Software Product, Process and Process Models

Software Product

- Computer software is the product that software engineers design and build.

➢ It encompasses:

  – Programs.
  – Documents in the form of hard and soft-copy.
  – Data that combine numbers and text, video, and audio information, images.

Overview

- **What? A software process** – as a framework for the tasks that are required to build high-quality software.
- **Who?** Managers, software engineers, and customers.
- **Why?** Provides stability, control, and organization to an otherwise chaotic activity.
- **Steps?** A handful of activities are common to all software processes, details vary.
- **Work product?** Programs, documents, and data.

Characteristics of Software Process

- **Predictability** of a process determines how accurately the outcome of follow- ing that process in a project can be predicted before the project is completed.
- **Support Testability and Maintainability** in the life of software the maintenance costs generally exceed the development costs.
- **Support Change**
- **Early Defect Removal**
- **Process Improvement and Feedback**

What is software engineering?

- IEEE **Definition** :
  - (1) The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  - (2) The study of approaches as in (1) above
- Its a discipline that is concerned with all aspects of software production.

- Software engineers should **adopt**
  - Systematic and organized approach to their work
  - Use appropriate tools and techniques depending on the problem to be solved
  - The development constraints and the resources available
- Apply Engineering Concepts to developing Software
- **Challenge** for Software Engineers is to produce high quality software with finite amount of resources & within a predicted schedule

Software process model

- **Process models** prescribe a distinct set of activities, actions, tasks, milestones, and work products required to engineer high quality software.

  To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools layers and the generic phases.
- This strategy is often referred to as a ***process model*** *or a s**oftware engineering paradigm**.*
- *A process model for software engineering is chosen based on the* nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

Types of Process Model

- Software development lifecycle model (Waterfall Model)
- Prototyping model
- V model
- incremental Model
- Iterative Model
- Spiral Model
- Agile Model

