Software Engineering

A Quick Recap of Basic Concepts of Software Engineering and SDLCs

Software Engineering

- Introduction to Software Engineering
  - Software Process
  - Software Process Model
  - Software Product
- Importance of Software Engineering
- Software Development Life Cycle

What is Software

Software = Computer Programs (Instructions) + Data + (Procedures + Rules) + Associated Documents

(Procedures + Rules) include installation instructions, and user guide.

Associated Documents can be training manuals and other supplementary documents.

Question 1

Which of the following is not included in the definition of software

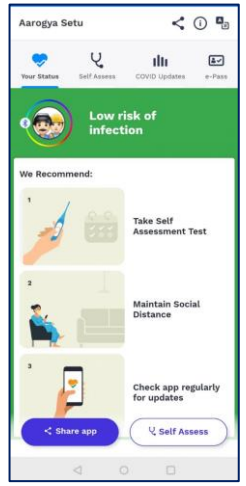
A. Programs or instructions
B. Data
C. Extended Drive
D. Installation Guide

Answer: C

Examples of Software

1. Operating Systems – MS Windows, Linux
2. Device Drivers – for Printer and other devices
3. Compilers, Debuggers – C++ compiler/debugger
4. Database Software – MySQL, Oracle RDBMS
5. Office Software - MS Word, MS Excel, MS Power Point
6. Web Related – Web Server, App Server
7. Browsers – MS Edge, Google Chrome
8. Graphic Design Tools – Adobe Photoshop
9. Project Management Tools – MS Project
10. Agile Lifecycle Management Tools – Atlassian JIRA, VersionOne
11. ERP Systems – SAP, Oracle Applications
12. Mobile Apps – Arogya Setu, PayTm, Gpay, WhatsApp
13. Applications Software – Payroll Application, Library Management System

Software Everywhere!



MOBILE APP

PRODUCTS & UTILITIES

WEB APPLICATIONS

ERP SYSTEMS

COMPILERS, DEBUGGERS

OFFICE SOFTWARE, DESIGN TOOLS, PM TOOLS

WEB SERVERS, BROWSERS

DB ENGINES

OPERATING SYSTEM

DEVICE DRIVERS

HARDWARE

DEVICES

General Software Characteristics

It should be

- **Complete** (should meet all requirements)
- **Exclusive and reliable** (should deliver solution to a particular problem and remain stable)
- **Precise** (No unwanted steps or operations. Leads to resource optimization, efficiency and cost reduction)
- **Efficient** (should produce optimal results)

Question 2

Which of the following is not a characteristic of software?

A. Exclusive and Reliable
B. Precise
C. Cost-Effective
D. Efficient

Answer: C

Software Engineering

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

"engineering discipline" - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

"all aspects of software production" - Not only technical process of development, but also project management and the development of tools, methods etc. to support software production.

Software Engineering

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

ENGINEERING DISCIPLINE

- Requirement Engineering
- Technical Architecture
- Solution Design
- Coding/Programming
- Testing/Quality Assurance
- Production Release, Monitoring
- Software Maintenance
- End-User Support, …

ALL ASPECTS OF SOFTWARE PRODUCTION

- Market Analysis
- Competitor Analysis
- Project Management
- Software Economics (Cost, and other expenditure)
- User Acceptance and Adoption
- User Satisfaction Surveys, …

Importance of Software Engineering

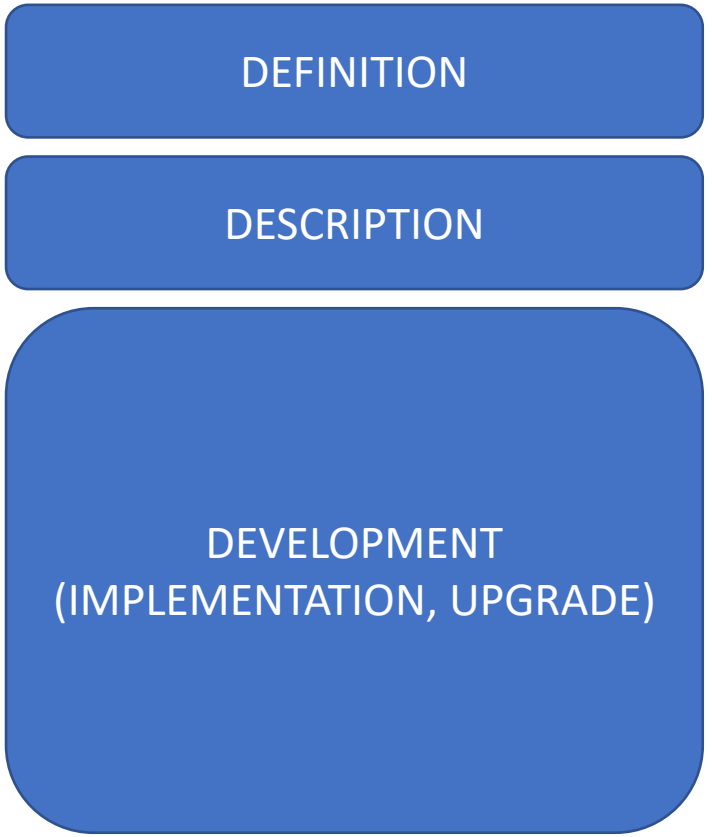
All nations and most of their citizens rely on advanced software systems. It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.

