Software Engineering is defined as

- Software engineering is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software
- An establishment and use of sound engineering principles in order to obtain an economical software that is reliable and works efficiently on real machines

Software fails to meet the user requirements

Software crashes frequently

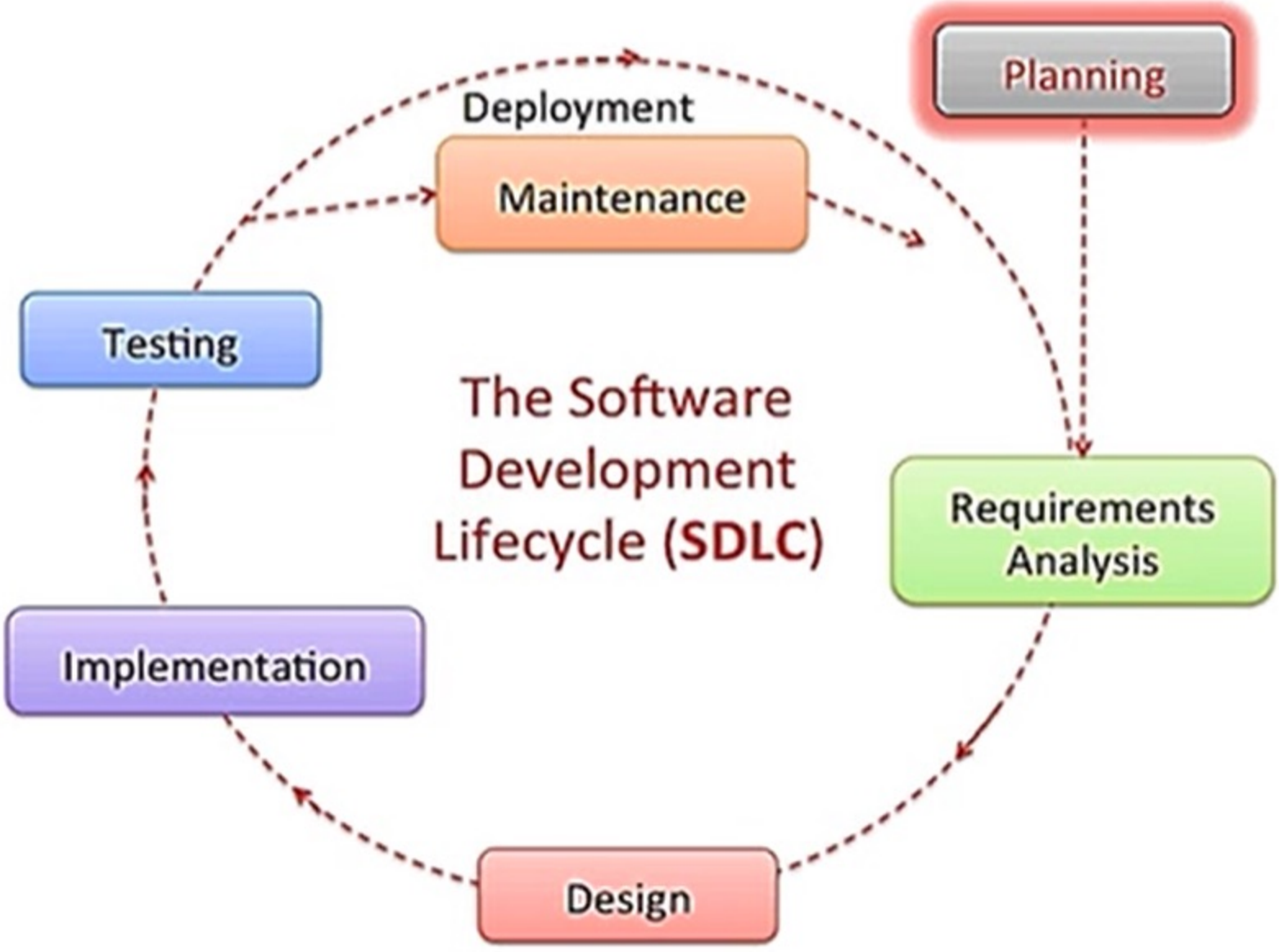
Development of Software became expensive

Difficult to alter, debug, and enhance the software

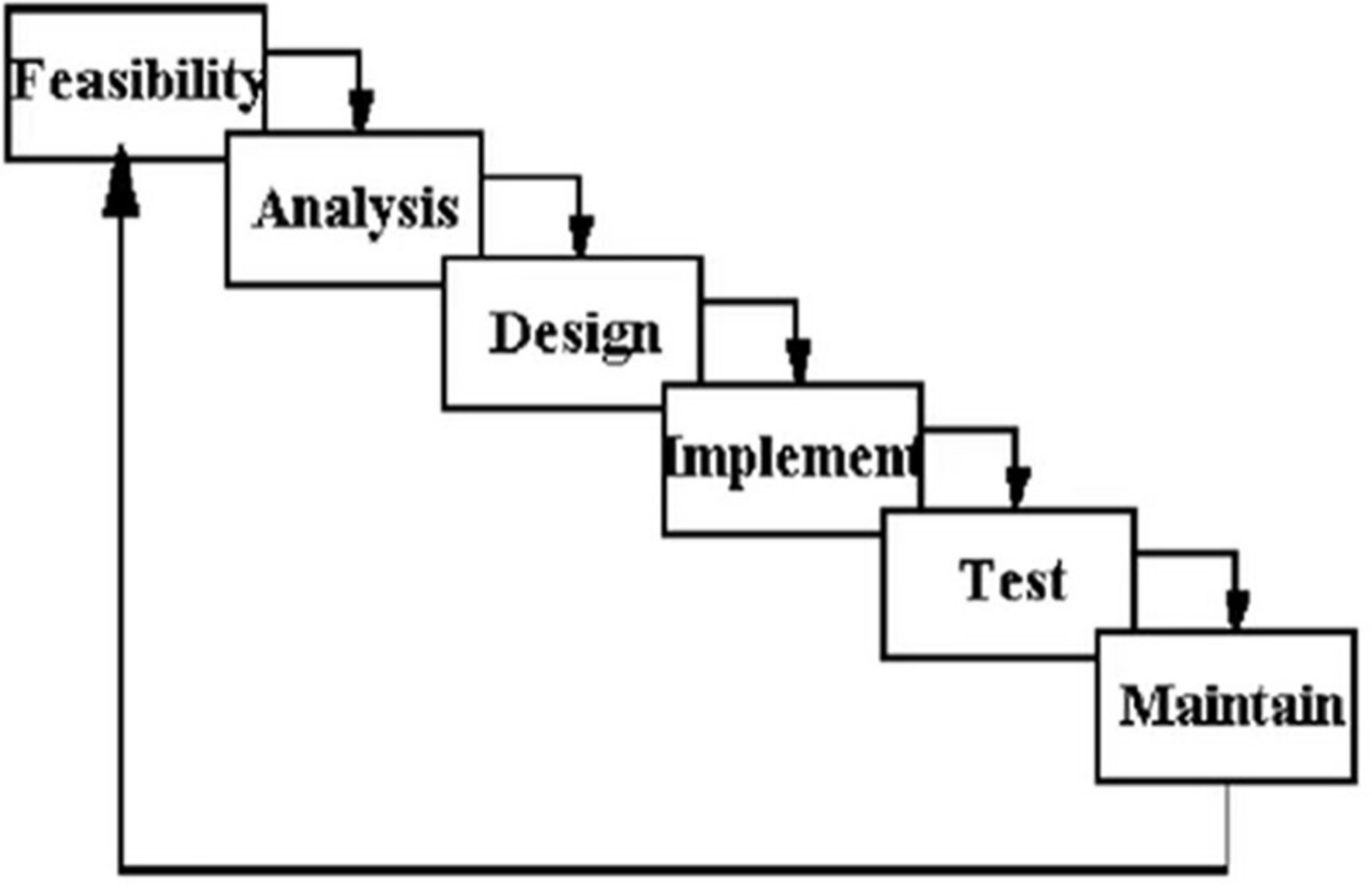
Software was often delivered late

Software used resources non-optimally

Software Development Life Cycle



Phases in SDLC



Analysis

The goal of the system analysis is to define the requirements of the system

Requirement gathering requires client as well as the service provider to get the detailed and accurate requirements

