# Software Engineering

## Lecture 3

## (SDLC & Software Process Model)

# Software Process

- A **software process** (also knows as software methodology) is a set of related activities that leads to the production of the software. These activities may involve the development of the software from the scratch, or, modifying an existing system.

- Software development life cycle (**SDLC**) is a series of phases that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs.

# Software Process

- Any software process must include the following four activities:

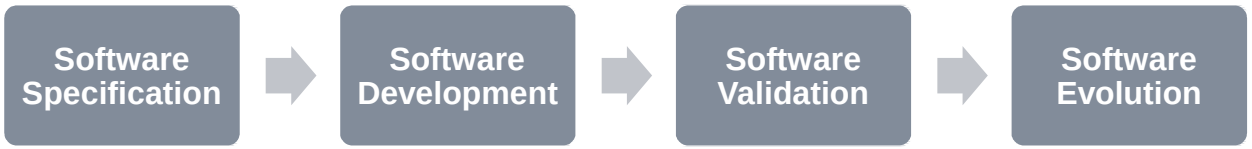



**Figure : Software Development Activities/Processes**

# SDP Activities/Processes – The Breakdown

- **Software Specification:** This activity involves stating the functionalities (operations) as well as the constraints/limitations of the software
  - Major Activity -> Requirements
- **Software Design and Development:** This activity involves developing (producing) a software according to the specification outlined in the first stage of the development process.
  - Major Activity -> Design and Implementation

# SDP Activities/Processes – The Breakdown

- **Software Validation:** This activity checks the software developed to ensure it conforms to the specification earlier concluded upon.
  - Major Activity -> Testing
- **Software Evolution:** This activity ensures that the software produced is modifiable/adjustable to meet customer and market requirements changes.
  - Major Activity -> Maintenance

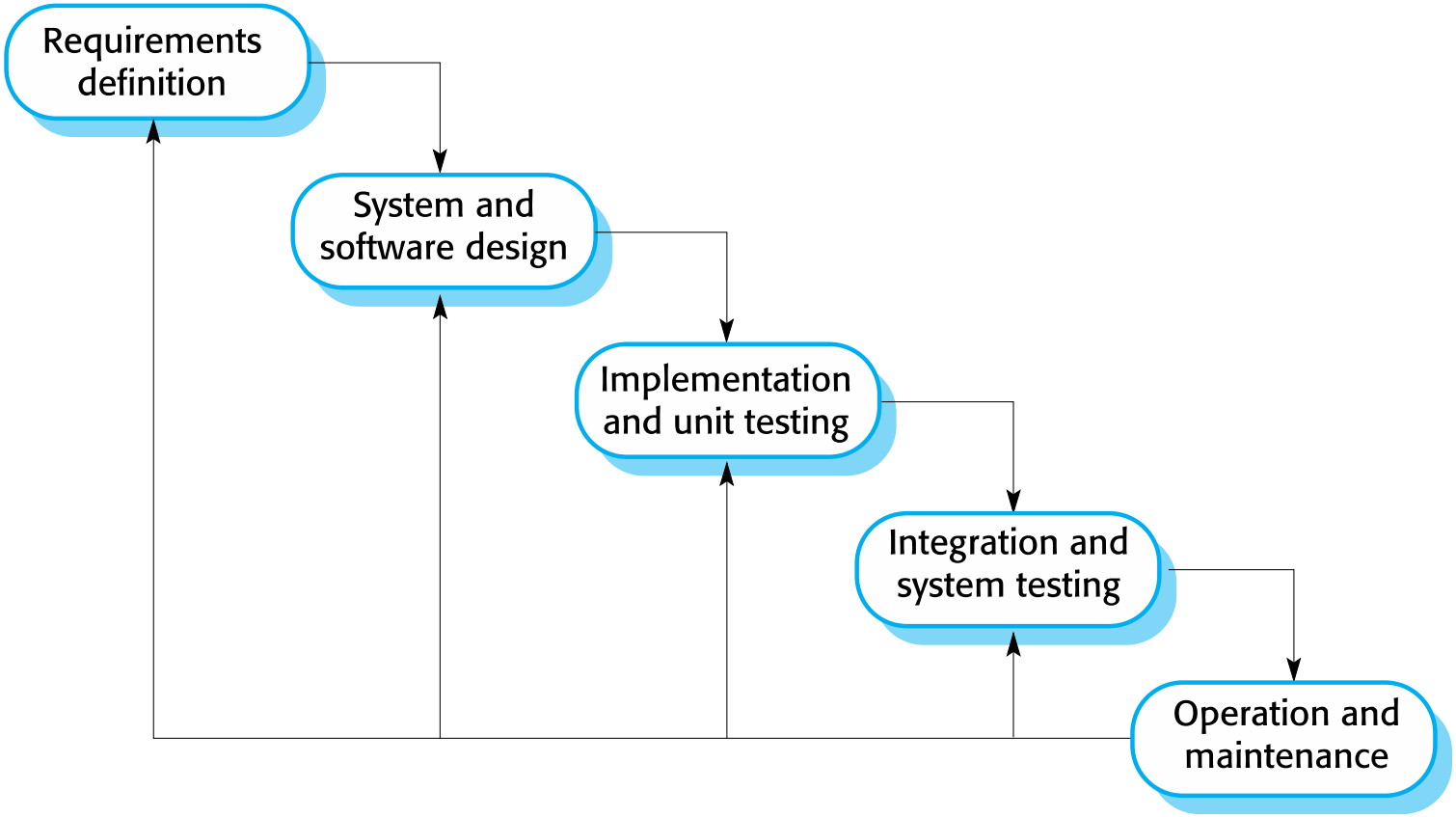
# SDLC Models

- This refers to the existing and established frameworks for developing software.
- A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

# Waterfall Model

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
  - In the waterfall model, you must plan and schedule all of the activities before starting working on them (plan-driven process).
- The reason why this method titled Waterfall is because this method follows a sequential process in software development process. The processes in this methodology are descending or downward such as Waterfall.
- Waterfall's phases are Requirements, Analysis, Design, Construction, Testing, Implementation and Maintenance.

