SOFTWARE DEVELOPMENT PRACTICES

Arindam Chakravorty

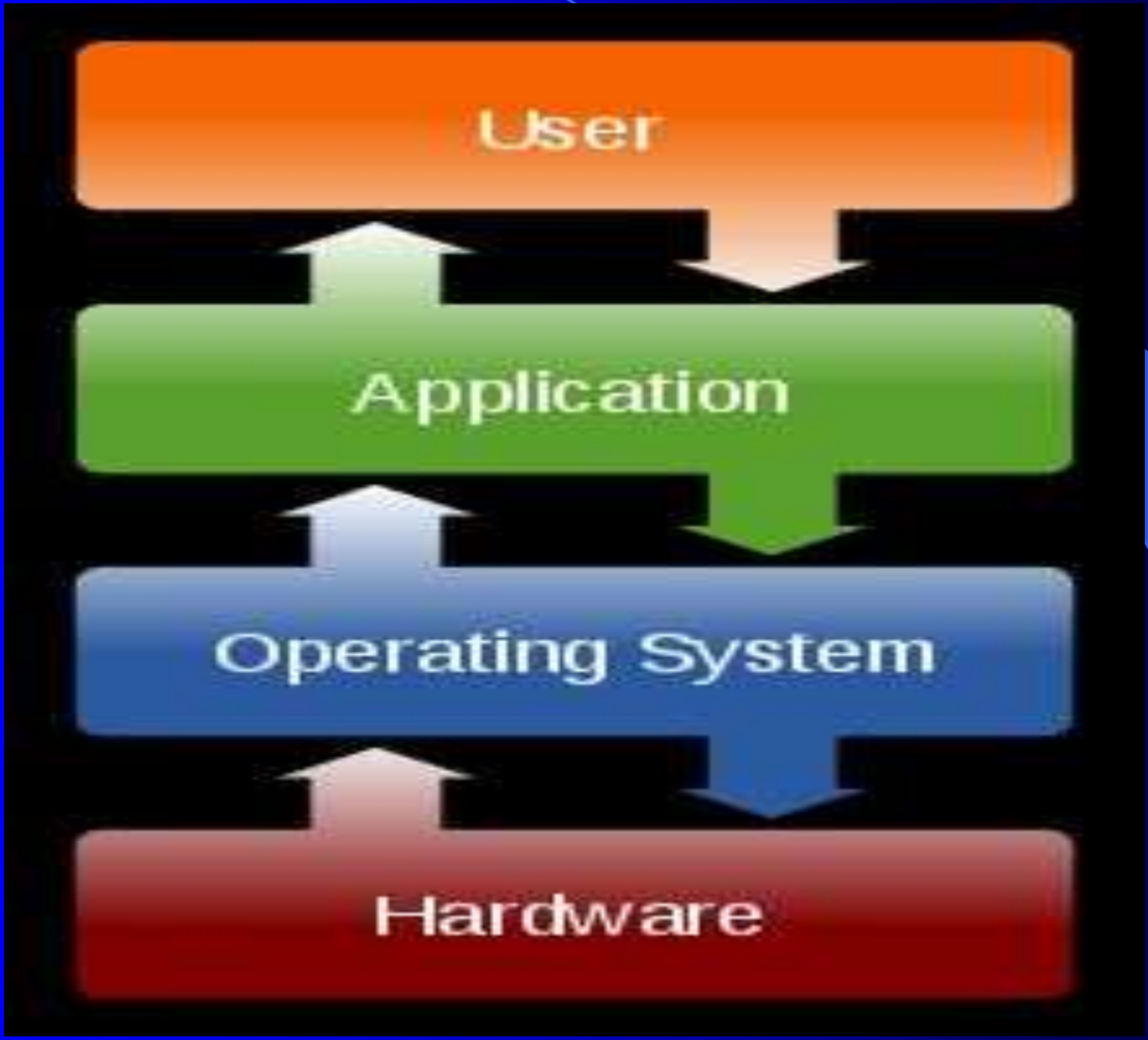
Debashis Chakraborty

STCET

What is a SOFTWARE?