For most types of system, the costs of changing the software is significant after it has gone into use. A software requirements related issue or defect identified in production costs 10 to 100 x times as compared to the cost of fixing it at the initial stage itself. Software engineering

- Provides a strong foundation to software development
- Helps understand the significance of customer needs and collaboration
- Helps understand software life cycle
- Includes the best practices of software development (E.g. Test data coverage, test data quality, …)
- Helps produce reliable, high-quality, trustworthy systems economically and quickly

Software Process

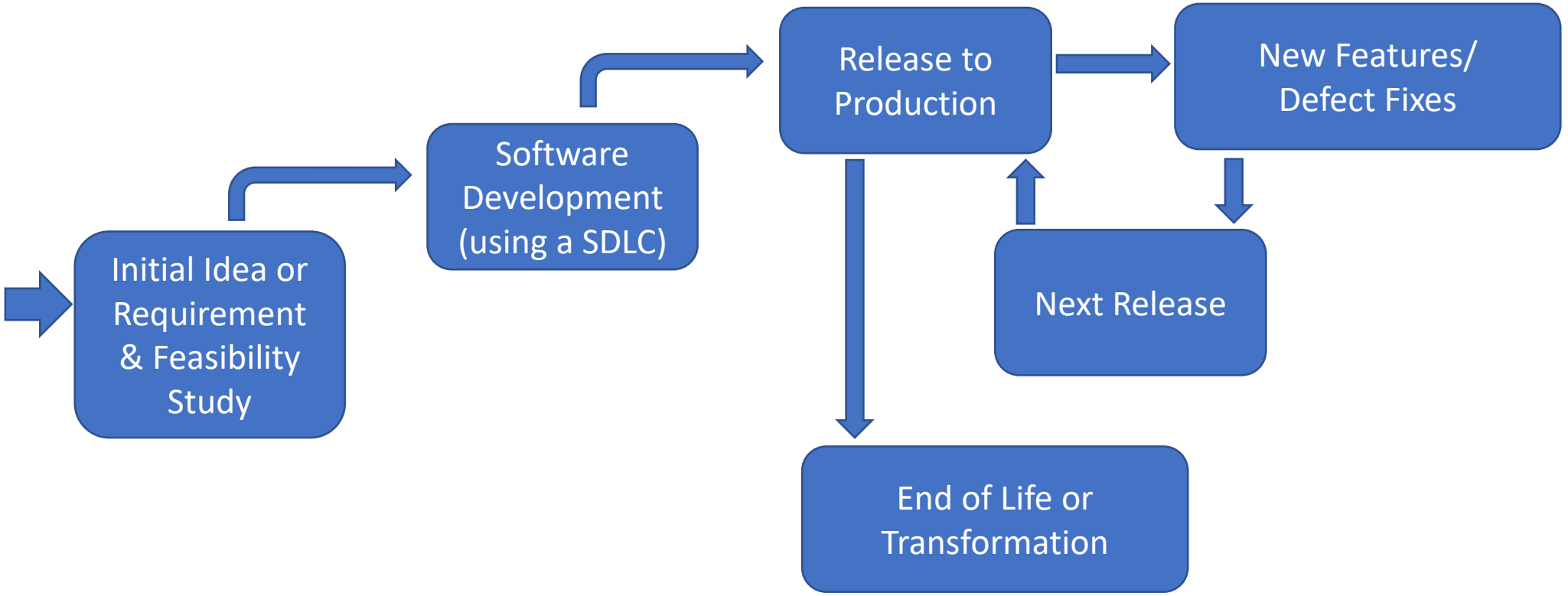
- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.



WHAT
Objectives,
Outcomes,
Scope

HOW
Process
Approach
Data/Control
Flow

The Life of Software



General Software Development Phases

- Requirement Analysis Phase
- Designing Phase
- Coding Phase
- Testing Phase
- Implementation (User Acceptance & Production Rollout) Phase
- Maintenance Phase

Question 3

In which phase of the software development developers write code and perform unit testing?