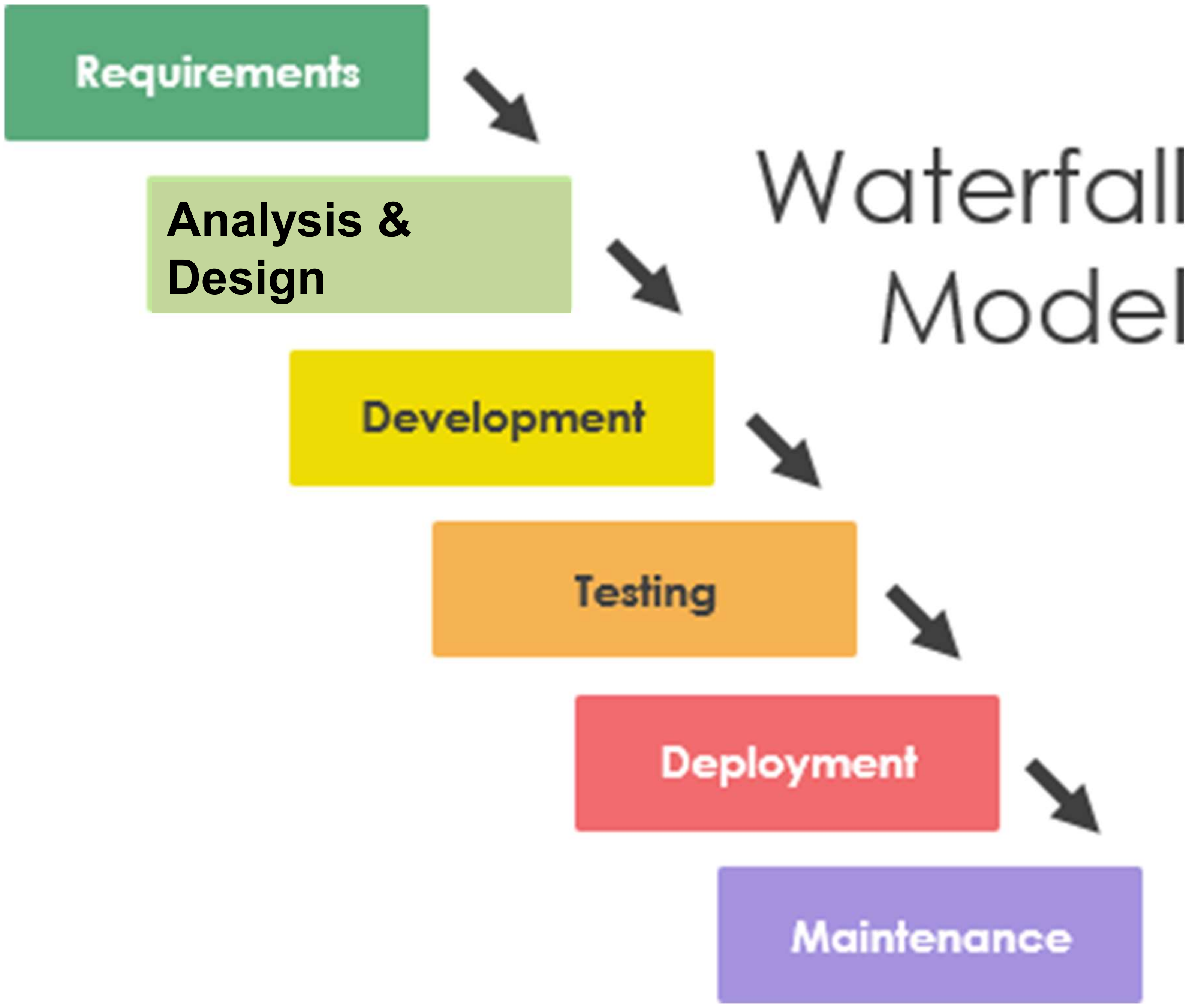
Problem Solving Loop

- **Status quo** “represents the current state of affairs”.
- **Problem definition** identifies the specific problem to be solved;
- **Technical development** solves the problem through the application of some technology,
- **Solution integration** delivers the results (e.g., documents, programs, data, new business function, new product) to those who requested the solution.

Water Fall Model

- Also called the *classic Software life cycle or Linear Sequential model,*
- *S*uggests a systematic, sequential approach to software development.
- It has following **Phases**.

1. Software requirements analysis.
2. Design
3. Code Generation
4. Testing
5. Maintenence


Requirements
Analysis & Design
Development
Testing
Deployment
Maintenance
Waterfall Model

System/information engineering and modelling

- Work begins by establishing requirements for all system elements.
- Leads to allocating subset of these requirements to software.
- This system view is essential when software must interact with other elements such as hardware, people, and databases.

Software requirements analysis

- Is intensified and focused specifically on software.
- Analyst must understand

1. Information domain for the software,
2. Required function,
3. Behaviour
4. Performance
5. Interface.

- Requirements for both the **system** and the **software** are **documented** and **reviewed** with the customer

Design

- A multistep process that focuses on four distinct attributes of a program:

1. Data structure
2. Software architecture
3. Interface representations
4. Procedural (algorithmic) detail

- Design process:
- Translates requirements into software representation that can be assessed for quality before coding begins.
- Is documented and becomes part of the software configuration.

Code generation / Development.

- The design must be translated into a machine-readable form called code generation.
- If design is performed in a detailed manner, code generation can be accomplished mechanistically.

Testing

- Starts after code generation.
- Focus on checking:
- Logical internals of the software:
  - ensuring that all statements have been tested/
- Functional externals
  - conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.
- In testing we perform both verification & validation.

Maintenance/ Support

- Software will undergo change after it is delivered to the customer.
- Change will occur because:

1. Errors have been encountered
2. Software must be adapted to accommodate:
   a) changes in its external
   b) customer needs for functional or performance enhancements.

- Software support/maintenance reapplies each of the preceding phases to an existing program.

Advantages

- This model is simple and easy to understand and use.
- Easy to manage due to the rigidity of the model –
- Phases are processed and completed one at a time.
- Phases do not overlap.
- Works well for smaller projects where requirements are very well understood.

Limitations:

1. **Real projects rarely follow the sequential flow that the model proposes.**

--Although the linear model can accommodate iteration, it does so indirectly.

--As a result, changes can cause confusion as the project team proceeds.

2. It is often difficult for the customer to state all requirements explicitly.

--The linear sequential model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

3. The customer must have patience.

--A working version of the program(s) will not be available until late in the project time-span.