- Collection of computer programs and related data - provides instructions for computer what to do and how to do it.
- Set of *programs*, *procedures*, *algorithms* and its *documentation* concerned with data processing.
- Different from computer hardware - encompasses physical interconnections and devices required to store and execute the software.

Layered Architecture



Types of software

- **System software**

  Designed to operate the computer hardware to provide basic functionality and platform for running application software

- **Programming software**

  Tools in the form of programs or applications - software developers use to create & maintain other programs and applications

- **Application software**

  Perform tasks that benefit from computation - allow the computer to perform a specific data processing job for the user.

Software as a product

- Utility for the end user - Delivers computing potential
- Produces, manages, acquires, modifies, displays, or transmits information
- Requires quality control
- Requires maintenance

- Developed instead of manufactured
- Do not wear out in the sense of other products
- Becomes obsolete / incompatible
- Mostly custom built

Software as vehicle for delivering a product

- Supports or directly provides system functionality
- Controls other programs (e.g., an operating system)
- Effects communications (e.g., networking software)
- Helps build other software (e.g., software tools)

The Software Crisis

- **Projects running over-budget**
- **Projects running over-time**
- **Inefficient Software**
- **Low quality Software**
- **Software does not meet requirements**
- **Project is unmanageable / Code difficult to maintain**

Myths and Reality

Myth: Software can be changed easily

Reality: Requirement changes are a major cause of software degradation

Myth: Schedule problems can be solved by adding more programmers.

Reality: Maybe. It increases coordination efforts and ***may slow things down***.

Myths and Reality

Myth: Coding may start even before all requirements are known

Reality: Incomplete up-front definition is ***the major cause*** of software project failures.

Myth: Writing code is the major part of creating a software

Reality: Coding may be as little as 10% of the effort, while design may involve upto 40% and 50 - 70% effort may be required after delivery

Myths and Reality

Myth: The only deliverable that matters is working code

Reality: Documentation, test history, and program configuration are critical parts of the delivery.

Myth: I am a (super) programmer. Let me program it, and I will get it done

Reality: A formula for failure. Software are developed by teams, not individuals, and success requires ***much more*** than just coding.