A. Requirement Analysis Phase
B. Designing Phase
C. Coding Phase
D. Testing Phase

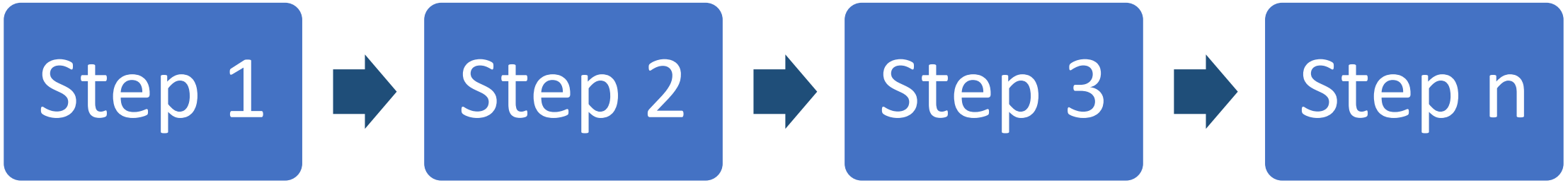
Answer: C

Software Process

- Linear Sequential Model
- Layered Model
- Prototyping Model
- RAD (Rapid Application Development) Model
- Process Framework
- Assessing & Improving the Capability: CMMi

Linear Sequential Model

Also known as Predictive Life Cycle Model or Classical Life Cycle Model



ADVANTAGES

- Easy to understand and implement
- Widely used
- Identifies deliverables and milestones upfront

DISADVANTAGES

- Assumes that requirements can be baselined(frozen) before the Design phase
- Each steps takes its own time (for e.g. Requirements Analysis)
- Specialists run each step (difficult in large projects)
- Handoff after each step to a different group of people

Layered Model

Provide support to processes and methods by making the tasks automated or semi-automation

Provide technical capabilities or guidance specific to requirement analysis, design, development, testing

Ensure timely development of the software system

A strong base of quality processes is required to define engineering processes at the next level

Prototyping Model

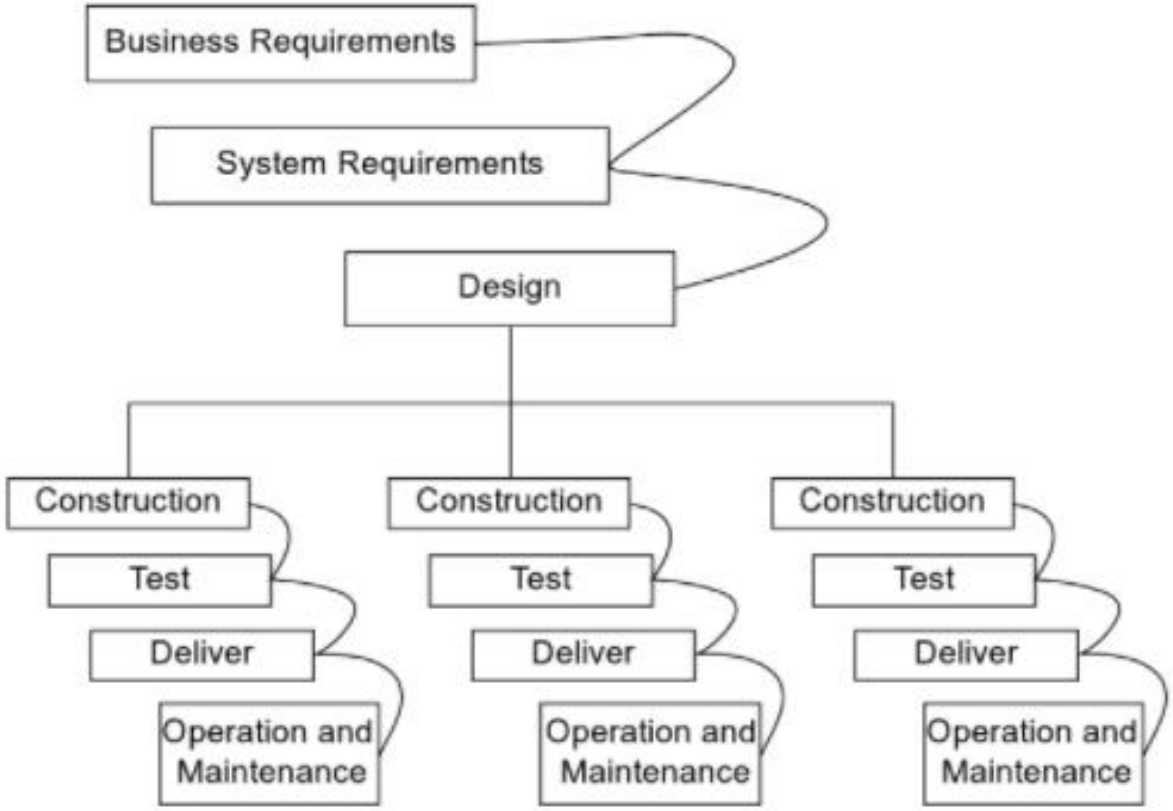
1. Gather requirements, perform a quick design and build a prototype
2. Demonstrate the prototype to end users or customer and gather feedback
3. Refine requirements based on feedback
4. Focus on design, implementation and testing

Two types of prototypes

EVOLUTIONARY PROTOTYPE: Has a good architecture/design foundation and can evolve into a product. Changes can be incorporated to evolve the prototype. The investment and efforts to build the prototype can be leveraged to build the product.