# Waterfall Model



# Waterfall Model

- It is the most well-known model and it is very simple and easy to understand. It is also called a linear model. Here, each phase begins only after the previous phase is finished and there is no overlapping in these phases.
- This model can applicable to projects where:
  - The requirements are precisely documented
  - Product definition is stable
  - The technologies stack is predefined which makes it not dynamic
  - No ambiguous requirements
  - The project is short
- This model cannot be apply to the projects where the requirements are not clearly defined and changed by time.

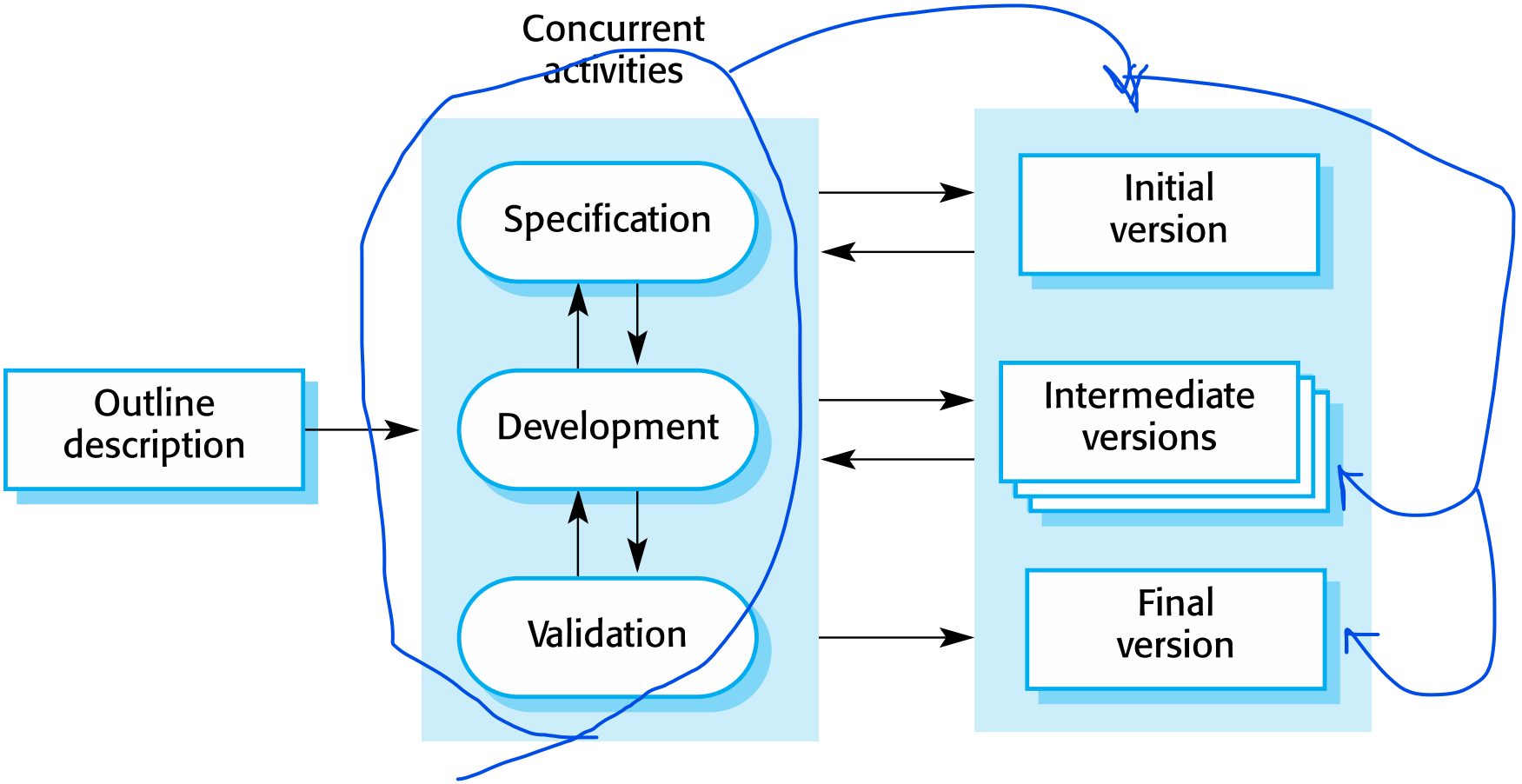
# Waterfall Model

| ADVANTAGES | DISADVANTAGES |
| --- | --- |
| Simple to use and understand | The software is ready only after the last stage is over |
| Management simplicity thanks to its rigidity: every phase has a defined result and process review | High risks and uncertainty |
| Development stages go one by one | Not the best choice for complex and object-oriented projects |
| Perfect for the small or mid-sized projects where requirements are clear and not equivocal | Inappropriate for the long-term projects |
| Easy to determine the key points in the development cycle | The progress of the stage is hard to measure while it is still in the development |
| Easy to classify and prioritize tasks | Integration is done at the very end, which does not give the option of identifying the problem in advance |

# Incremental development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user feedback, and evolving it through several versions until an acceptable system has been developed.
- The activities of a process are not separated but interleaved with feedback involved across those activities.