Software Engineering

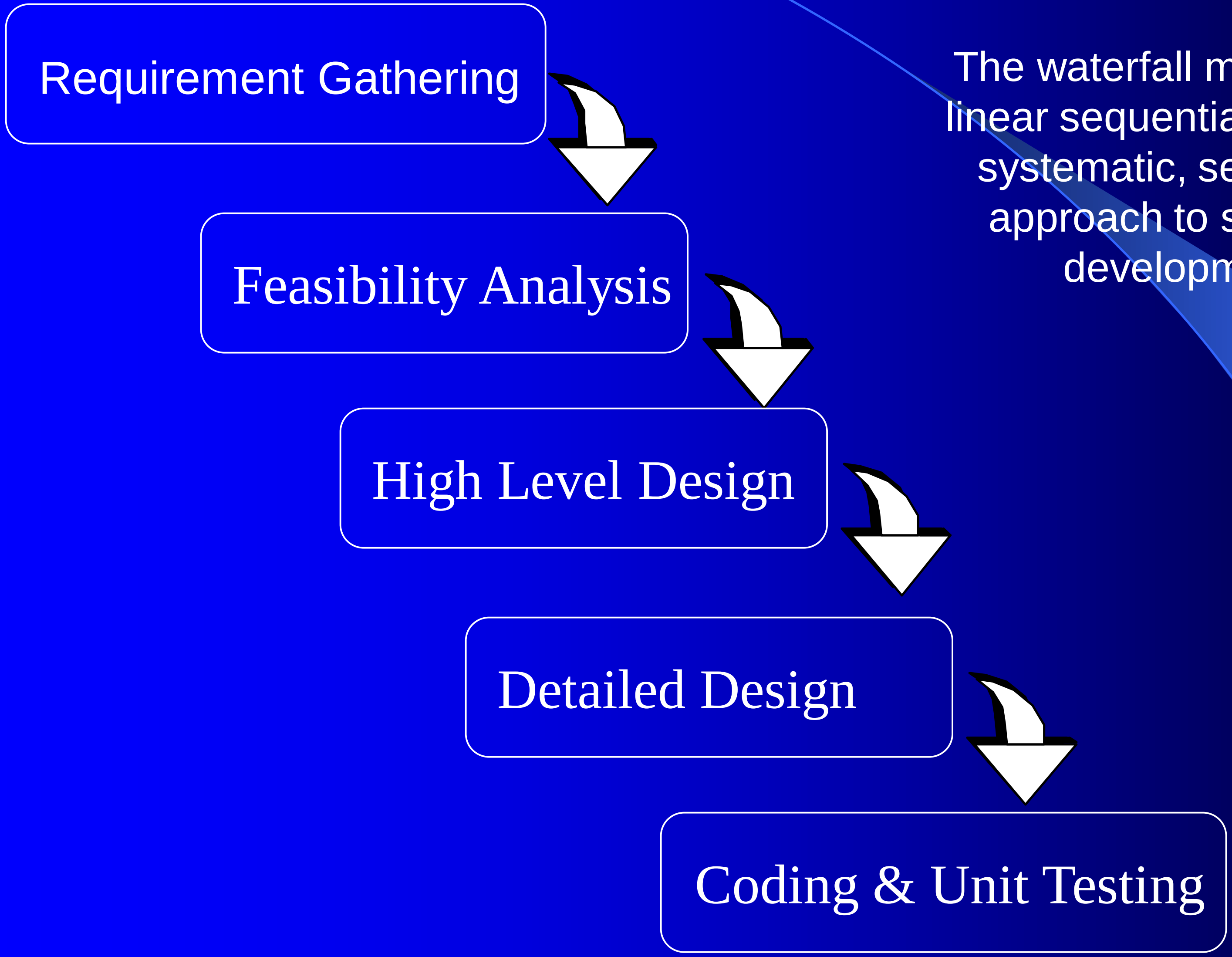
- Application of engineering to software
- Systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

Types of Software Development Models Software Development Life Cycle (SDLC)