SRS(Software Requirement Specification) is the primary artifact of Analysis phase

Activities in Analysis Phase

Requirements gathering and analysis

Preparing Requirements Specification(SRS)

Design

Software design deals with transforming the customer requirements into a set of documents that is suitable for implementation in a programming language

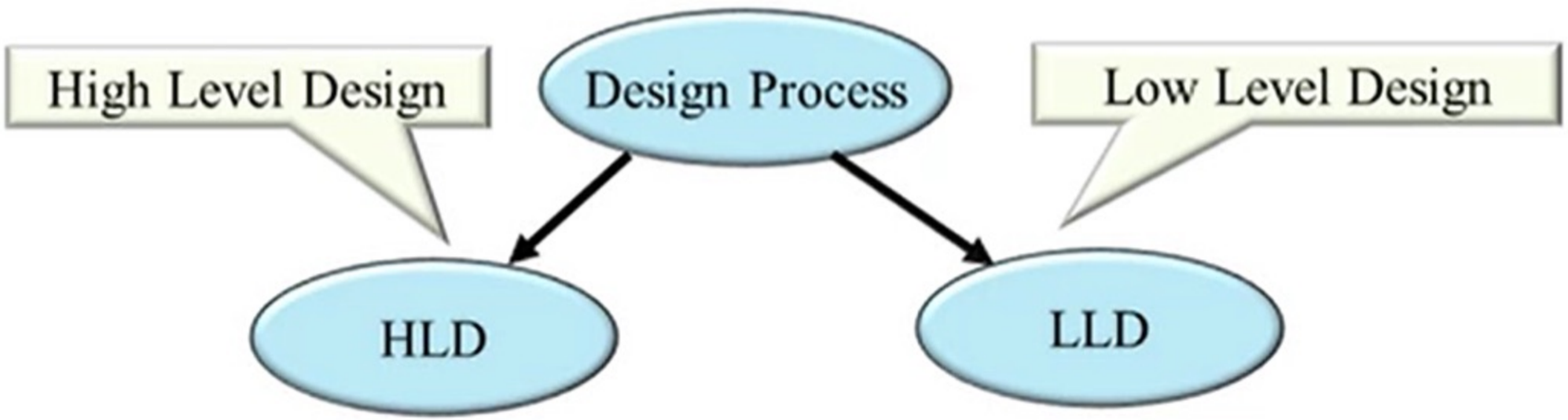
It is the process of defining the architecture, interface, component and other characteristics of a system

The design stage takes the requirements identified in the approved requirements document (SRS) as its initial input

Levels of a Design

Decompose the entire project into units / modules and identify the system architecture, data structure and processing logic

- DD(Design Document)= HLD + LLD



Construction(Code + Unit Testing)

Modular and subsystem programming code will be accomplished during this stage

Unit testing /module testing is done in this stage by the developers

This stage produces the source code, executable, and databases applicable

Testing

TEKNOTURF

Testing is the process of executing the program with the intent of finding errors

Software testing is a process of verifying and validating that a software application or program meets the business and technical requirements

Levels of Testing

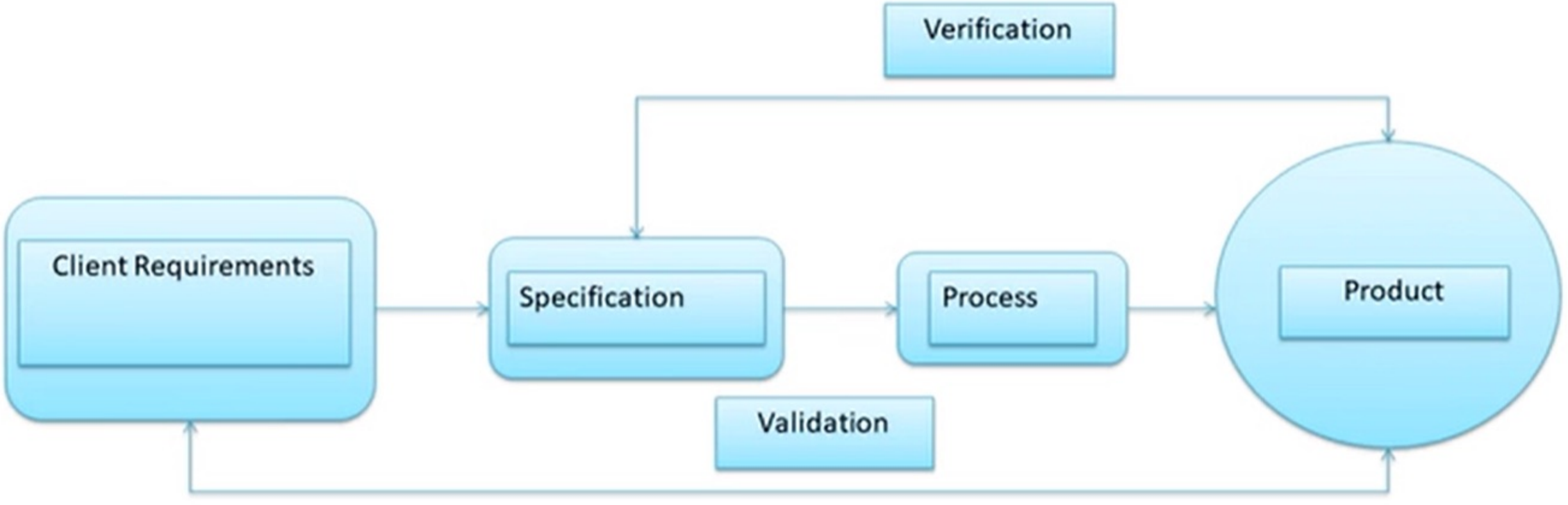
- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Software testing includes

Verification	• Confirms that the software meets its technical specifications
Validation	• Confirms that the software meets the business requirements
Defect	• Variance between the expected and actual result