# Incremental development



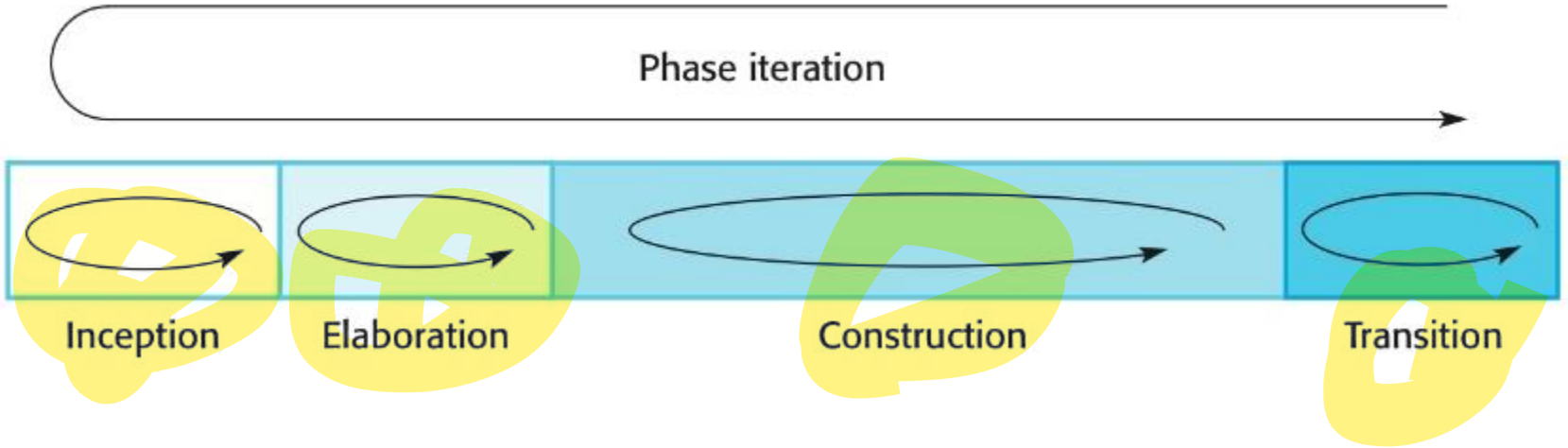
# Incremental development

- **When to use**
  - Large projects where requirements are not understood
  - Working in unfamiliar technical arenas
  - Changing the requirement due to rapidly changing technology
  - Business user can be moderately to heavily engaged
  - Need functional requirements turned into something tangible quickly

# Iterative development

- Iterative development model aims to develop a system through building small portions of all the features, across all components.
- We build a product which meets the initial scope and release it quickly for customer feedback. An early version with limited features important to establish market and get customer feedback.
- In each increment, a slice of system features is delivered, passing through the requirements till the deployment.

# Iterative development



# Phases

- Inception Phase
  - Determining the scope and purpose/vision, feasibility of the project
- Elaboration Phase
  - Capturing the requirements
  - Determining the structure of the system
- Construction Phase
  - Building the system
- Transition Phase
  - Product installation and hand over the system

What customer is dreaming

# Phases

- **Inception:** Customer communication, project vision and planning activities (feasibility study)
- **Elaboration:** multiple iterations that refines the requirements and models of the system
- **Construction:** develop software code
- **Transition:** user testing and installation
- **Production:** operation

# Inception Stage

- Perform feasibility study …..
- Identify the project vision (vision document)
- Identify general business requirements
- Identify project and business risks
- Produce initial use-case model (10-20%)
- Plan the elaboration stages
- Rough architecture of the software (subsystems)

# Elaboration Stage

- An iterative process where refinements are made on system requirements, system design, develop part of the code and test it.
- Products from these iterations:
  - Refinements on use-case model
  - Software architecture description
  - Executable prototypes
  - Initial design model
  - Refinement on project risks and plan

# Construction Stage

- Translate the design into software components
- Products of this stage are:
  - Design model
  - Integrated software components
  - Test plan and test cases
  - User documentation