- Waterfall Model
- Prototyping Model
- Incremental Model
- Spiral Model

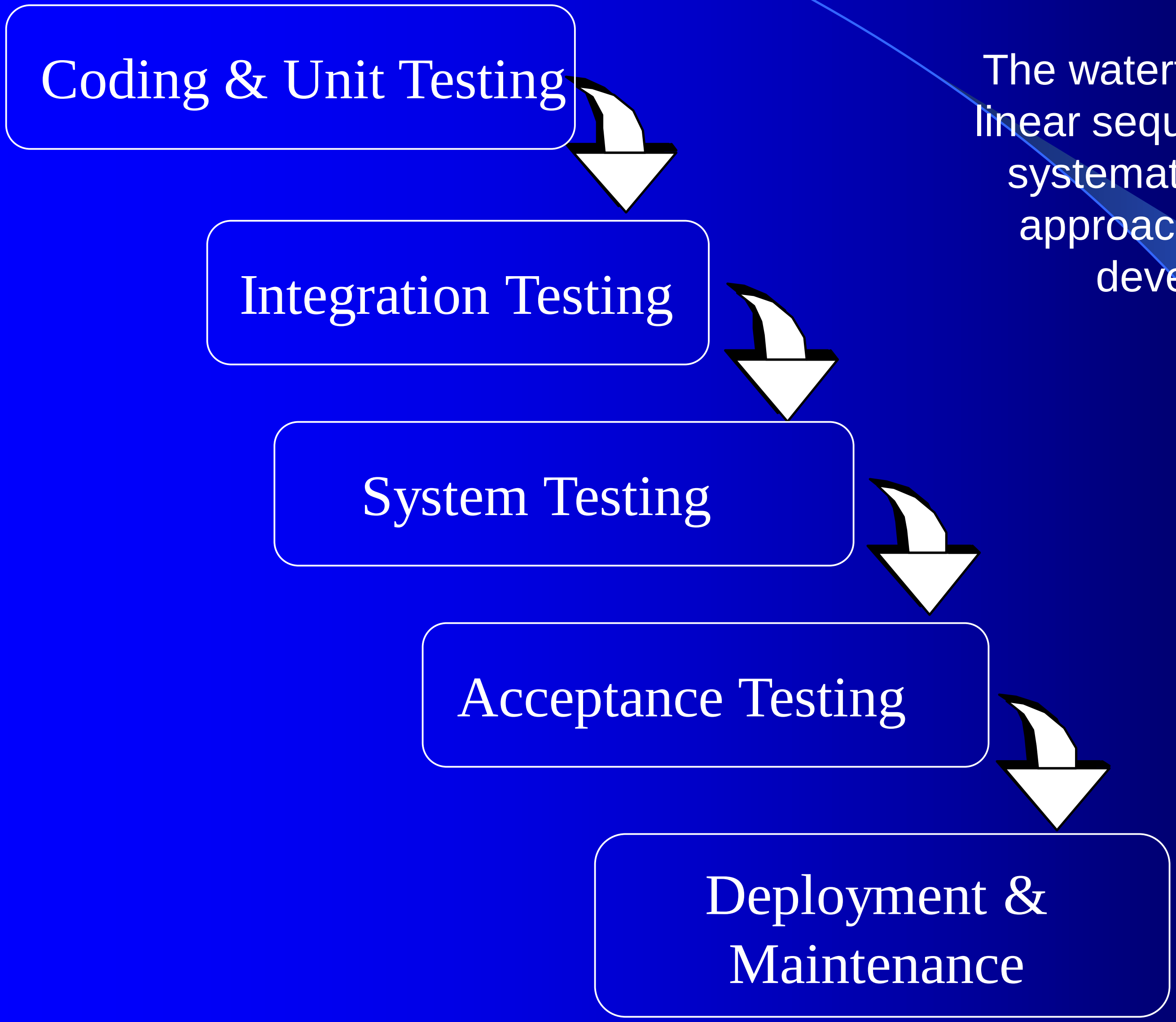
And many more…

The Waterfall Model



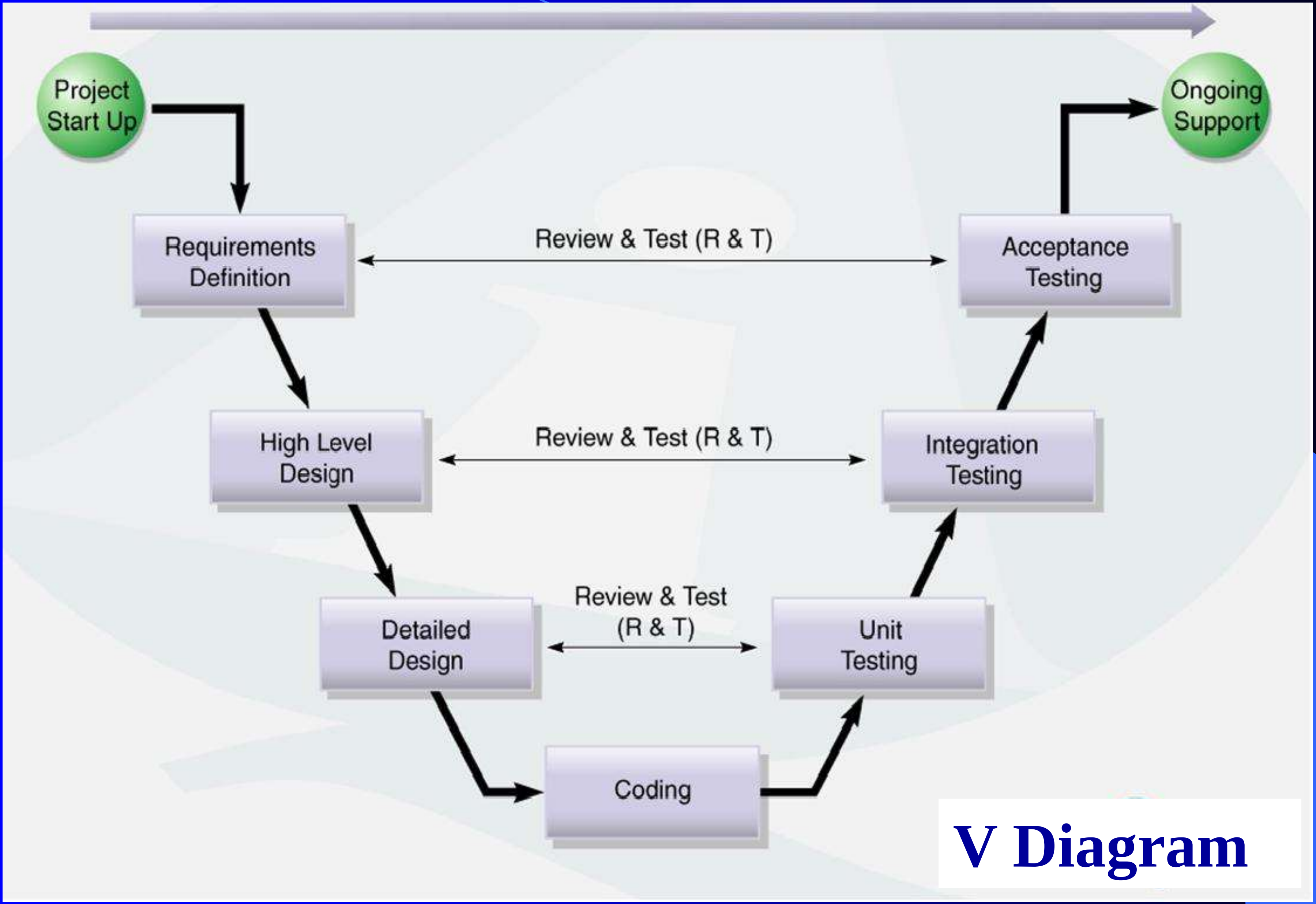
The waterfall model - the linear sequential model, a systematic, sequential approach to software development