Verification, Validation and Defect Finding

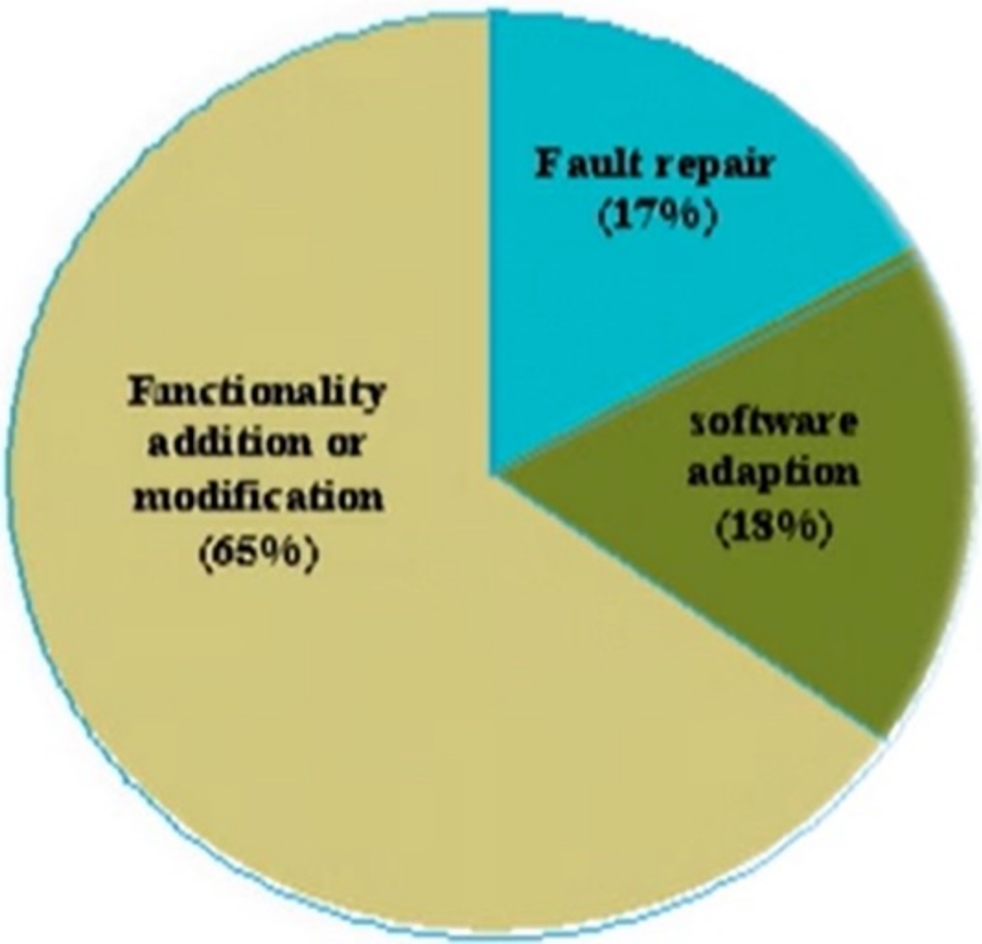
TEKNOTURF



Maintenance

TEKNOTURF

Changes or enhancements happen everywhere. Software is no exemption. Any change that is made to the software after it is deployed is known as maintenance.



Software Development Life Cycle Models

TEKNOTURF

- Waterfall model
- V-model
- Prototype model
- RAD (Rapid Application Development)
- Incremental model
- Spiral model

Waterfall Model

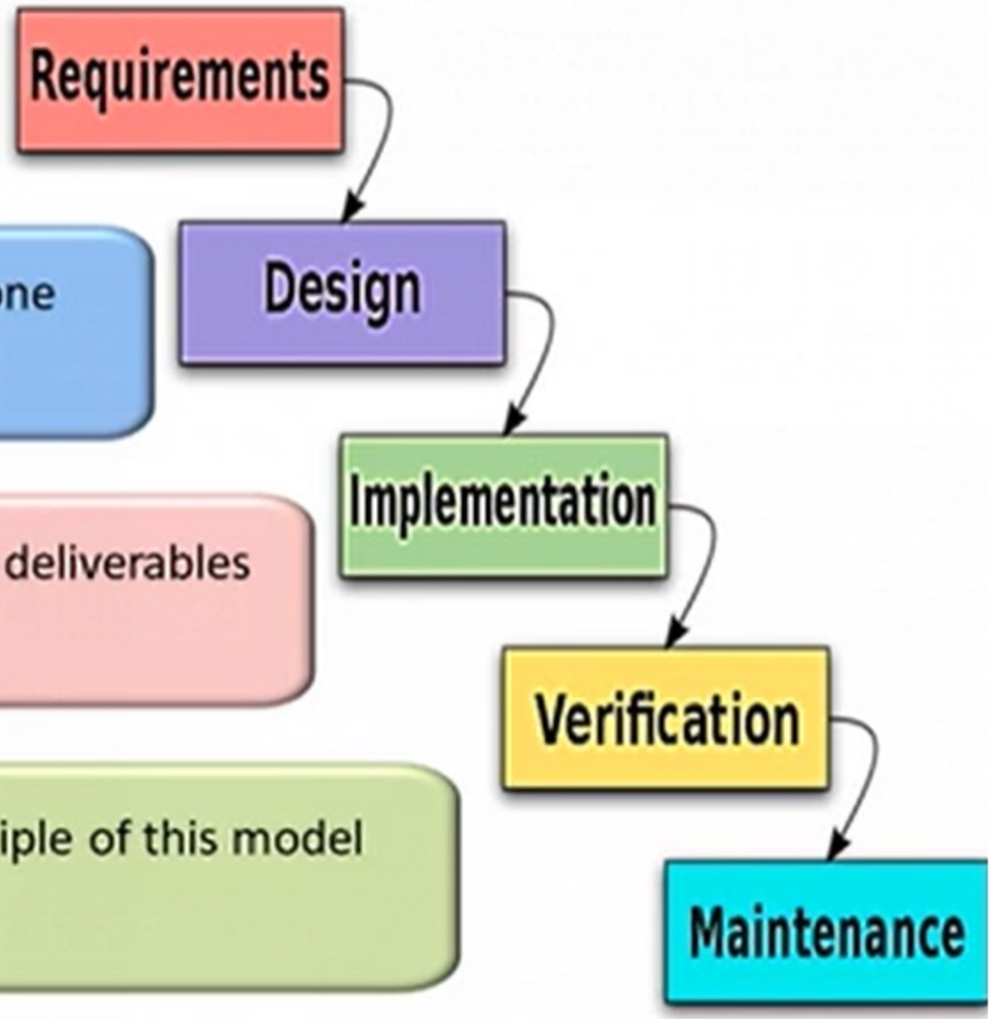
TEKNOTURF

This model was proposed by Winston Royce in 1970

Waterfall model derives its name due to the cascading effect from one phase to the other as depicted in the diagram

Each phase has a well defined start and end point, with identifiable deliverables to the next phase

It is a linear sequential model, systematic in approach and the principle of this model suggests - "Can't retrieve to previous phase"



Advantages of Waterfall Model

TEKNOTURF

- Simple and easy to use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process
- Phases are processed and completed one at a time
- Works well for smaller projects where requirements are very well understood
- Linear approach
- Equivalent importance to all the phases
- Contract Related issues can be addressed effectively

Limitations of Waterfall Model

- This model is suitable if the requirements are well-defined and stable
- User gets a feel of the system only at the later stages of development
- Backtracking cost is high in case of a problem
- Increased development of time and cost
- Systems must be defined up front
- Rigidity
- Hard to estimate costs & project overruns

