prototype.	Combines iterative development with systematic risk management.
Phases	<ol style="list-style-type: none"> 1. Requirement Gathering 2. Quick Design 3. Prototype Building 4. User Evaluation 5. Refinement 6. Final System Development 	<ol style="list-style-type: none"> 1. Identification of Objectives 2. Risk Analysis 3. Development and Testing 4. Planning the Next Iteration
User Involvement	High - Users interact with the prototype and provide feedback throughout the process.	Moderate to High - Users are involved in reviewing iterations and managing risks.
Risk Management	Minimal focus on risk management; the primary focus is on refining requirements through prototyping.	Central to the model, with risk assessment and mitigation at every phase.



3. Concurrent Development Model

The **Evolutionary Concurrent Development Model** is a software development approach that combines the principles of evolutionary development with concurrent engineering. It allows for the simultaneous progression of different aspects of the software project, emphasizing iterative development, feedback loops, and flexibility. This model is particularly effective in complex projects where requirements are expected to evolve over time, and different subsystems or components can be developed in parallel.

Modeling activity



In above figure each block represents the state of software engineering activity



Advantages of the Concurrent Development Model

There are so many benefits of using the concurrent development model. These are as follows:

- We can use this model in all types of software development processes.
- It is so much easier to understand.
- It also provides immediate feedback after the testing process is done.
- It also provides an accurate picture of the state of the project.



Disadvantages of the Concurrent Development Model

Using the concurrent development model also has some disadvantages. These are as follows.

- We have to improve communication between the team members.
- It requires remembering the status of the different activities.




Feasibility Study

- Feasibility Study can be considered as pre investigation that helps the management to take decision about whether study of system should be feasible for development or not.
 - It identifies the possibility of improving an existing system.
 - It is used to obtain the outline of the problem and decide whether feasible or appropriate solution exists or not.
 - The main objective of a feasibility study is to acquire problem scope instead of solving the problem.
 - The output of a feasibility study is a formal system proposal act as decision document.



Types of Feasibilities

- **Economic Feasibility** : The main aim of Economic Feasibility Analysis (EFS) is to estimate the economic requirements of candidate system before investments funds are committed to proposal.
- **Technical Feasibility** : It analyzes and determines whether the solution can be supported by existing technology or not.

- 
- **Operational Feasibility** : It determines whether the system is operating effectively once it is developed and implemented.
 - **Behavioral Feasibility** : It helps in determining if the system requires special effort to educate, retrain, transfer, and changes in employee's job status on new ways of conducting business.
 - **Schedule Feasibility** : It ensures that the project should be completed within given time constraint or schedule.



Problem recognition

- The main aim of requirement analysis is to fully understand main objective of requirement that includes why it is needed, does it add value to product, will it be beneficial, does it increase quality of the project, does it will have any other effect.
- All these points are fully recognized in problem recognition so that requirements that are essential can be fulfilled to solve business problems.



Requirement Engineering

- Requirement Engineering means that requirements for a product are defined, managed and tested systematically.
- Requirements engineering builds a bridge to design and construction.
- Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution, validating the specification, and managing the requirements as they are transformed into an operational system.

Requirement Engineering tasks

- **Inception**—ask a set of questions that establish ...
 - basic understanding of the problem
 - the people who want a solution
 - the nature of the solution that is desired, and
 - the effectiveness of communication and collaboration between the customer and the developer
- **Elicitation**—gathering requirements from all stakeholders
- **Elaboration**—create an analysis model that identifies data, function and behavioral requirements
- **Negotiation**—agree on a deliverable system that is realistic for developers and customers

Requirement Engineering tasks

- **Specification**—can be any one (or more) of the following:
 - A written document
 - A set of models
 - A collection of user scenarios (use-cases)
 - A prototype
- **Validation**—a review mechanism that looks for
 - errors in content
 - areas where clarification may be required
 - missing information
 - inconsistency (a major problem when large products or systems are engineered)
 - conflicting or unrealistic (unachievable) requirements.

Requirement Engineering tasks

Requirements management

- **Objective:** To manage changes to requirements throughout the project lifecycle.
- **Activities:**
 - Tracking requirements as they evolve during development.
 - Managing requirement changes through a formal change control process.
 - Maintaining traceability between requirements and other project artifacts (e.g., design, code, test cases).
 - Ensuring that all stakeholders are informed of changes and their impacts.

Requirements validation

- Requirements validation examines the specification to ensure that all software requirements have been stated unambiguously; that means all inconsistencies, omissions, and errors have been detected and corrected; and the work products conform to the standards established for the process, the project, and the product.
- The primary requirements validation mechanism is the technical review.

Requirements validation

- The review team that validates requirements includes software engineers, customers, users, and other stakeholders who examine the specification looking for errors in content or interpretation, areas where clarification may be required, missing information, inconsistencies (a major problem when large products or systems are engineered), conflicting requirements, or unrealistic (unachievable) requirements.