THROWAWAY PROTOTYPE: Will have to be discarded or thrown away after the demo because it does not have solid design or end-to-end implementation. Cannot be leveraged to build the product.

RAD (Rapid Application Development) Model



FIVE PHASES

1. Business Modeling
2. Data Modeling
3. Process Modeling
4. Application Generation
5. Testing and Turnover

- Requires a very short span of time (60 to 90 days) to deliver software.
- Software construction, testing, delivery and operations & maintenance are executed in parallel by two or more teams.

Process Framework

- Process framework is a collection of related processes
- Process framework of a software application can be seen as a collection of processes that deliver the overall vision together
- Some basic processes (or functionalities) are common across multiple software products (for example, login process, shopping cart management process)
- These actions can be applied to every product without any further modifications

Assessing and Improving the Capability

CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

CMMI is a process model for improving the process capability and performance related to

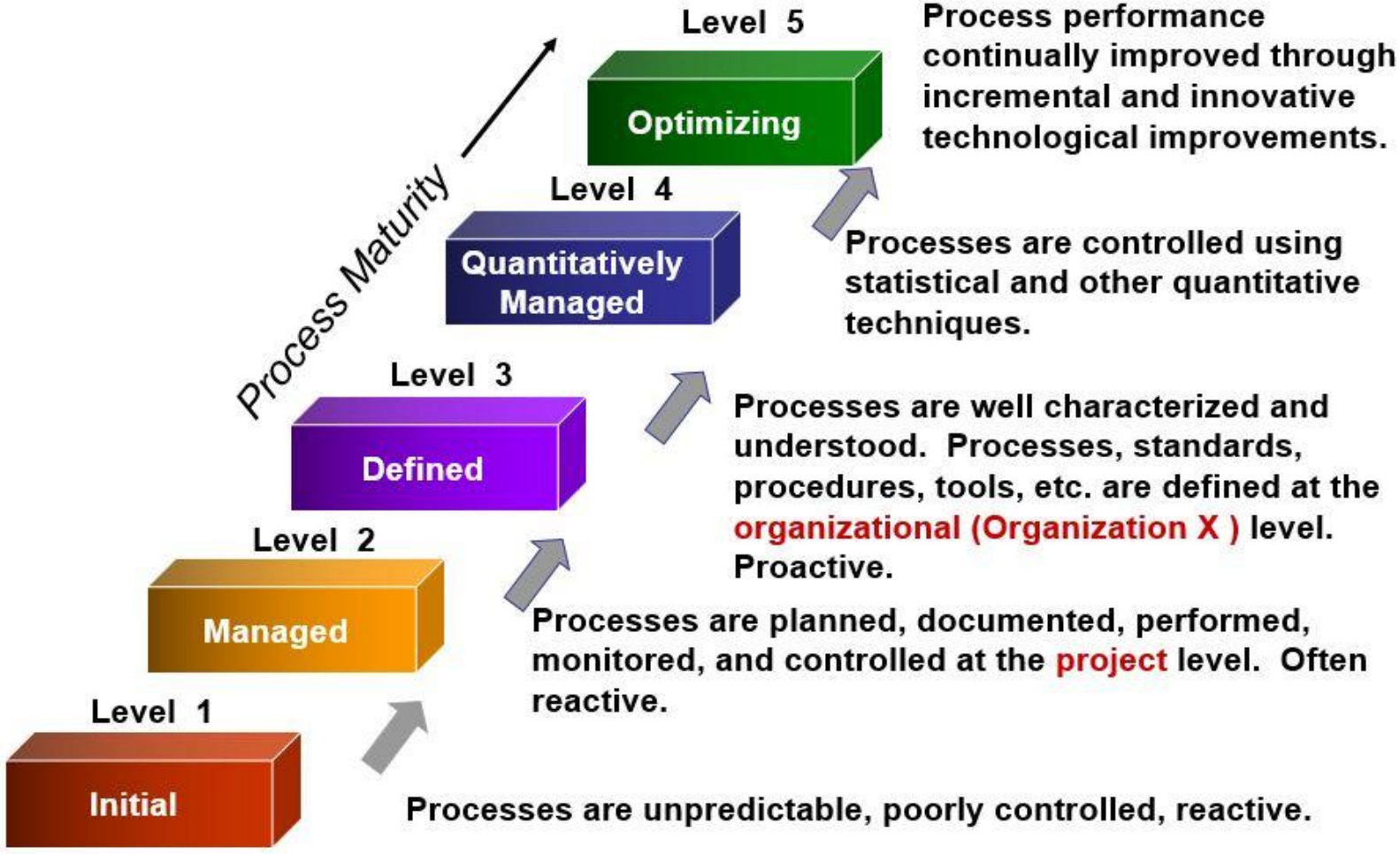
- Product and Service Development
- Service Establishment and Management
- Product and Service Acquisition

CMMI evolved from the software CMM.