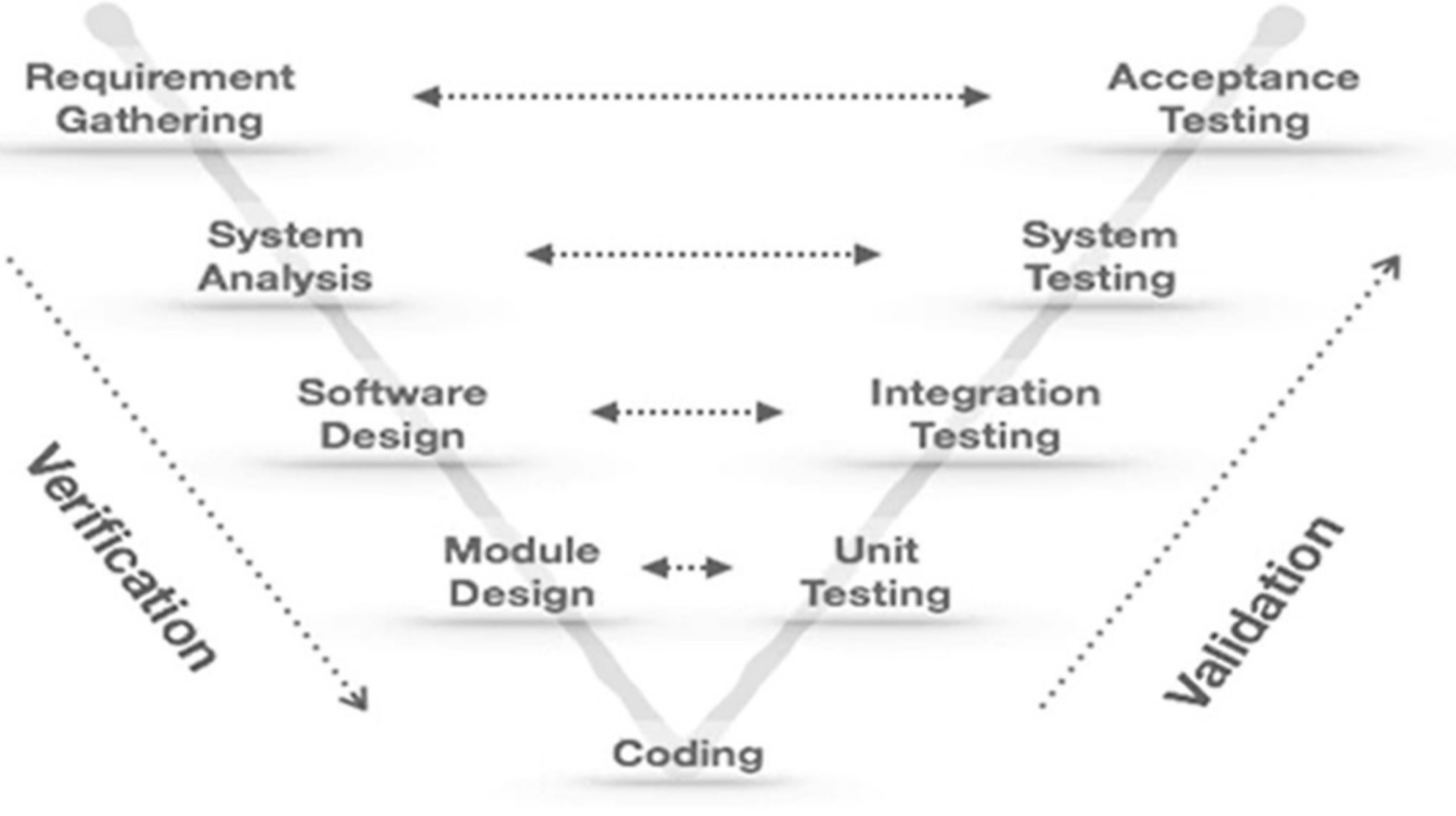
Waterfall Model

TEKNOTURF

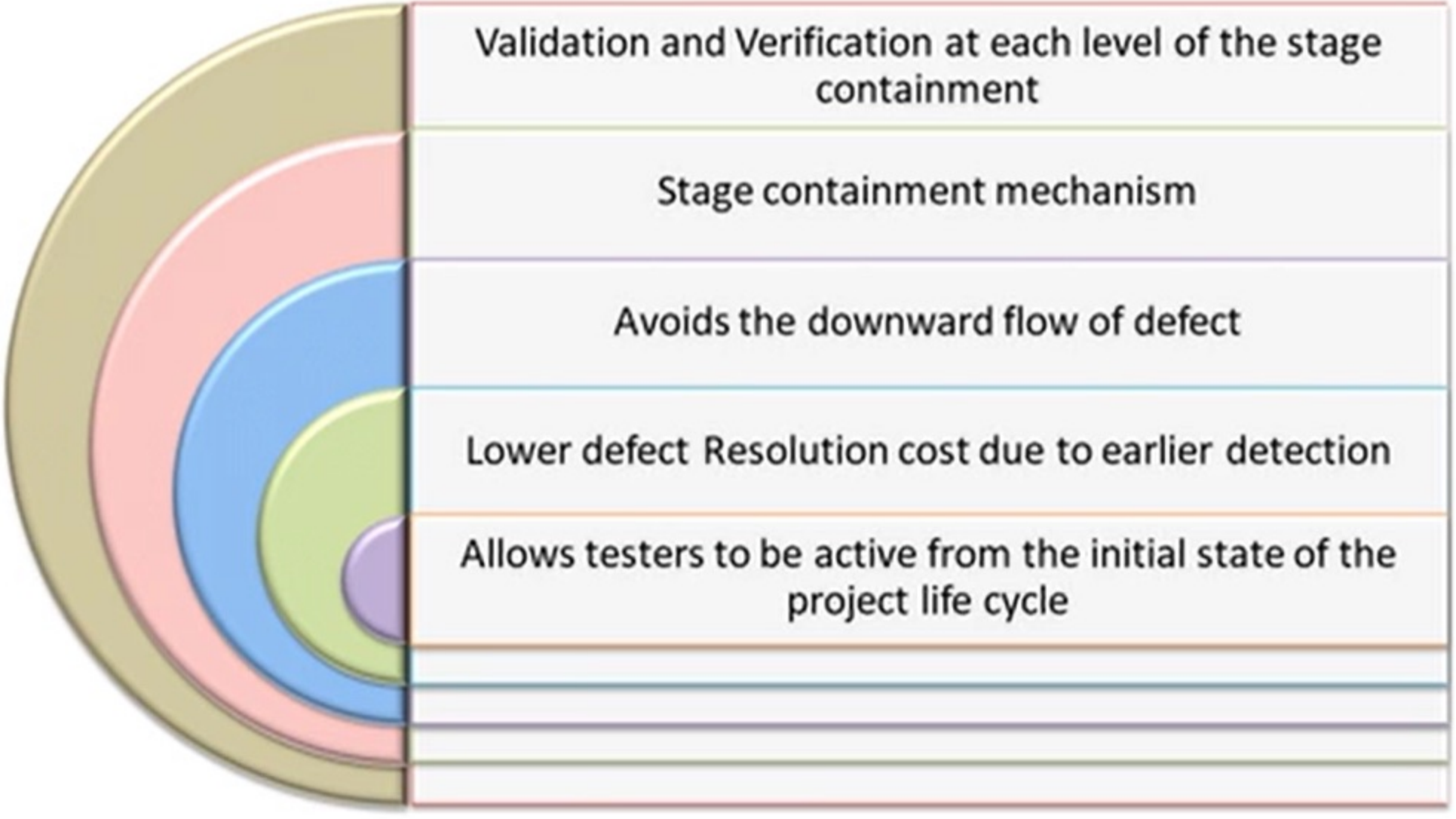
Suitable when

- Software requirements are clearly defined and known
- Product definition is stable
- Software development technologies and tools are well known
- New version of the existing software system is created

V-Model


Requirement Gathering
Acceptance Testing
System Analysis
System Testing
Software Design
Integration Testing
Module Design
Unit Testing
Coding
Verification
Validation

Advantages of V-Model



Limitations of V-Model

Prototype Model

TEKNOTURF

Creates prototypes, which is an incomplete version of the software program being developed.

Simulates only few aspects of the features of the System to be built

Prototype Model

TEKNOTURF

Prototyping can also be used by the end users to describe and prove requirements that the developers have not considered.