--A major blunder, if undetected until the working program is reviewed, can be disastrous.

4. Freezing the requirements usually requires choosing the hardware

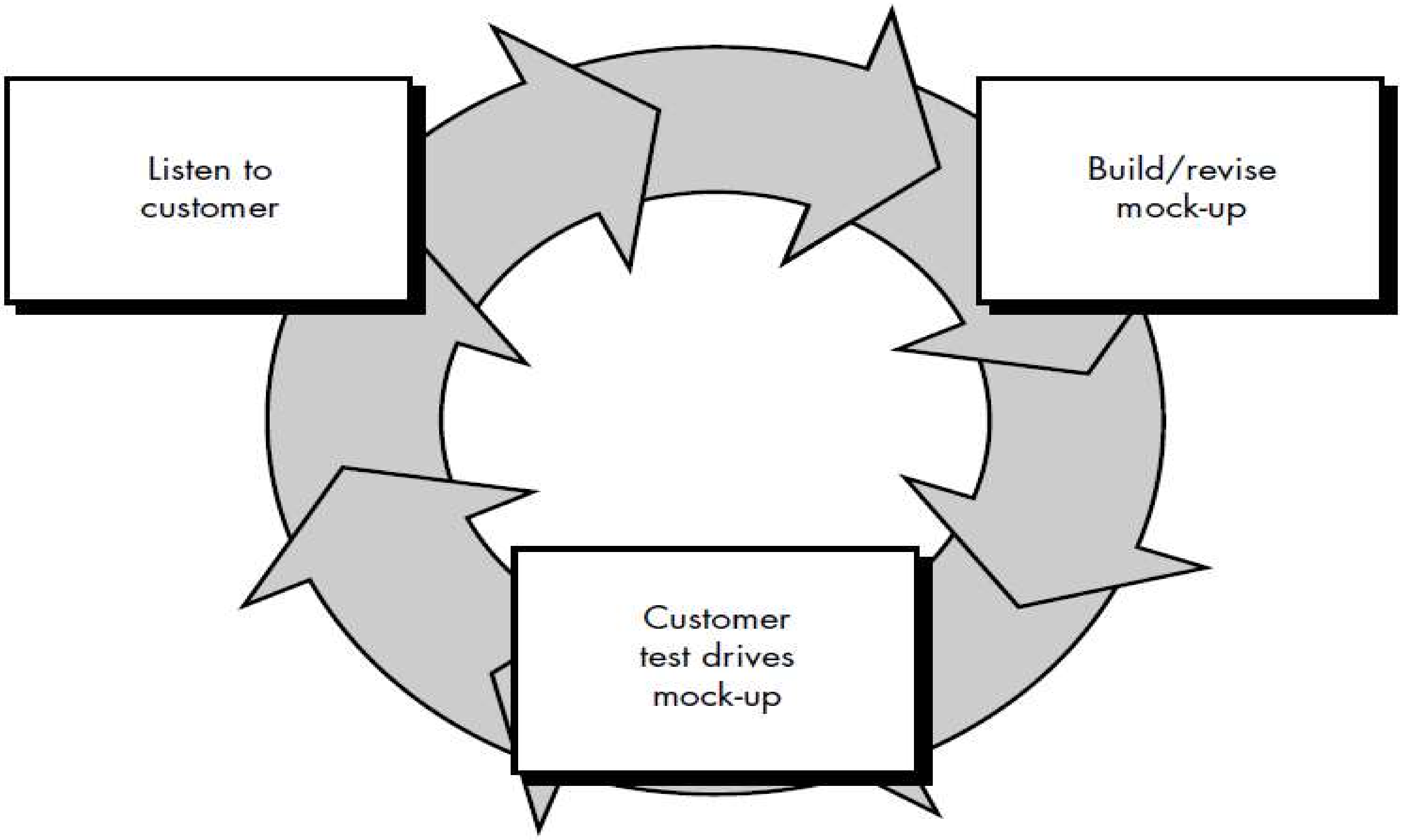
5. It is a document-driven process that requires formal documents at the end of each phase.