CMMI has five levels of maturity.

People Capability Maturity Model (PCMM) is another model that deals with the improvement of human resources (People Assets) of the organization.

5 Levels of CMMI



Software Process Models

Software process models include systematic methods and steps for software development (from initial requirement analysis to maintenance).

Software process models help the project team to focus on

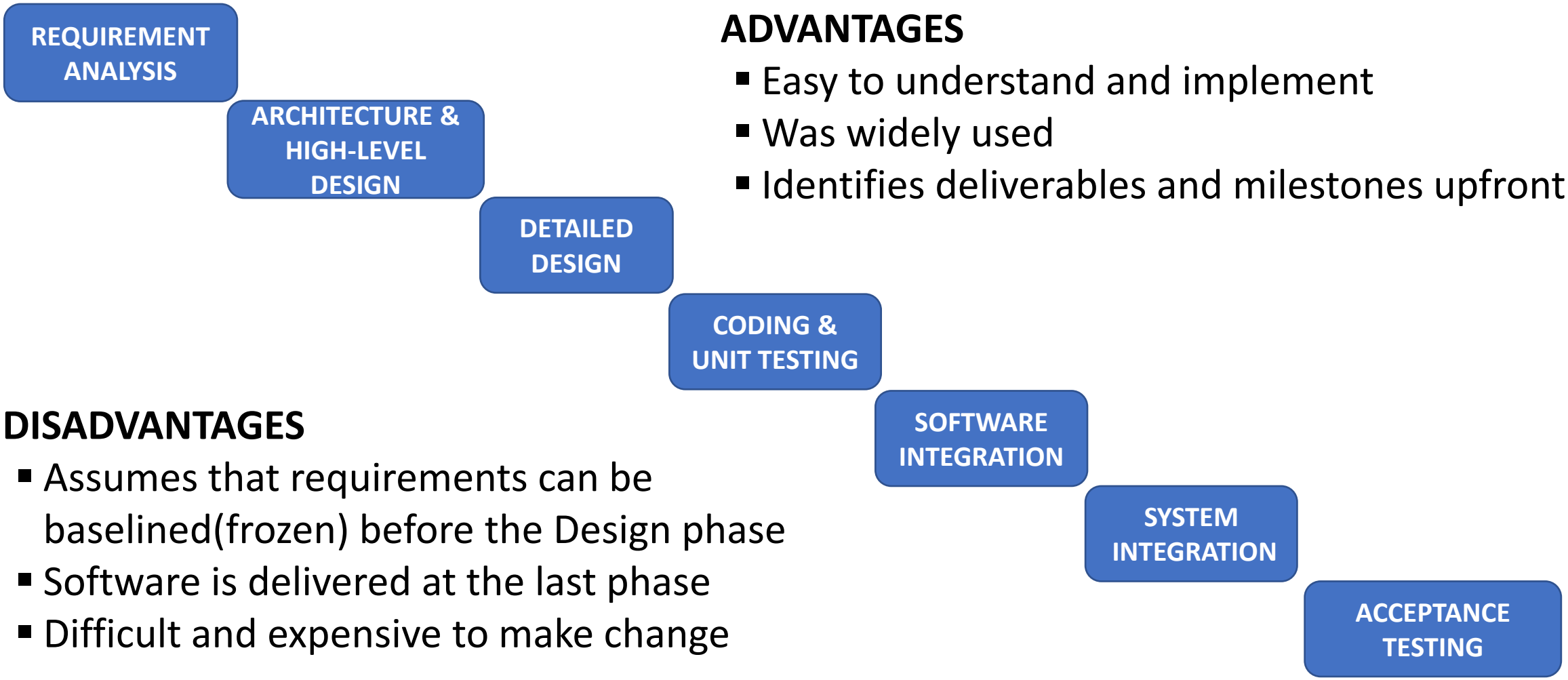
1. Planning and estimation
2. Identification of tasks and sub-tasks to accomplish a milestone or phase
3. Identify and manage risks, issues, constraints and dependencies

Categories of Software Process Models

(SDLC – Software Development Life Cycle)

1. Traditional Software Process Models
   a) Waterfall Model
   b) Incremental Process Model
2. Evolutionary Process Models
   a) Spiral Model
3. Component-based Development Model
4. The Formal Methods Model

Waterfall Model (devised by Royce in 1970)