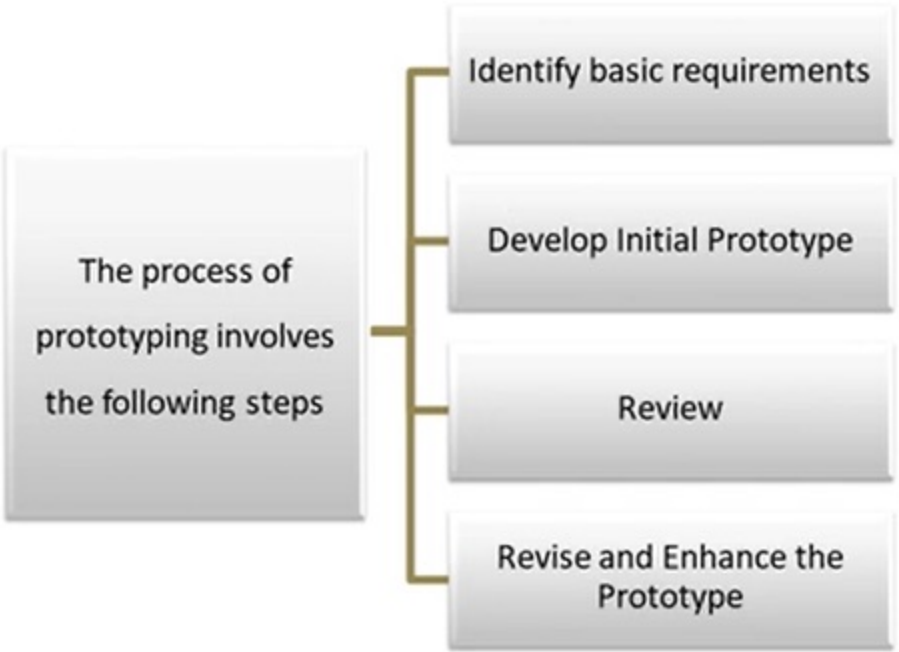
Developers build a prototype during the requirements phase.

Prototype is evaluated by the end users to provide corrective feedback

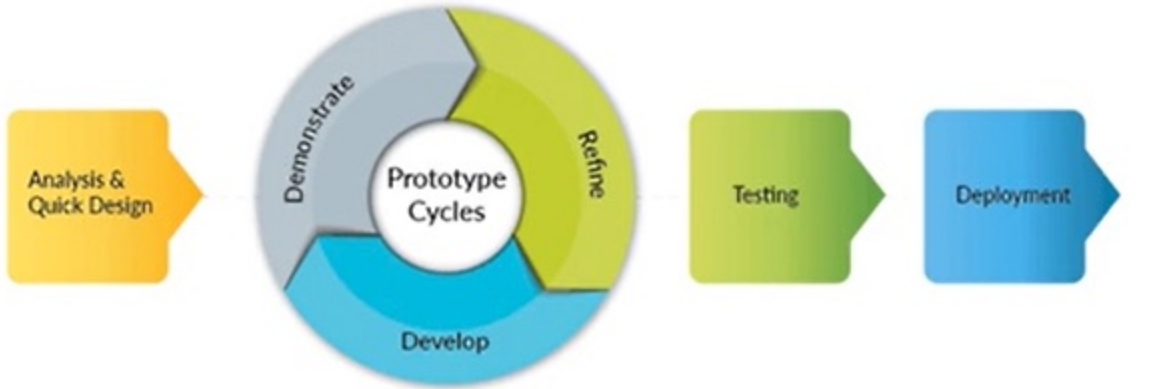
Developers further refine the prototype based on feedback.

When the user is satisfied, the prototype code is brought up to the standards needed for the final product.

Prototype Model



Prototype Model

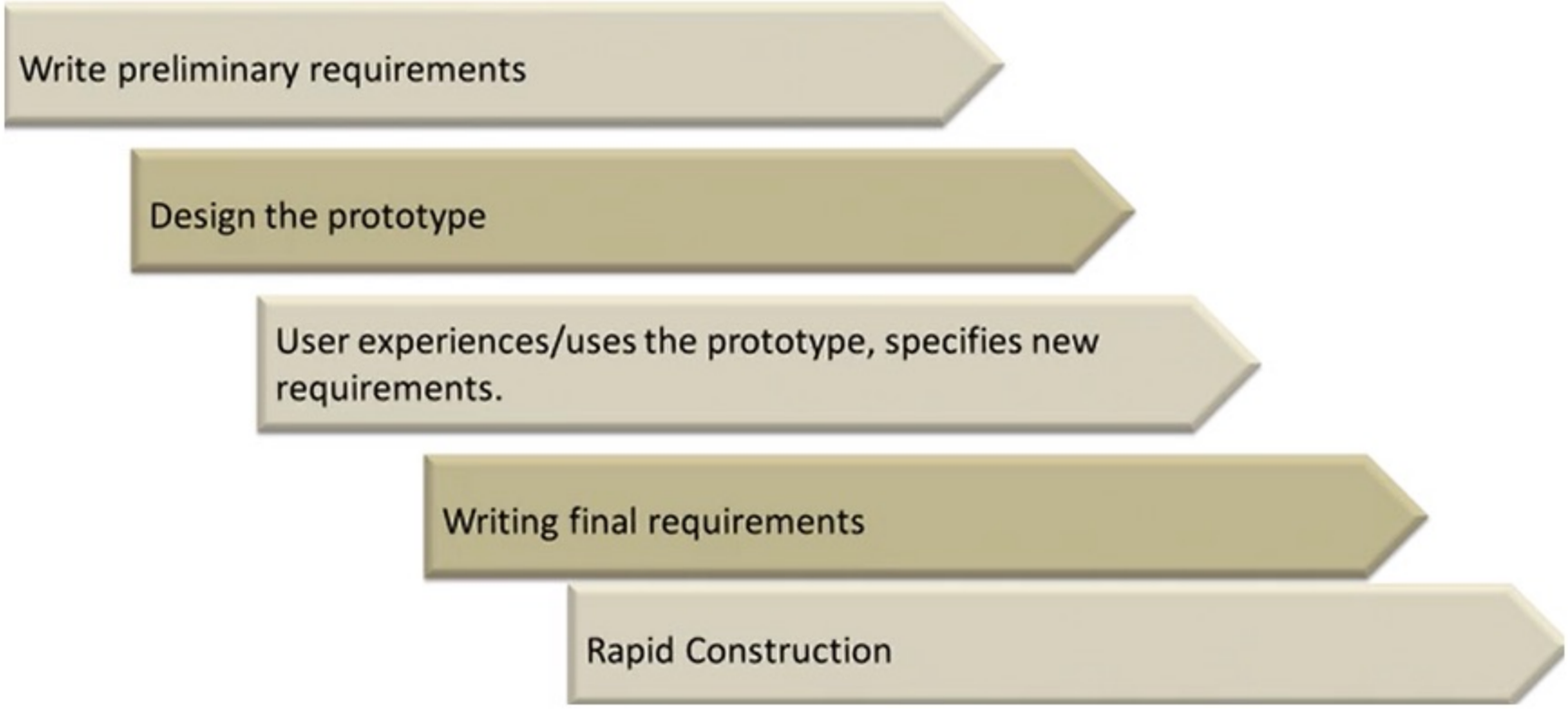


What are the types of prototypes?