PROTOTYPING MODEL

THE PROTOTYPING MODEL

- **Need:**

1. Customer defines a set of general objectives for software but is not able to identify detailed input, processing, or output requirements
2. The developer may be unsure of
   a) Efficiency of an algorithm,
   b) Adaptability of an operating system
   c) Form that human/machine interaction should take.


Listen to customer
Build/revise mock-up
Customer test drives mock-up

The Process:

1. Identify whatever requirements are known, and outline areas where further definition is mandatory.
2. A "quick design" occurs.
3. The quick design leads to the making of a prototype.
4. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed.
5. Iteration occurs as

   Prototype is tuned to satisfy the needs of the customer,

   Enables developer to better understand what needs to be done.

- **The prototype serves as a mechanism for identifying software requirement.**

- Often not used, as it is feared that development costs may become large.
- However, in some situations, the cost of software development without prototyping may be more than with prototyping.
- **There are two major reasons for this:**

1. First, the experience of developing the prototype might reduce the cost of the later phases when the actual software development is done.
2. Secondly, in many projects the requirements are constantly changing, particularly when development takes a long time.

- **Prototyping:**
- Suitable for projects where requirements are hard to determine
- Confidence in the stated requirements is low.
  - Use of waterfall model in such projects leads to requirement changes and associated rework while the development is going on.
- Requirements frozen after experience with the prototype are likely to be more stable.
- Excellent technique for reducing some types of risks associated with a project.

Prototyping can be problematic

Following are reasons:

1. Software quality or long-term maintainability concerns
2. The developer often makes implementation compromises

- Customer and developer must both **agree** that the prototype is built to **serve as a mechanism for defining requirements.**