Docs , not SRS

# Transition Stage

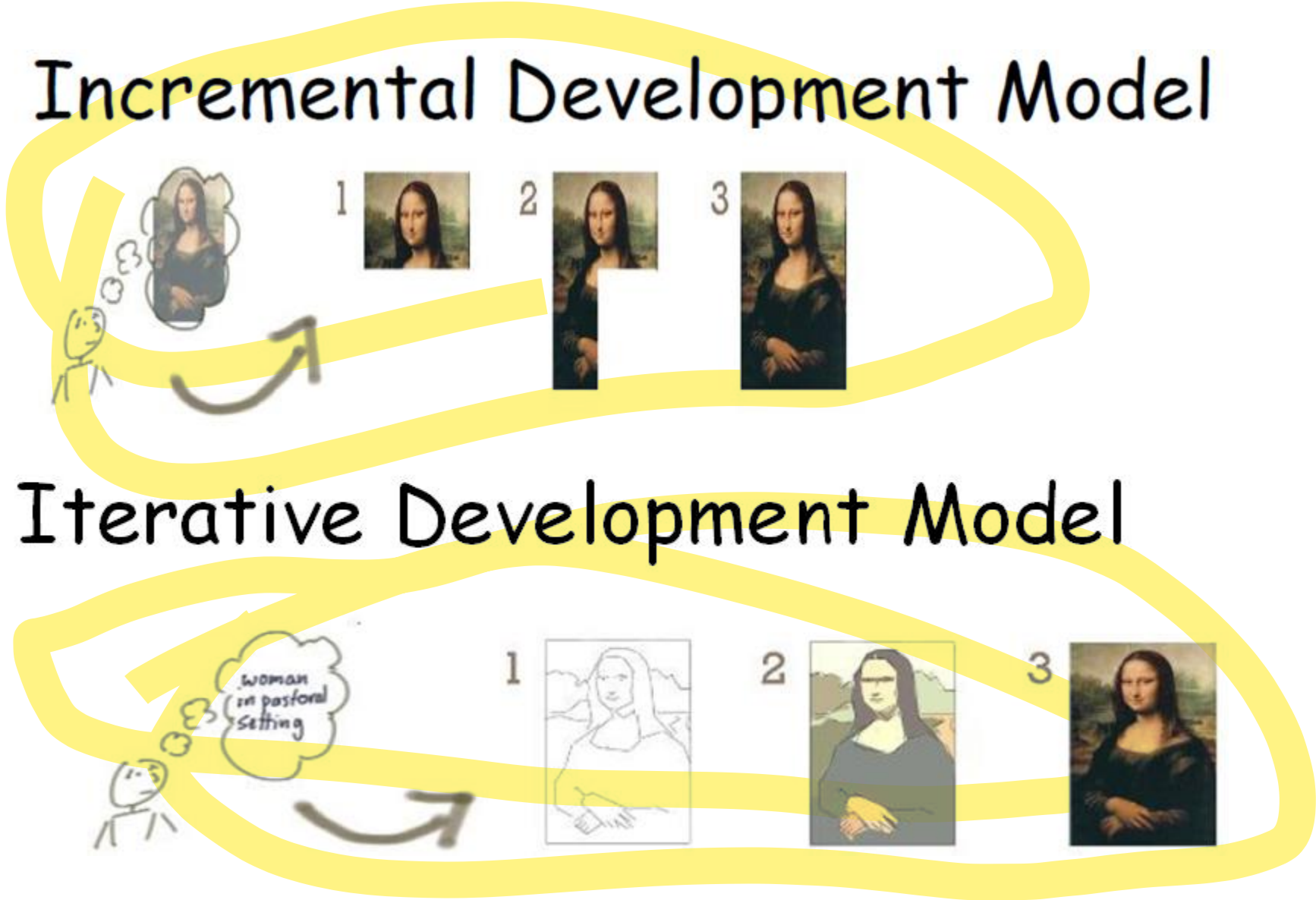
- Deliver the software and documentation
- Get user feedback from Beta tests

# Incremental vs. Iterative

## ❑ Product development example

- Take the context of product development by a software team as an example. The team might develop a small increment of working, yet unrefined, functionality and then iteratively refine that over time. They will then add more features until the functionality was considered satisfactory. For example, in a team's first iteration, a website payment engine might be coded to only allow payment with debit cards (a first increment). The second iteration might produce an increment that supports payment by credit card. Finally, the third iteration might add an increment allowing payment via PayPal.