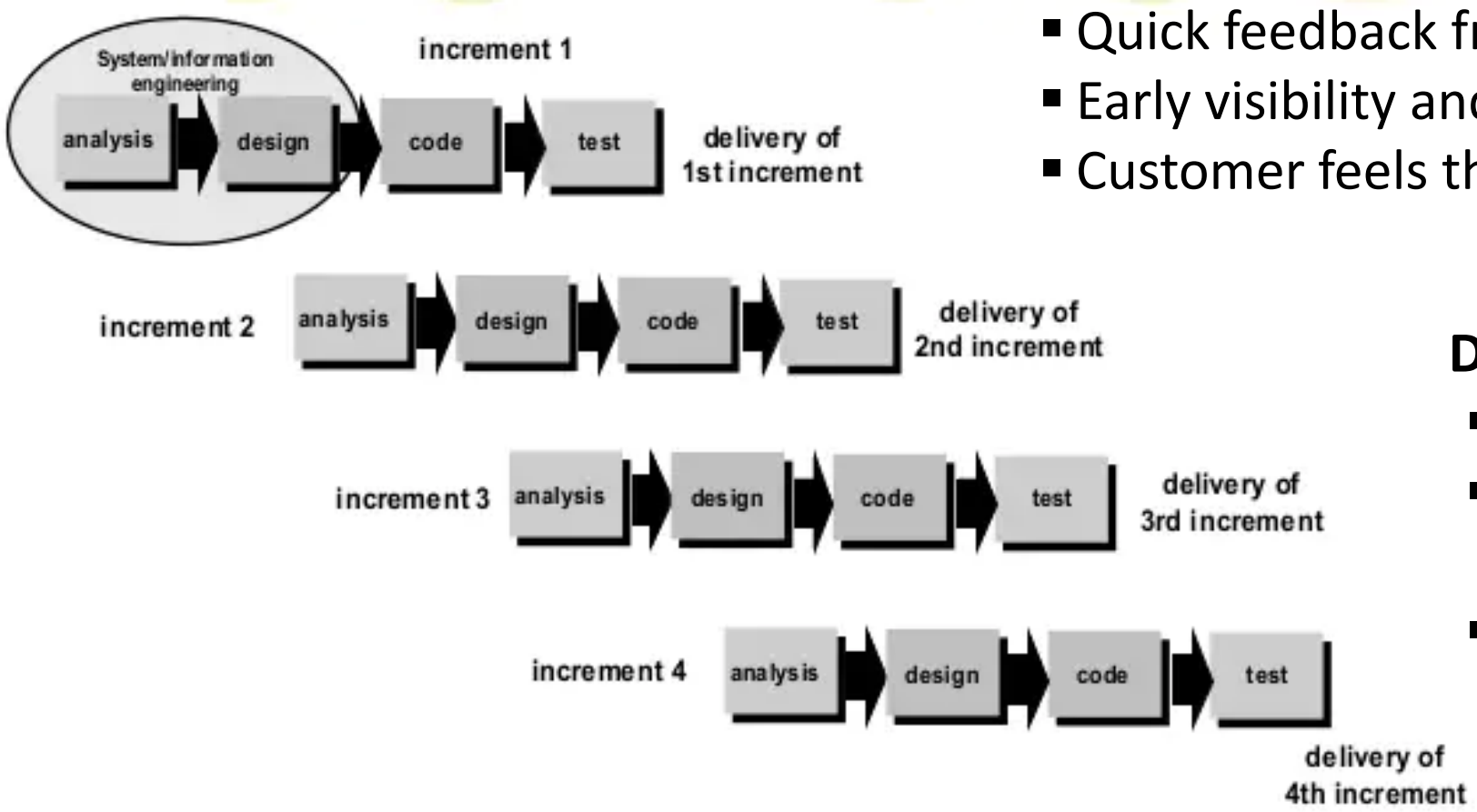
ADVANTAGES

- Easy to understand and implement
- Was widely used
- Identifies deliverables and milestones upfront

DISADVANTAGES

- Assumes that requirements can be baselined(frozen) before the Design phase
- Software is delivered at the last phase
- Difficult and expensive to make change

Incremental Process Model



ADVANTAGES

- Quick feedback from business stakeholders
- Early visibility and customer feedback
- Customer feels that product is delivered early

DISADVANTAGES

- Scheduling is difficult
- Tracking and monitoring is challenging
- Testing of each part and final product becomes exhaustive