The Waterfall Model



The waterfall model - the linear sequential model, a systematic, sequential approach to software development

The V Diagram



The Waterfall Model

Advantages:

- Simple
- Step-by-step
- Focused

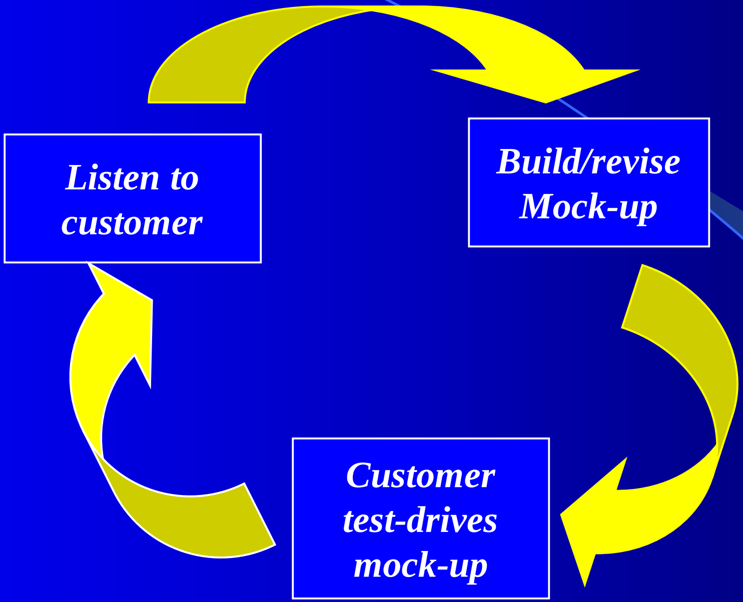
Problems:

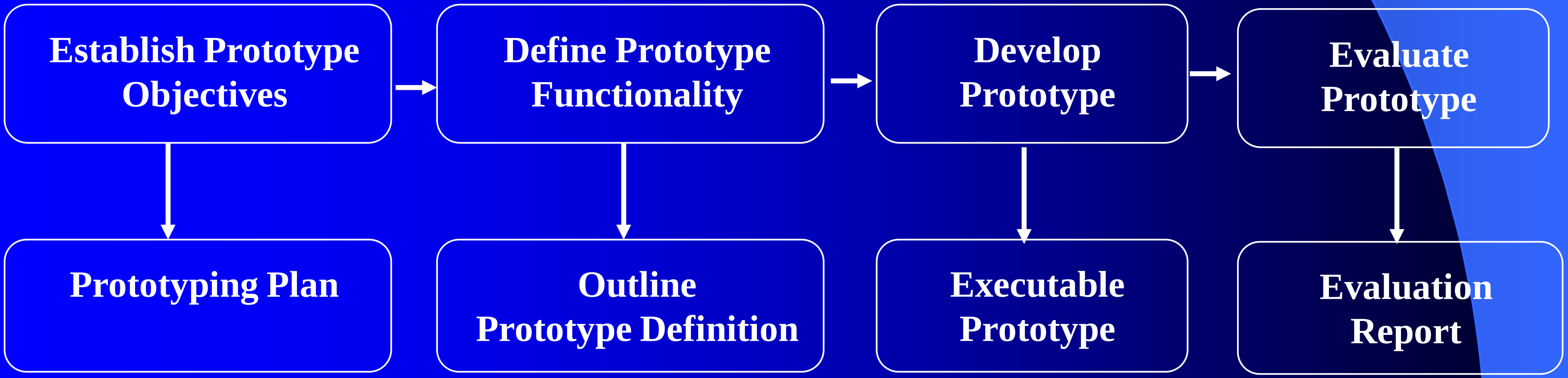
- Impractical - Real projects rarely follow the sequential flow
- Difficult for the end user to state all requirements explicitly at the onset
- Sense of insecurity in end user
- Errors in early stages get magnified

The Prototyping Model

- Evolutionary Prototype
- Throw away Prototype

The Prototyping Model




Establish Prototype Objectives
Define Prototype Functionality
Develop Prototype
Evaluate Prototype
Prototyping Plan
Outline Prototype Definition
Executable Prototype
Evaluation Report

The Prototyping Model

Advantages:

- Concrete definition of end user requirements
- Reduced insecurity in end user
- Suitable in following cases:
  - i) Customer cannot provide the detailed requirements
  - ii) Complicated system-user interactions
  - iii) Use of new technologies
  - iv) Develop new domain application systems