# Incremental vs. Iterative



# Prototyping

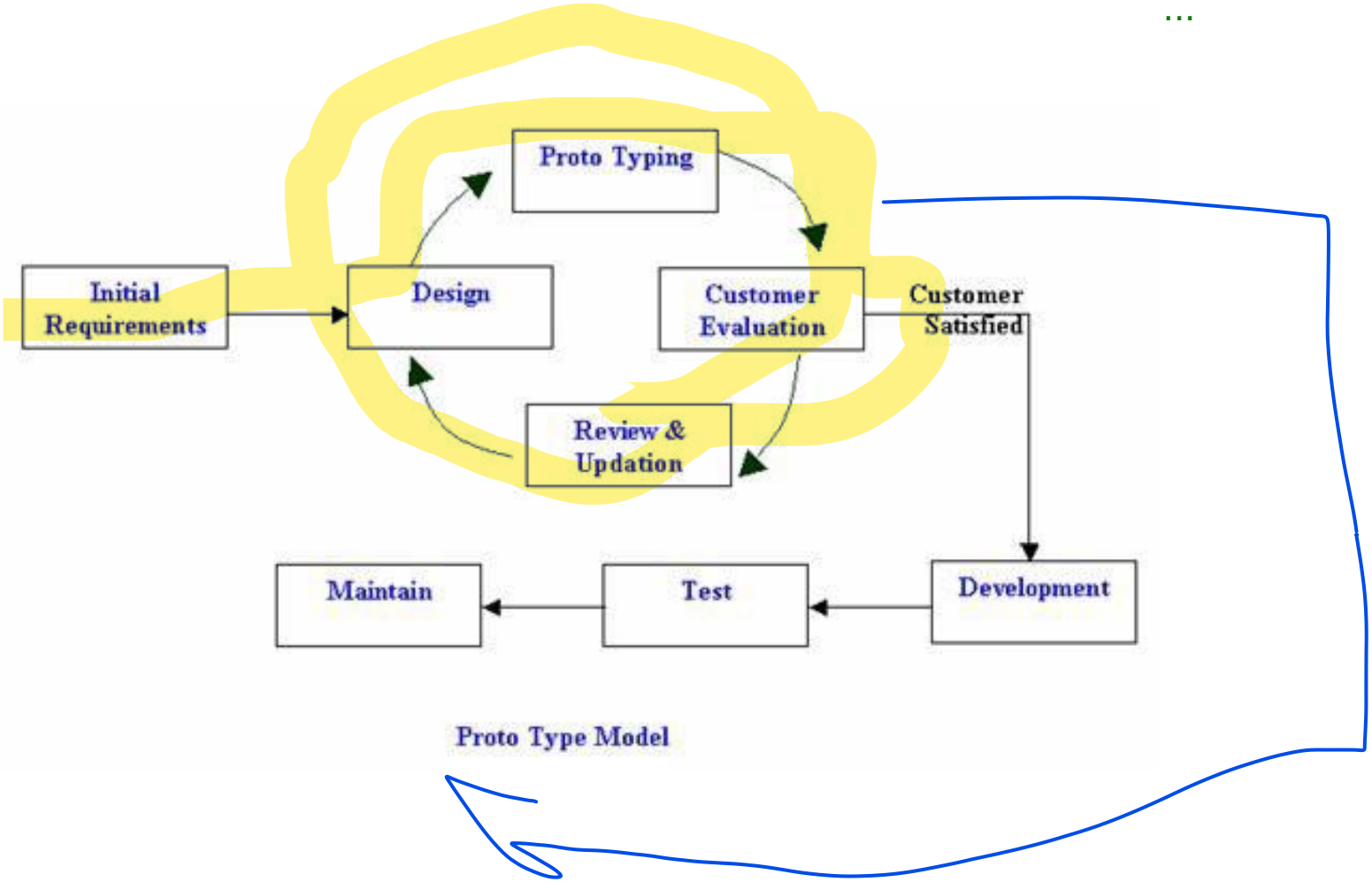
- A prototype is a version of a system or part of the system that's developed quickly to check the customer's requirements or feasibility of some design decisions.
- So, a prototype is useful when a customer or developer is not sure of the requirements, or of algorithms, efficiency, business rules, response time, etc.
- In prototyping, the client is involved throughout the development process, which increases the likelihood of client acceptance of the final implementation.

# Software prototyping

- A prototype is an **initial version** of a system used to demonstrate concepts and try out design options.
- A prototype can be used in:
  - The requirements engineering process to help with requirements elicitation and validation;
  - In design processes to explore options and develop a UI design;
  - In the testing process to run back-to-back tests.

# Software prototyping

Demo1
Demo2
...
...



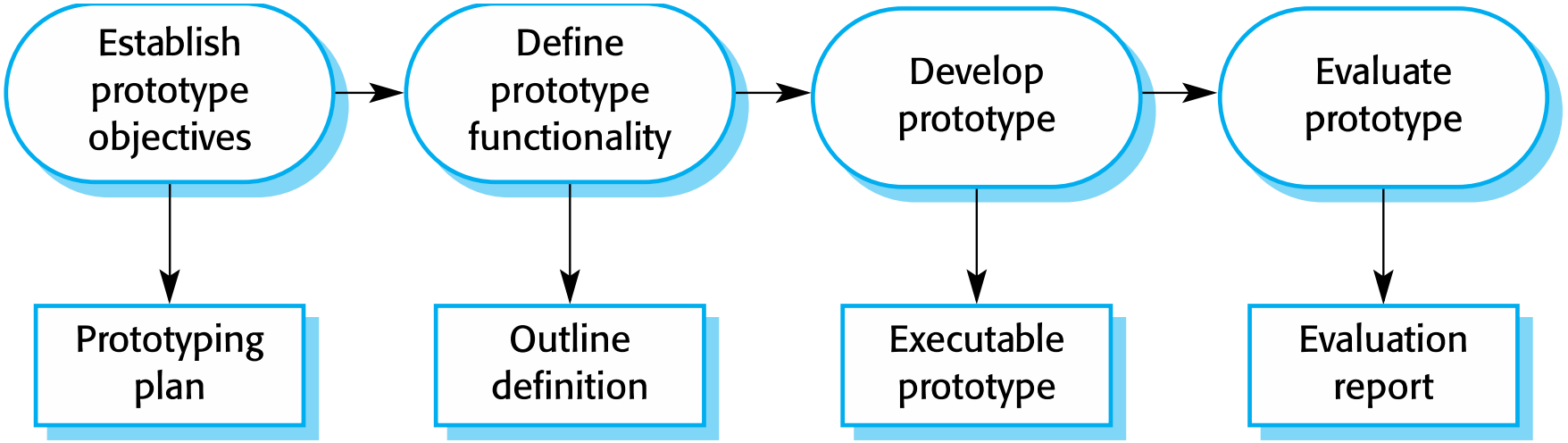
# Software prototyping

- **When to use**
  - Project objectives are unclear
  - High pressure to implement something
  - Frequent requirement changes
  - Minimal resource constrains
  - No strict approval processes needed
  - Innovative, flexible design that need to accommodate future changes

# Benefits of prototyping

- Improved system usability.
- A closer match to users' real needs.
- Improved design quality.
- Improved maintainability.
- Reduced development effort.