Spiral Model

ADVANTAGES

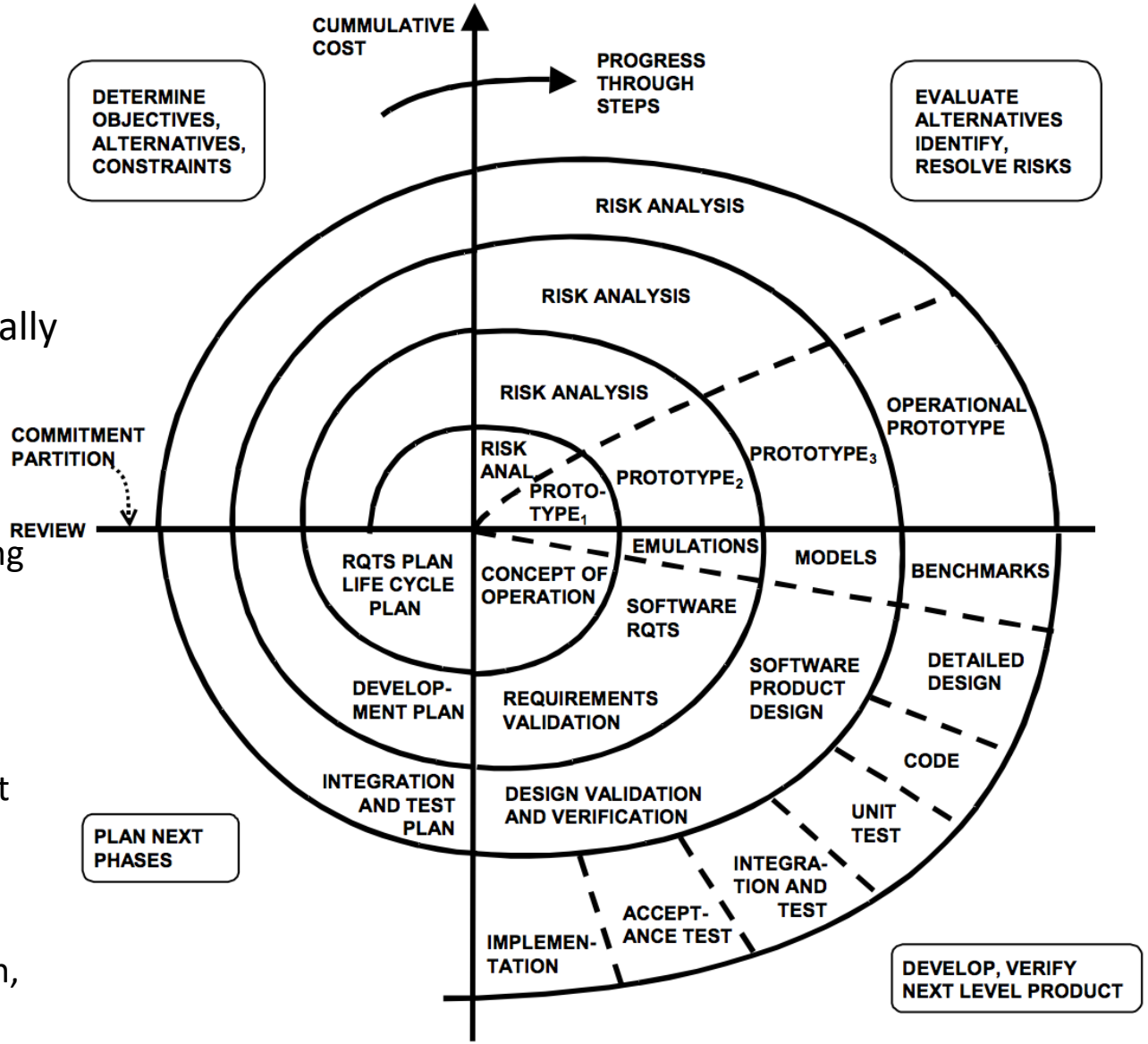
- Risk analysis at the early stage
- Prototyping enhances customer feedback
- Reduces rework as the product matures gradually

DISADVANTAGES

- Very challenging to implement

WIN-WIN SPIRAL MODEL

- Defines a set of negotiation activities at the beginning of each pass around the spiral
- 3 ANCHOR POINTS
  - Life Cycle Objectives (LCO) – Defines a set of objectives for each major SE activity
  - Life Cycle Architecture (LCA) – Establishes goal that must be met by the system/software Architecture
  - Initial Operational Capability (IOC) – Represents a set of objectives associated with operational activities (site preparation, installation/distribution, support, etc.)