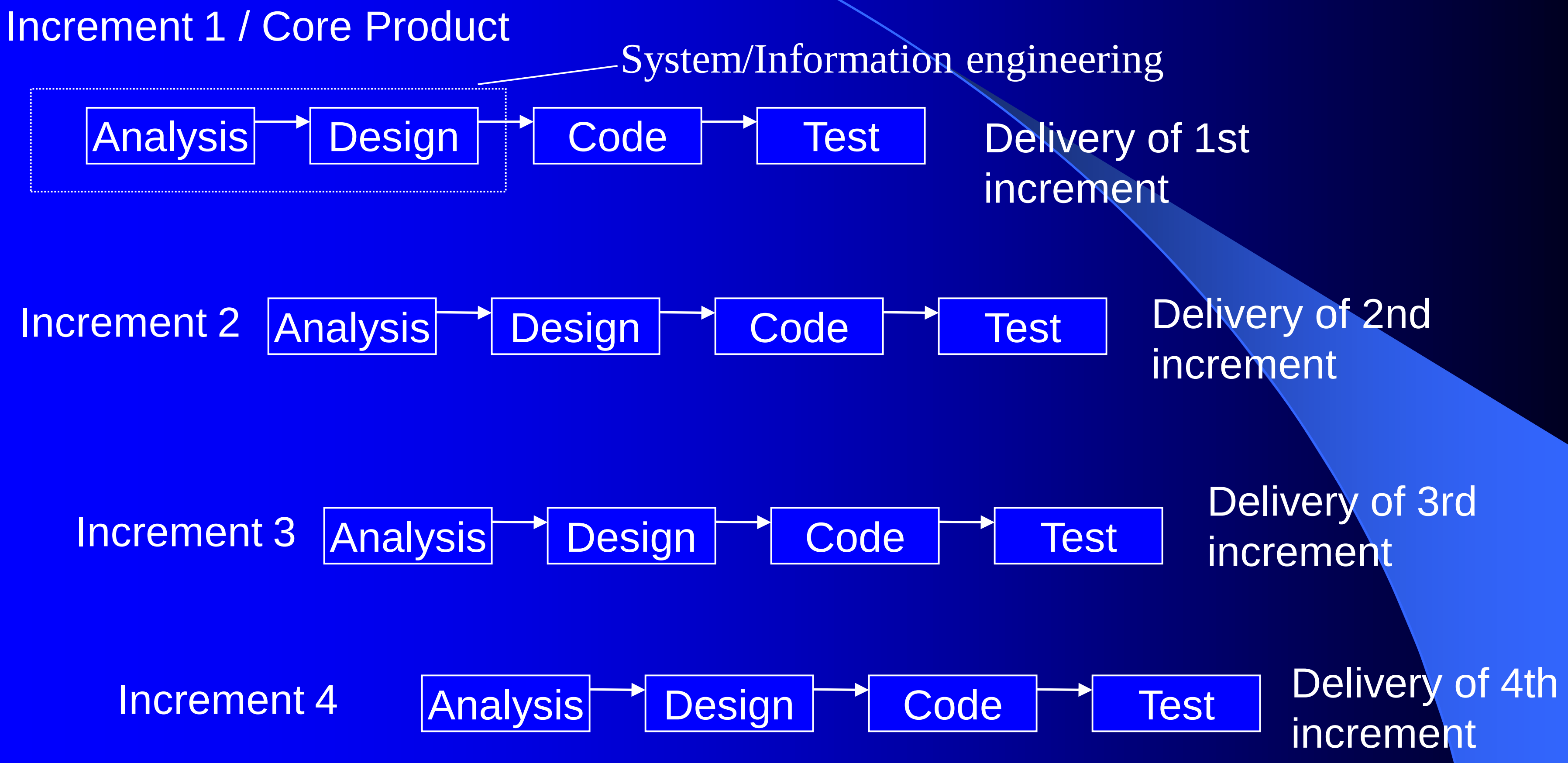
Problems:

- Developers tend to develop the product based on the prototype only
- Developers may compromise in order to get a prototype working quickly
- End user may think that the prototype is the final product

The Incremental Model

- Combines elements of the linear sequential model with the iterative philosophy of prototyping
- Focuses on the delivery of an operational product with each increment
- Useful in case of limited staffing, funds and stringent time constraints