# The process of prototype development



# Prototype development

- The phases of a prototype are:
- **Establish objectives**: The objectives of the prototype should be made explicit from the start of the process. Is it to validate system requirements, or demonstrate feasibility, etc.
- **Define prototype functionality**: Decide what are the inputs and the expected output from a prototype. To reduce the prototyping costs and accelerate the delivery schedule, you may ignore some functionality, such as response time and memory utilization unless they are relevant to the objective of the prototype.
- **Develop the prototype**: The initial prototype is developed that includes only user interfaces.
- **Evaluate the prototype**: Once the users are trained to use the prototype, they then discover requirements errors. Using the feedback both the specifications and the prototype can be improved. If changes are introduced, then a repeat of steps 3 and 4 may be needed.

# Prototype development

- May be based on rapid prototyping languages or tools
- May involve leaving out functionality
  - Prototype should focus on areas of the product that are not well-understood;
  - Error checking and recovery may not be included in the prototype;
  - Focus on functional rather than non-functional requirements such as reliability and security

# Spiral Model

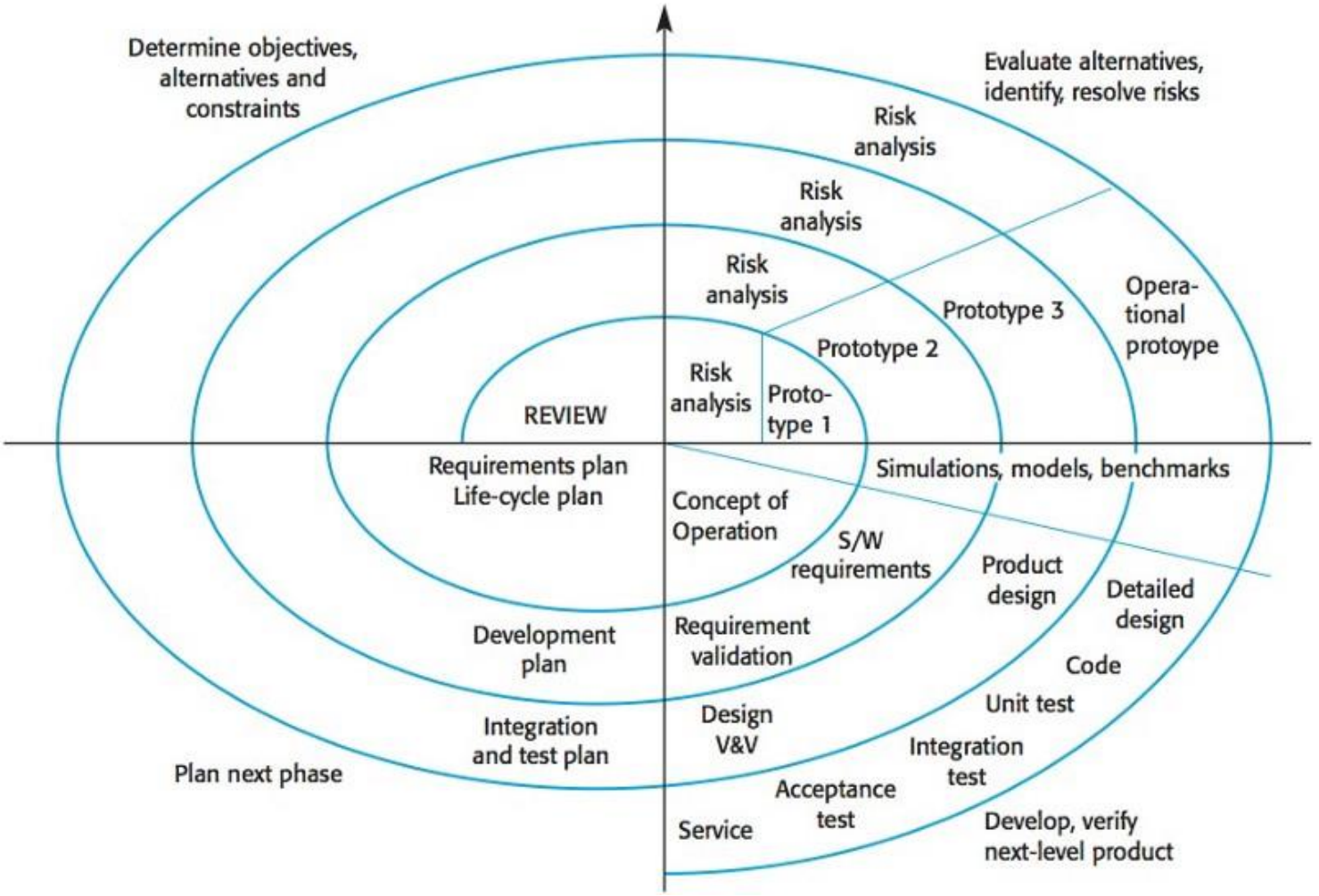
Online payment
* scecurity

- The spiral model is a risk-driven where the process is represented as spiral rather than a sequence of activities.
- It was designed to include the best features from the waterfall and prototyping models, and introduces a new component; risk-assessment.