- Throw away
- Evolutionary

Throw away prototyping-Steps

TEKNOTURF

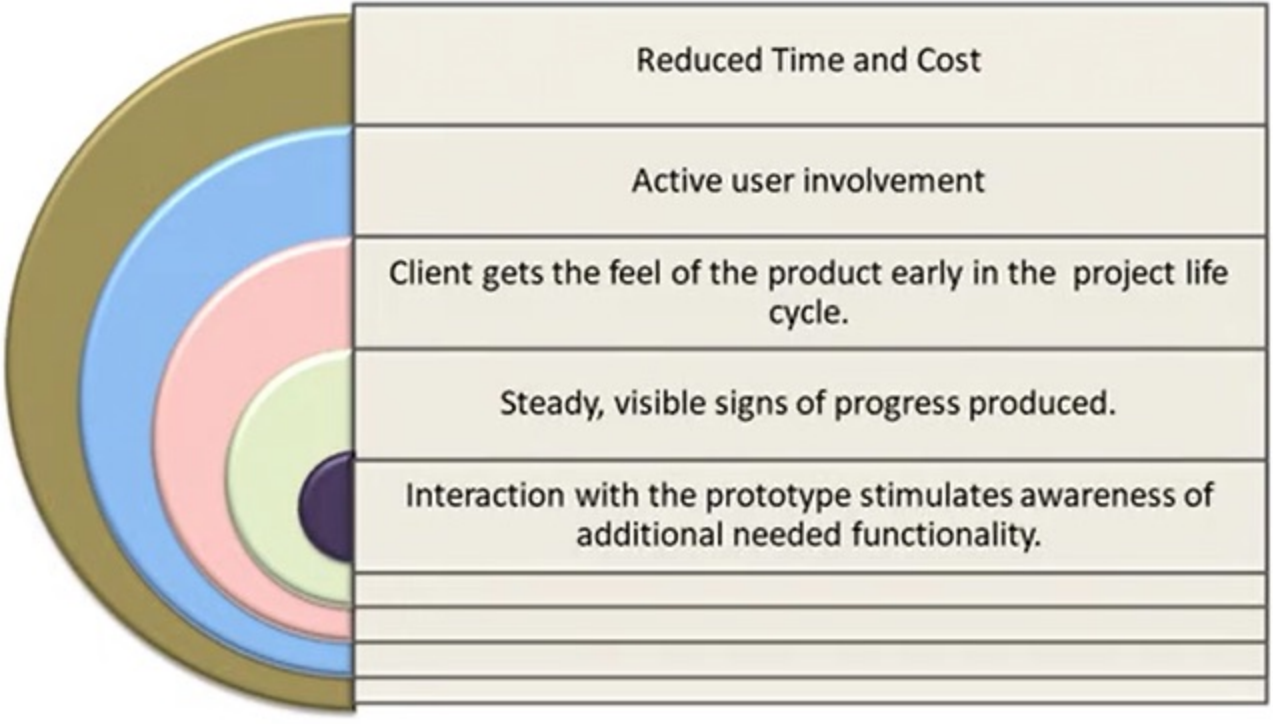


Evolutionary Prototype Model

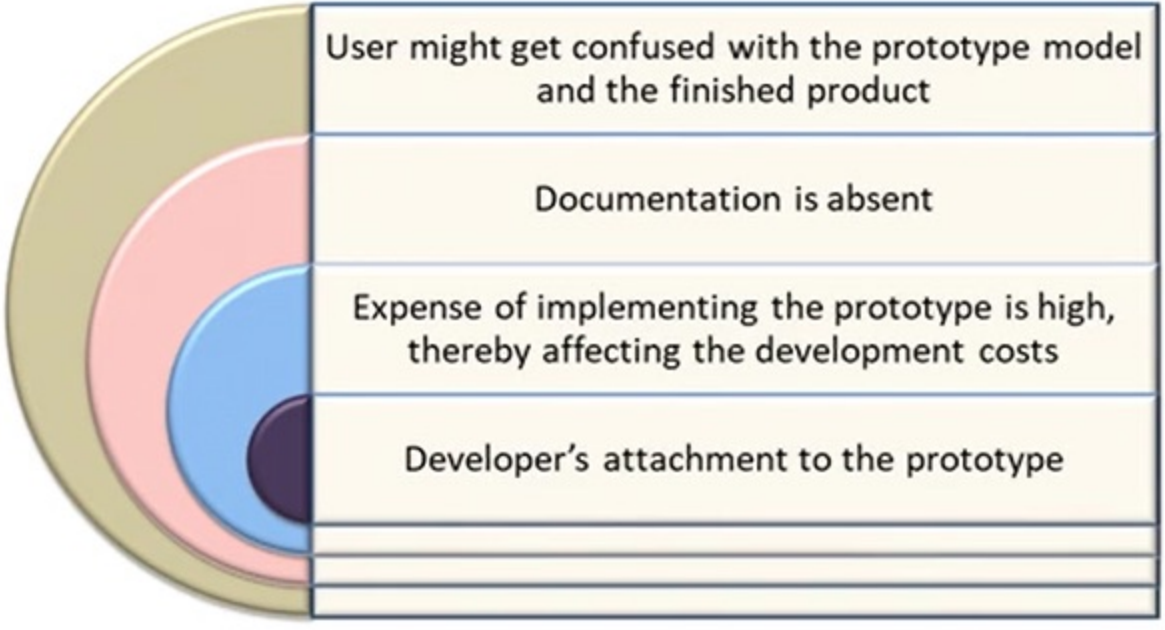
TEKNOTURF

- Requirements are prioritized and the code is developed initially for stable requirements, with an eye on quality
- Software is continuously refined and augmented in close collaboration with the client
- Build the software incrementally
- Adopt a rigorous, systematic approach
- Iterative model

Advantages of Prototype Model



Limitations of Prototype Model

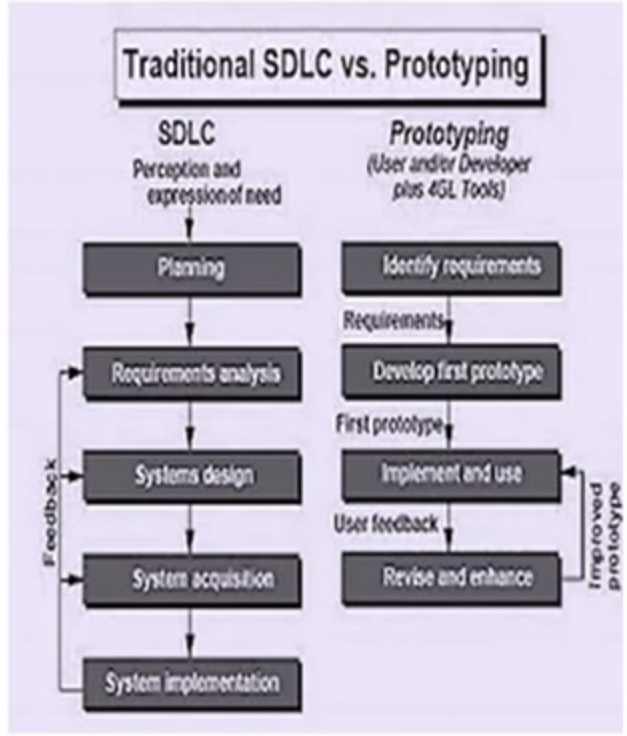


When to use Prototype Model?