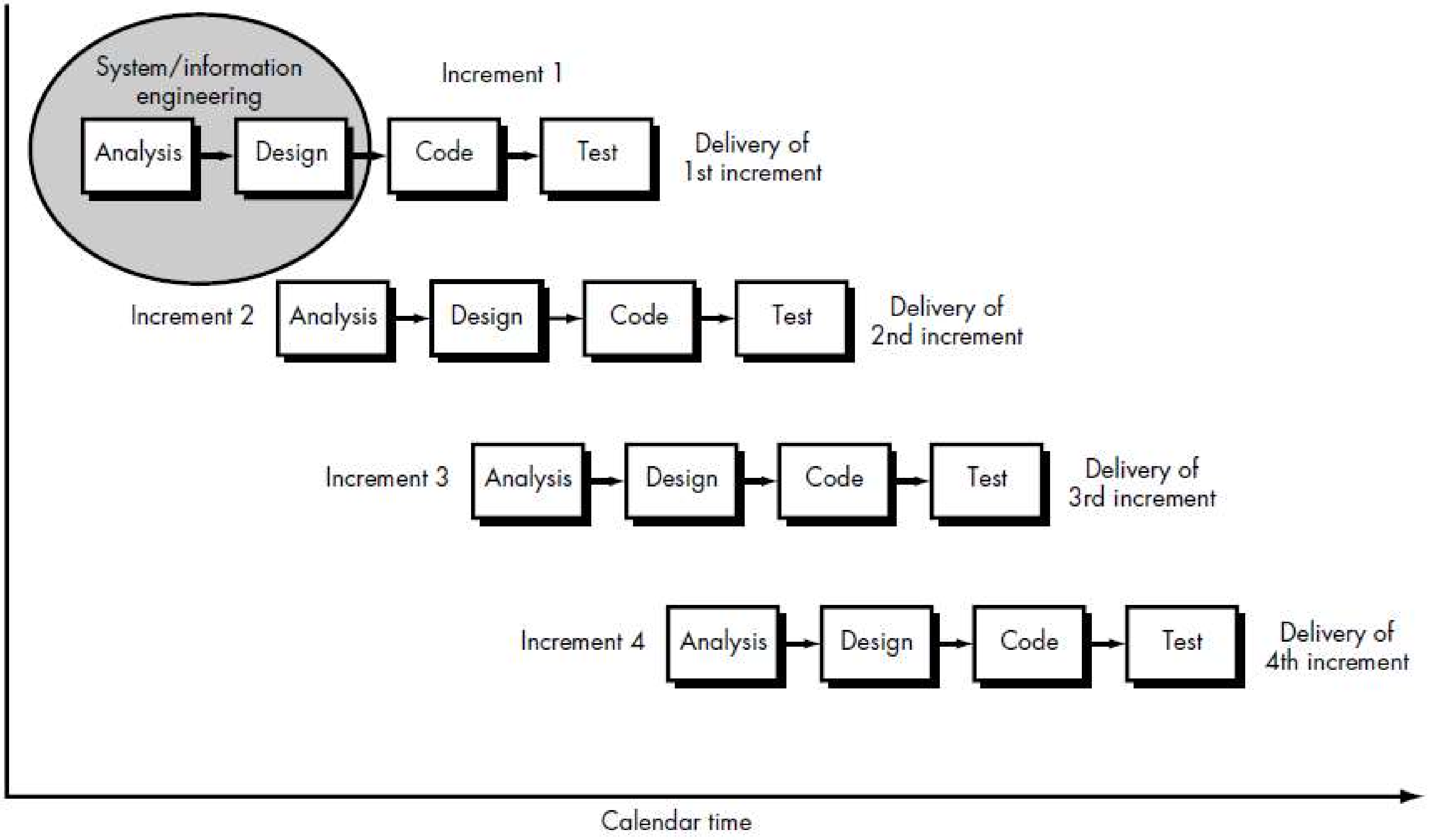
EVOLUTIONARY SOFTWARE PROCESS MODELS

- Evolutionary models are iterative.
- Enables development of increasingly more complete versions of the software.
- Various models that come under this category:

I. The Incremental Model.
II. Spiral Model
III. WINWIN Spiral Model
IV. The Concurrent Development Model

The Incremental Model

- *Combines elements of the linear sequential model (applied* repetitively) with the iterative philosophy of prototyping.
- Each linear sequence produces a deliverable “increment” of the software


System/information engineering
Increment 1
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
Calendar time

The Process

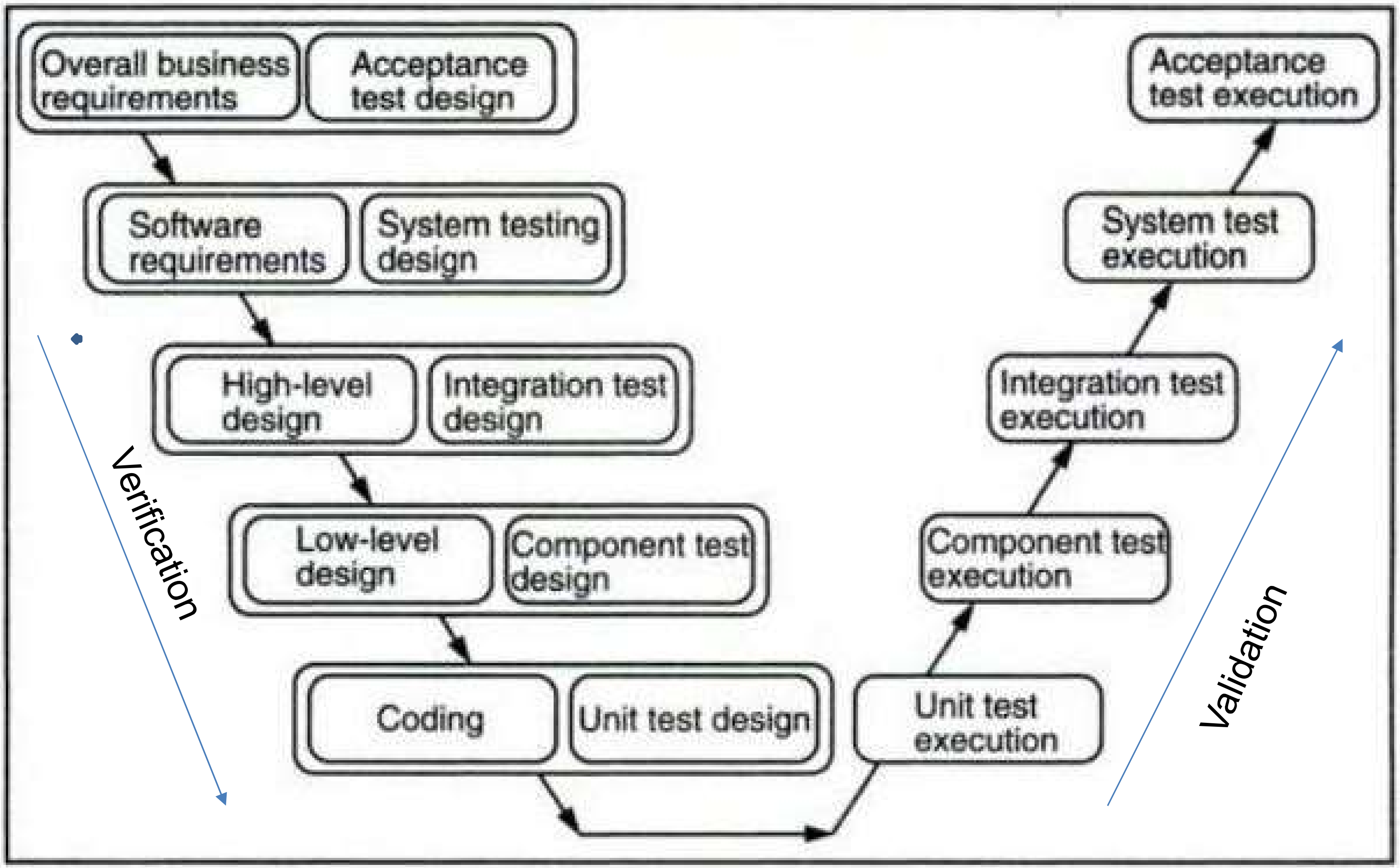
1. First increment is often a *core product.*
2. Basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered.
3. The core product is used by the customer (or undergoes detailed review).
4. As a result of use and/or evaluation, a plan is developed for the next increment.
5. The plan addresses
   a) the modification of the core product to better meet the needs of the customer.
   b) delivery of additional features and functionality.
6. This process is repeated following the delivery of each increment, until the complete product is produced.