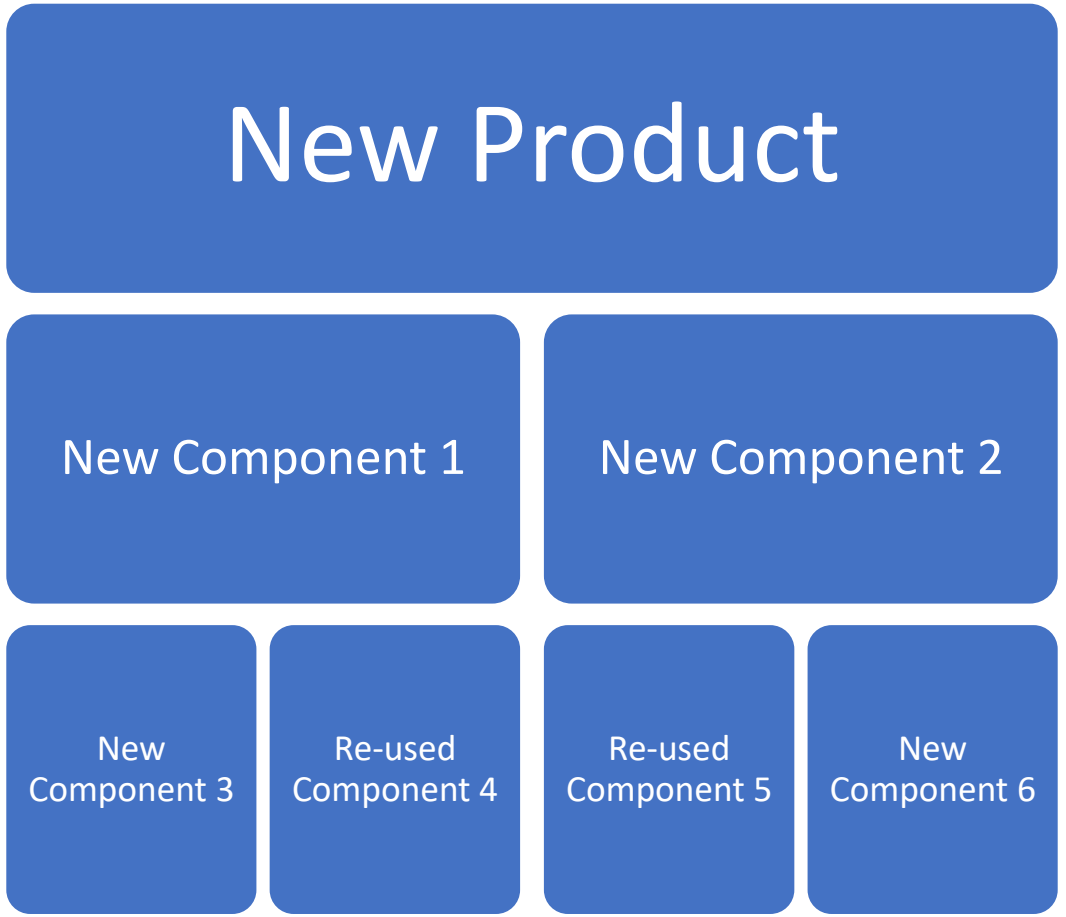
Component-based Development Model

ADVANTAGES

- Component Reuse
- Flexibility in adding new features
- Faster and cheaper
- Modifications are relatively easier

DISADVANTAGES

- Managing dependencies and integration with preceding and succeeding activities



Formal Methods Model (FMM)

FMM is concerned with the application of a mathematical technique to design and implement the software. FMM lays the foundation for developing a complex system and supporting the program development.

ADVANTAGES

1. Discovers ambiguity, incompleteness, and inconsistency in the software.
2. Offers defect-free software.
3. Formal specification language semantics verify self-consistency.

DISADVANTAGES

1. Time consuming and expensive.
2. Difficult to use this model with non-technical stakeholders
3. Extensive training is required to implement this model.

Question 4

Which of the following is not a disadvantage of waterfall model?

A. Assumes that requirements can be baselined(frozen) upfront
B. Software is delivered at the last phase
C. Customers provide frequent feedback on demos
D. Difficult and expensive to make change

Answer: C

Question 5

Which of the following is an advantage of spiral model?

A. Assumes that requirements can be baselined(frozen) upfront
B. Software is delivered at the last phase
C. Risks are identified and analyzed at an early stage
D. Difficult and expensive to make change

Answer: C

Software Product

Characteristics of a Good Software Product

- **Completeness** - Should cover all the requirements
- **Consistency** – In usage and operations
- **Durability** – In diverse environments (different countries, regions)
- **Efficiency** – Optimal utilization of resources (CPU, memory) and performance (speed)
- **Security** – Customer data should be confidential, private and secured
- **Interoperability** – Ability to interface and communicate with other processes, environments, products

Software Application and Software Product

How do they differ?

- **Genesis** (How are they identified? Who supplies the requirements?)
- **User Base** (Who are the users? Does the user base expand?)
- **Quality**
  - Flexibility (ability to customize and install for different user groups / regions)
  - User Experience & Usability
  - Performance and Scalability
  - Security
  - Compatibility and Interoperability

Summary

- Introduction to Software Engineering
  - Software Process
  - Software Process Model
  - Software Product
- Importance of Software Engineering
- Software Development Life Cycle