If risk is involved in sysyem

Ekta Solution ar alternative approach agei measure kore rakha

# Spiral Model



# Spiral Model

- Each loop in the spiral is split into four sectors:

- **Objective setting**: The objectives and risks for that phase of the project are defined.
- **Risk assessment and reduction:** For each of the identified project risks, a detailed analysis is conducted, and steps are taken to reduce the risk. For example, if there's a risk that the requirements are inappropriate, a prototype may be developed.
- **Development and validation:** After risk evaluation, a process model for the system is chosen. So if the risk is expected in the user interface then we must prototype the user interface. If the risk is in the development process itself then use the waterfall model.
- **Planning:** The project is reviewed and a decision is made whether to continue with a further loop or not.

# Spiral Model

- When to use Spiral model:
  - When budgets and risk assessment is important
  - For moderate to high-risk projects
  - Lasting project obligation unwise because of possible changes to economic main concern
  - Clients are uncertain of their requirements

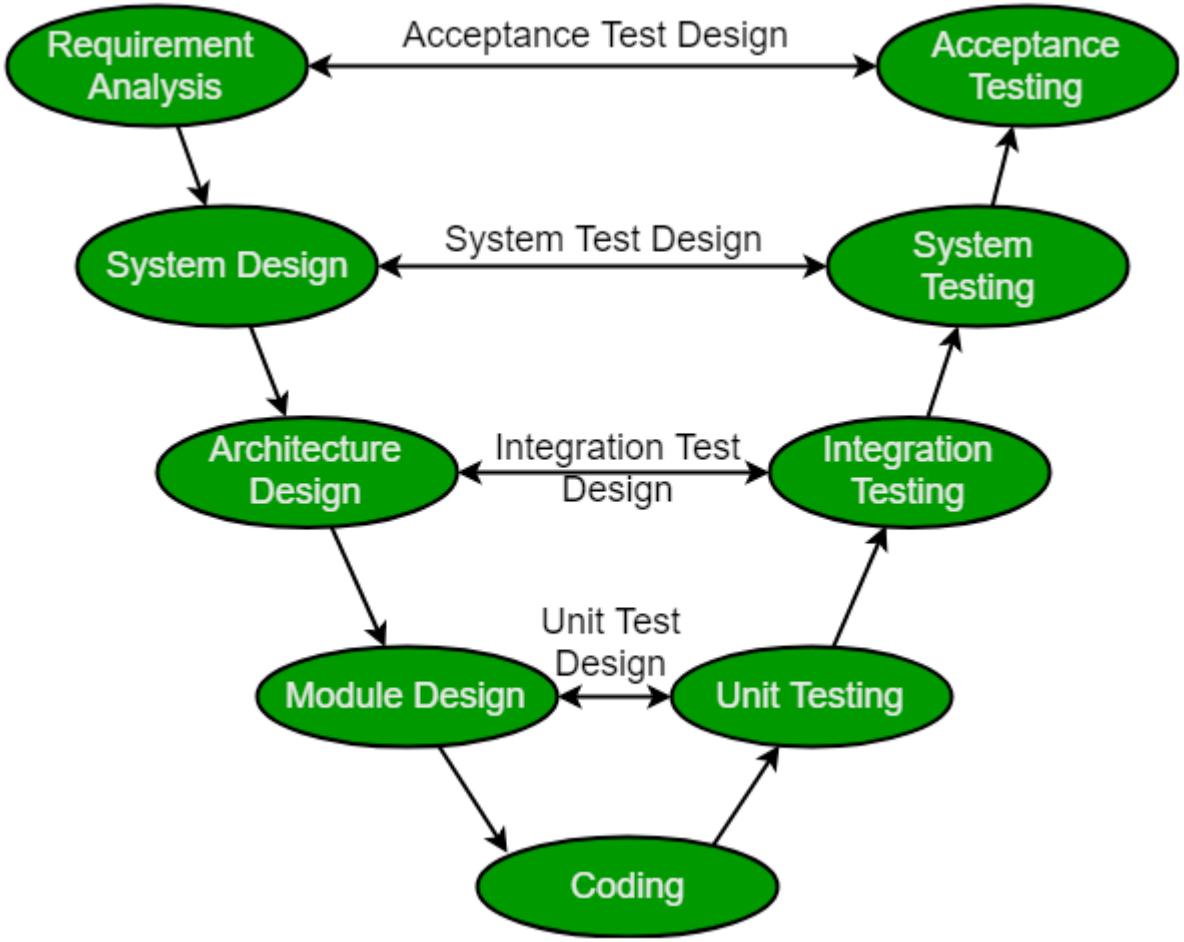
# Spiral Model

| ADVANTAGES | DISADVANTAGES |
| --- | --- |
| Lifecycle is divided into small parts, and if the risk concentration is higher, the phase can be finished earlier to address the treats | Can be quite expensive |
| The development process is precisely documented yet scalable to the changes | The risk control demands involvement of the highly-skilled professionals |
| The scalability allows to make changes and add new functionality even at the relatively late stages | Can be ineffective for the small projects |
| The earlier working prototype is done - sooner users can point out the flaws | Big number of the intermediate stages requires excessive documentation |