The Incremental Model

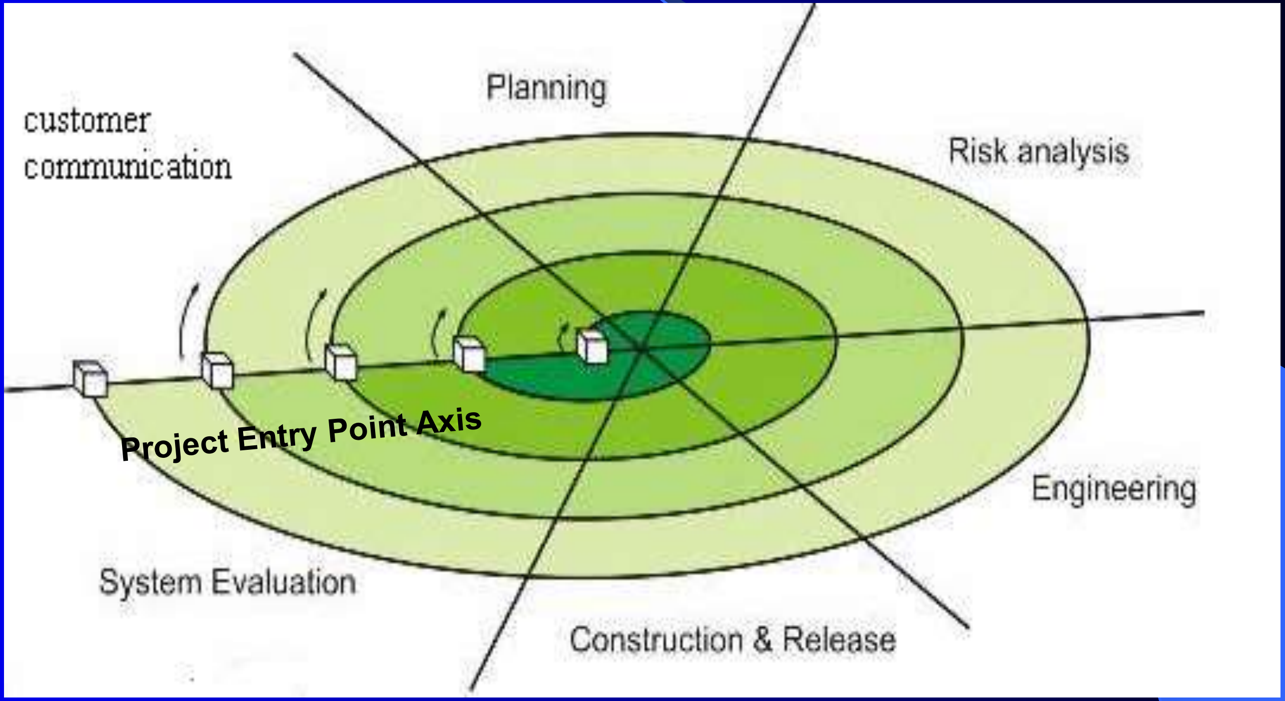

Increment 1 / Core Product
System/Information engineering
Analysis
Design
Code
Test
Delivery of 1st increment
Increment 2
Analysis
Design
Code
Test
Delivery of 2nd increment
Increment 3
Analysis
Design
Code
Test
Delivery of 3rd increment
Increment 4
Analysis
Design
Code
Test
Delivery of 4th increment

The Spiral Model

- Evolutionary software process model
- Allows risk analysis at every stage
- Couples the iterative nature of prototyping model with systematic linear sequential model
- Provides the potential for rapid development of incremental versions of the software

The Spiral Model

🟩	Concept Development
🟩	New Product Development
🟩	Product Enhancement
🟩	Product Maintenance



Software / System Requirement

- Ranges from a high-level abstract statement of a service or of a system constraint to a detailed mathematical / logical functional specification
- Basis for a bid for a contract - must be open to interpretation
- Basis for the contract itself - must be defined in detail

Types of Requirement

- User requirements

  Statements in natural language plus diagrams of the services - written for customers

- System requirements

  A structured document with detailed descriptions of the functions, services and operational constraints - part of a contract between client and developer

Feasibility Analysis

- Technical Feasibility
- Operational Feasibility
- Time Schedule Feasibility
- Economic Feasibility
- Market Feasibility
- Legal Feasibility
- Changing political and international scenario

The Software/System Requirements Specifications (SRS)

- Official statement of system requirements
- Official agreement between developer and end user
- Includes both definition of user requirements and specification of the system requirements
- NOT a design document – says WHAT the system should do rather than HOW it should do it

Artefacts of SRS Phase

- Software Requirement Specifications
- Acceptance and System Test Plan
- Project Management Plan
- Quality Assurance Plan

Design Phase

High Level Design

- Modularisation – Coupling & Cohesion
- Use Case Analysis
- Interoperability
- Scalability
- Maintainability
- Testability

Design Phase

Detailed Design

- Interface Design
- Data Flow Analysis
- Entity Relationship Analysis
- Database Schema
- Object Oriented Analysis & Design

Thank You