TEKNOTURF

When requirements are unclear, use "Throwaway Prototype Model"

When requirements are unstable, use "Evolutionary Prototype Model"



Rapid Application Development

RAD is a high speed version of the linear sequential model

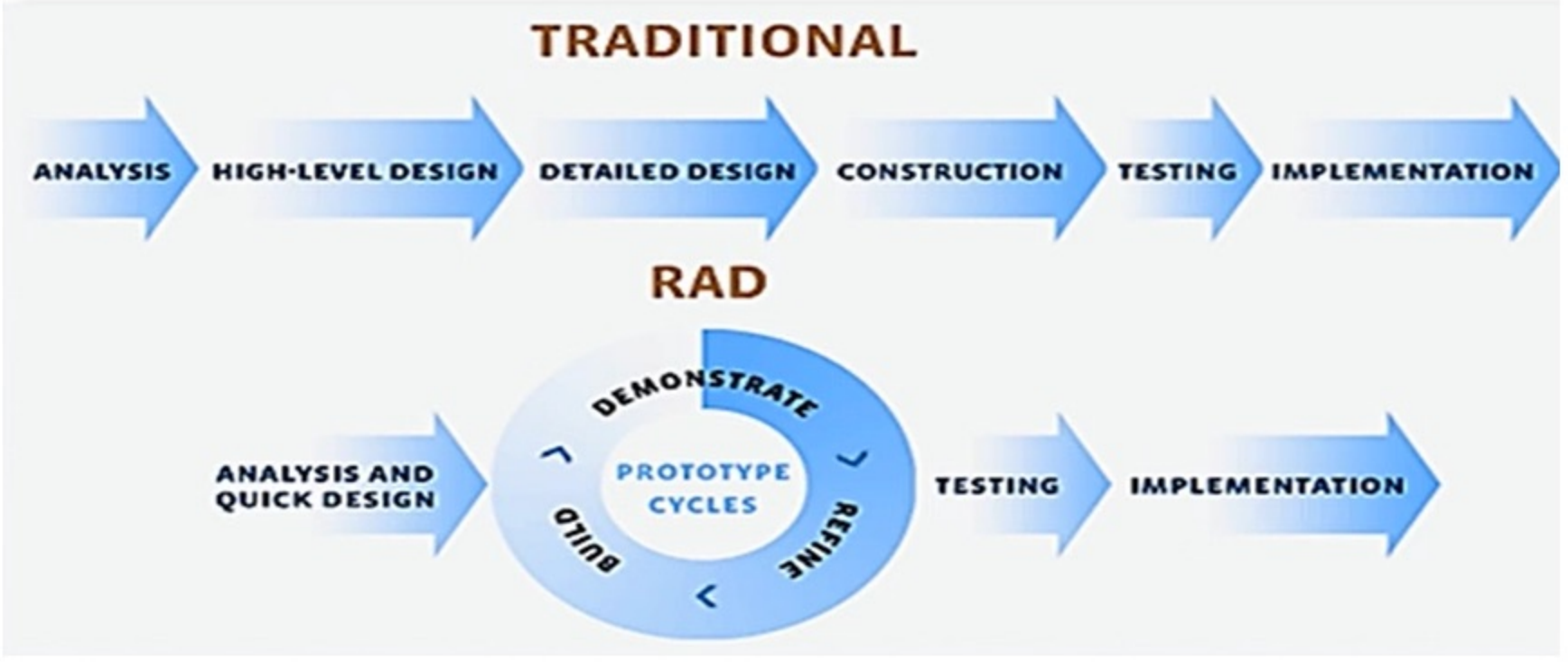
Characterized by a very short development life cycle

The RAD model follows a component based approach

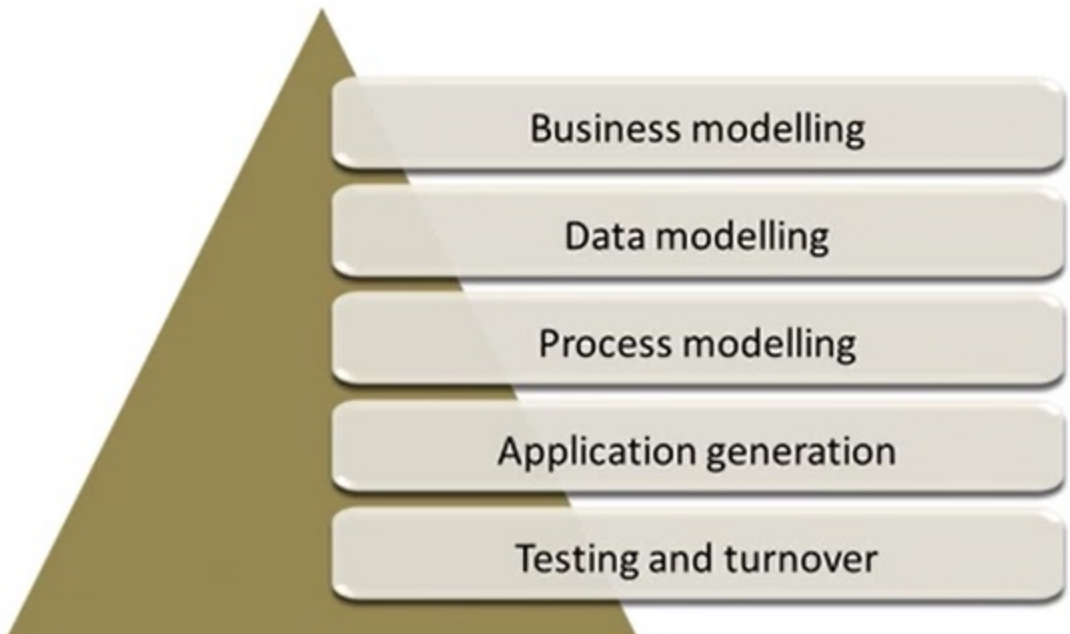
Individual components are developed by different people and assembled to develop a large software system

Traditional vs RAD

TEKNOTURF



Rapid Application Development phases



Advantages and Limitations of RAD

TEKNOTURF

Advantages:

- Due to the emphasis on rapid development, it results in the delivery of a fully functional project in short period.
- Facilitates Parallel Development.

Limitations:

- Developers and clients must be committed to rapid-fire activities in an abbreviated time frame.
- If either party is indifferent in needs of other, the project will run into serious problem.
- It is not suitable for large projects.

When to use RAD?