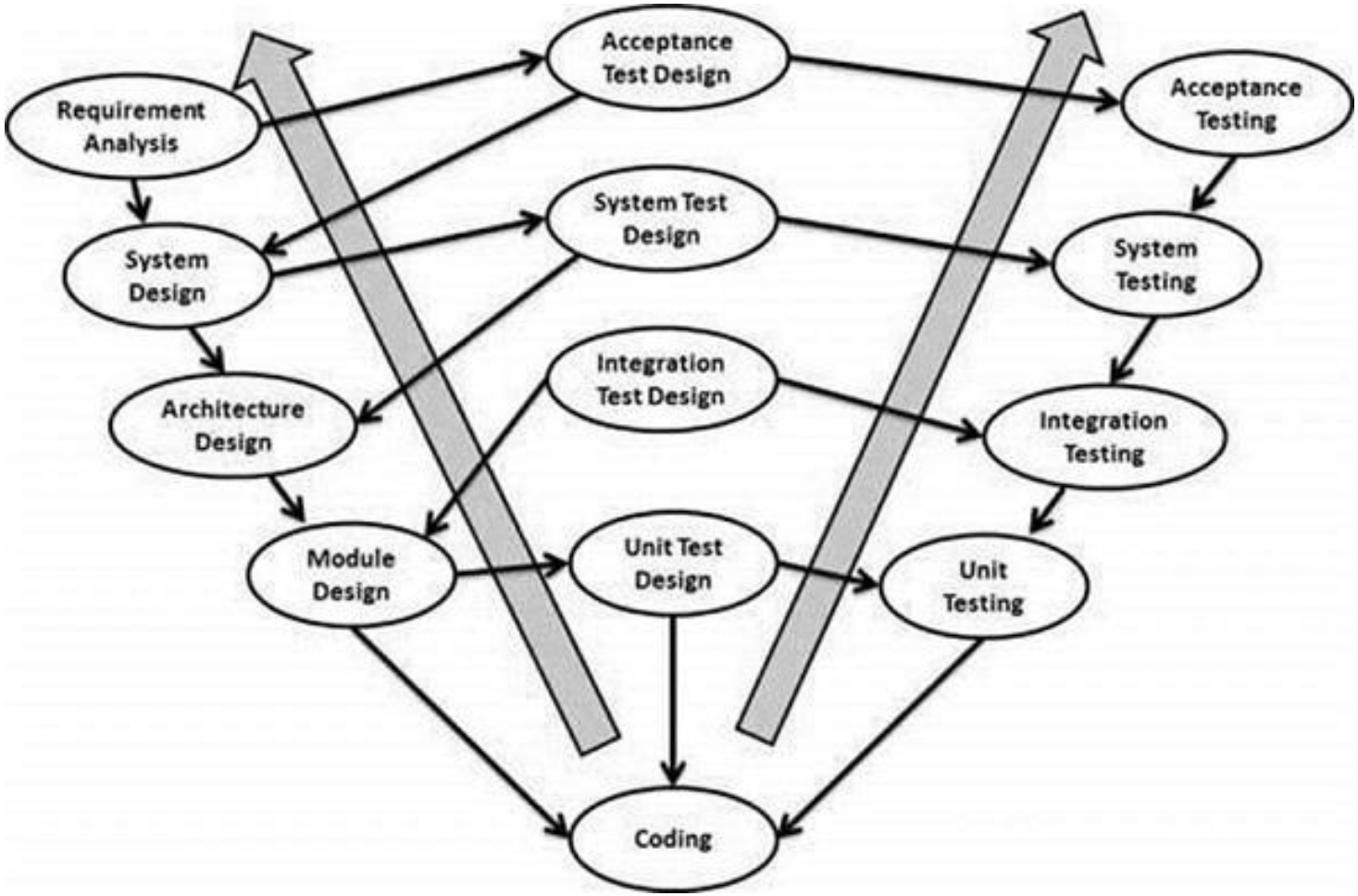
# V-Model

- The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.
- It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage.
- Development of each step directly associated with the testing phase.
- The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.

# V-Model


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding

# V-Model


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

# V-Model

- **Verification:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to find whether specified requirements meet.
- **Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after completion of the development phase to determine whether software meets the customer expectations and requirements.

# V-Model

## ❑Testing Phases:

- **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code level or unit level.
- **Integration testing:** After completion of unit testing Integration testing performs. In integration testing, integrate modules and test the system. Integration testing performs on the Architecture design phase. This test verifies the communication of modules among themselves.

# V-Model

- **System Testing:** System testing test the complete application with their functionality, interdependency, and communication. The System Testing test the functional and non-functional requirements of developed application.

- **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real time.

# V-Model

- **The usage**
  - Software requirements clearly defined and known
  - Software development technologies and tools are well-known

# V-Model

| Advantages | Disadvantages |
| --- | --- |
| ■ Simple and easy to use<br>■ Each phase has specific deliverables.<br>■ Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.<br>■ Works well for where requirements are easily understood.<br>■ Verification and validation of the product in the early stages of product development. | ■ Very inflexible, like the waterfall model.<br>■ Adjusting scope is difficult and expensive.<br>■ The software is developed during the implementation phase, so no early prototypes of the software are produced.<br>■ The model doesn't provide a clear path for problems found during testing phases.<br>■ Costly and required more time, in addition to a detailed plan |