- **Focuses on the delivery of an operational product with each increment.**

V- Model

- SDLC model where execution of processes happens in a sequential manner in V-shape.
- Also known as Verification and Validation model.
- Extension of the waterfall model
- Association of a testing phase for each corresponding development stage.
- Highly disciplined model and next phase starts only after completion of the previous phase.

V-Model


Overall business requirements
Acceptance test design
Software requirements
System testing design
High-level design
Integration test design
Low-level design
Component test design
Coding
Unit test design
Unit test execution
Component test execution
Integration test execution
System test execution
Acceptance test execution
Verification
Validation

- **Requirements** like BRS and SRS begin the life cycle model
- Before development starts, a system test plan is created.
- Test plan focuses on meeting the functionality specified in the requirements gathering.
- **The high-level design (HLD)** phase focuses on system architecture and design.
- Provide overview of solution, platform, system, product and service/process.
- Integration test plan is created to test the pieces of the software systems ability to work together.

- **Low-level design (LLD)** The actual software components are designed.
- Defines the actual logic for each and every component of the system.
- Component tests are created.
- **Coding phase:** All coding takes place here.
- Output:
- No. of program units,
- Individual program units need to be tested independently before they are combined to form components.

Advantages of V-model:

1. Simple and easy to use.
2. Testing activities like planning, test designing happens well before coding, saving time
3. Higher chance of success over the waterfall model.
4. Proactive defect tracking – that is defects are found at early stage.
5. Avoids the downward flow of the defects.
6. Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

1. Very rigid and least flexible.
2. No early prototypes of the software are produced.
3. If any changes happen in midway, then the test documents along with requirement documents has to be updated.

When to use the V-model:

- For small to medium sized projects where requirements are clearly defined and fixed.
- When ample technical resources are available with needed technical expertise.
- High customer confidence is required.

- As shown in following figure, for each type of test we move the design of tests along with the actual activities and retain the test execution after the product is built.

Questions

1. Is there ever a case when the generic phases of the software engineering process don't apply? If so, describe it.
2. Which of the software engineering paradigms presented here do you think would be most effective? Why?
3. Provide five examples of software development projects that would be amenable to prototyping. Name two or three applications that would be more difficult to prototype.
4. Propose a specific software project that would be amenable to the incremental model. Present a scenario for applying the model to the software.