TEKNOTURF

When requirements are not fully understood

User is involved throughout the life cycle

System can be modularized

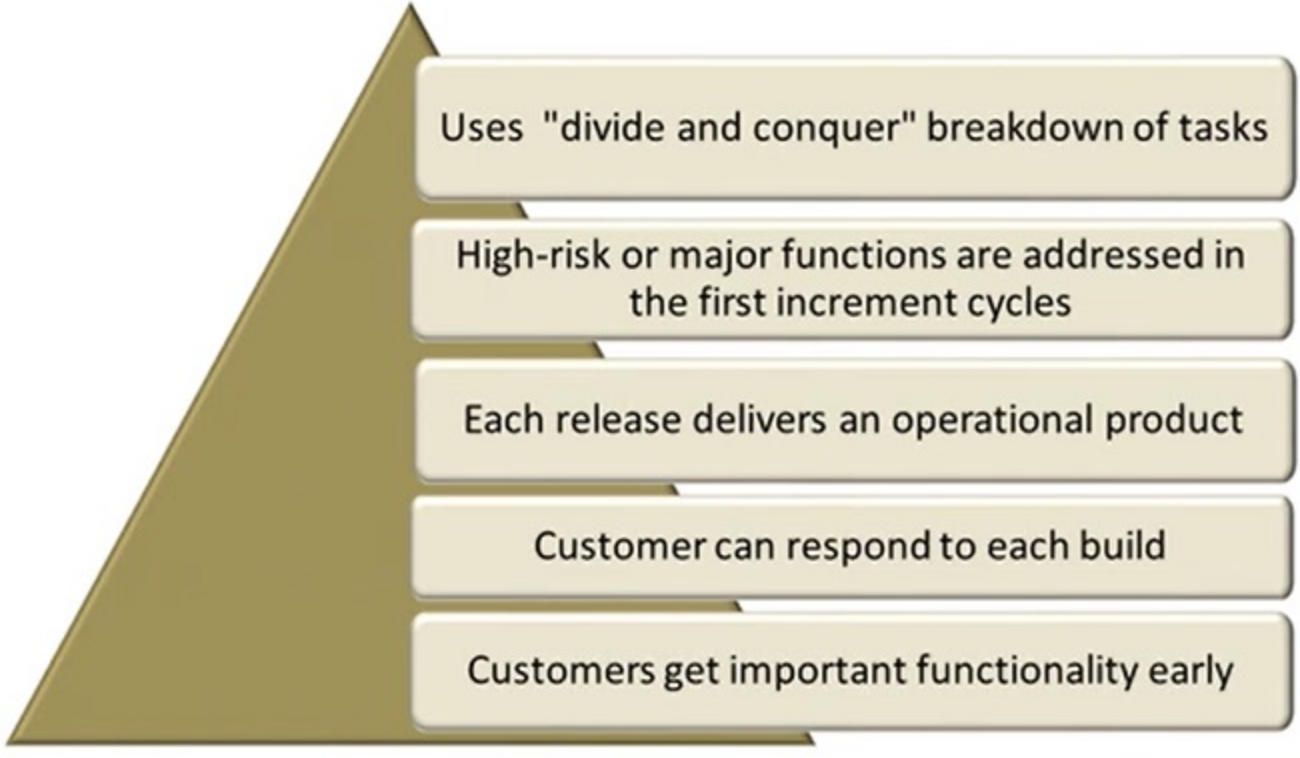
Incremental Model

TEKNOTURF

The incremental model prioritizes requirements of the system and implements them in groups

Each subsequent release of the system adds function to the previous release, until all designed functionalities have been implemented

Advantages of Incremental Model



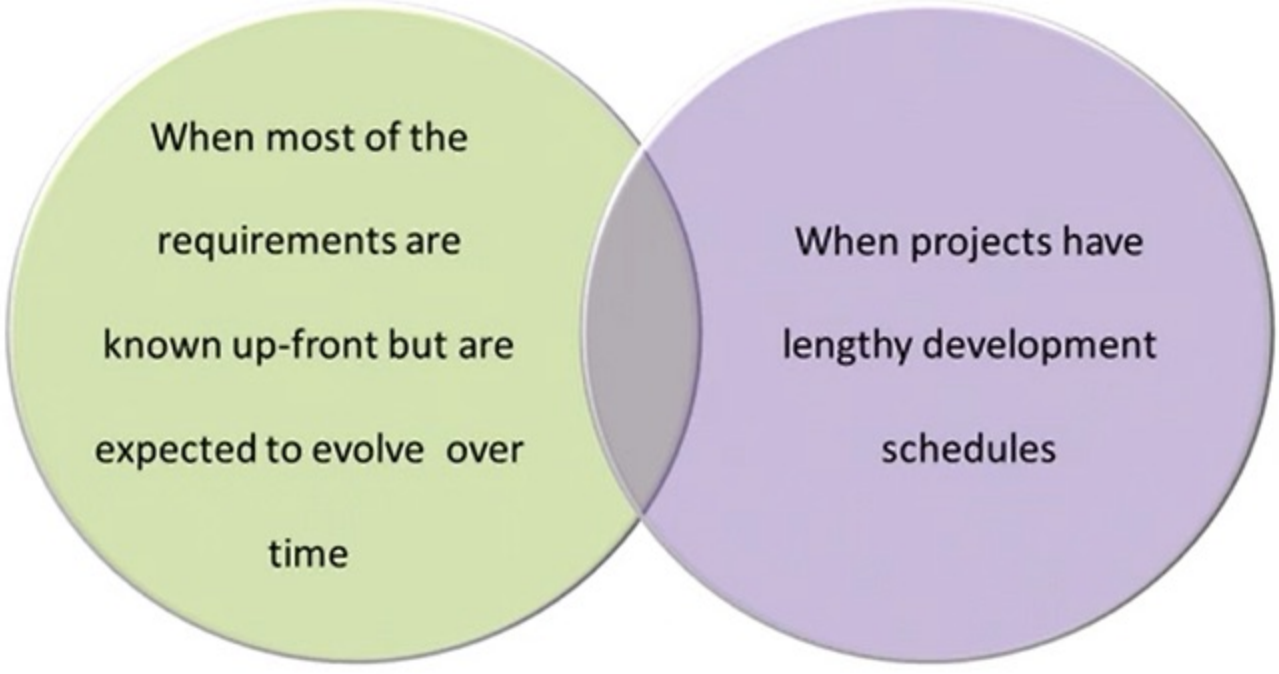
Limitations of Incremental Model

TEKNOTURF

- Requires early definition of a complete and fully functional system to allow for the definition of increments
- Requires good planning and design as basis for the system
- Absence of a well-defined module interface is a major obstacle for this model of development

When to use Incremental Model?

TEKNOTURF



Spiral Model

TEKNOTURF

- Proposed by Barry Boehm in 1986
- Diagrammatic representation of this model appears like a spiral with many loops
- Suitable for technically challenging software products that are prone to several kinds of risks
- Accommodates prototyping. This model combines the features of the prototyping model and the waterfall model
- It is favoured for large, expensive, and complicated models
- Suggested for High-Risk Scenarios based projects

Spiral Model

Advantages of Spiral Model

- Provides early indication risk.
- Users see the system early because of rapid prototyping tools.
- Critical high-risk functions are developed first.
- Early and frequent feedback from users.

Limitations of Spiral Model

TEKNOTURF

When to use Spiral Model?

Risk perceived is very high

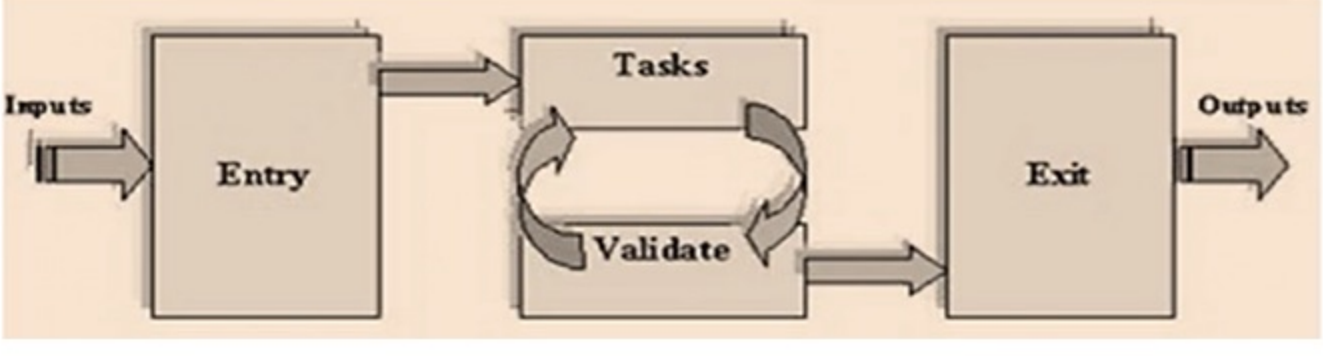
Requirements are complex

Significant changes are expected

Process Representation Technique – ETVX Model

IBM introduced the ETVX model to document their process.

Each phase in a process performs a well-defined task and generally produces an output.



ETVX Model

ETVX Model Example

TEKNOTURF

Entry

- Hall Ticket

Task

- Show Hall Ticket
- Get Question Paper
- Get Answer Sheet
- Write Answer for Respective questions

Verification

- Verify whether questions are written for appropriate questions
- Review the answers written

Exit

- Submission of the answer sheet to the invigilator

ETVX Model Example for Analysis Phase

Entry

- Feasibility Report

Task

- Collect requirement
- Analyze requirement
- Write SRS

Verification

- Review SRS

Exit

- SRS

Program

- A computer program is a sequence of instructions written to perform a specified task in a computer

Software

- A software is a set of programs, procedures and its documentation concerned with the operation of a data processing system

Software Process

- A Process is a series of definable, repeatable, and measurable tasks leading to a useful result