Chapter 2 – Software Processes

Topics covered

- Software process models
- Process activities
- Coping with change
- The Rational Unified Process
  - An example of a modern software process.

The software process

- A software process is set of related activities that leads to the production of a software product

- Many different software processes but all involve:
  - ***Specification*** – **The functionality of the software and constraints on its operation must be defined.**;
  - ***Design and implementation*** – **defining the organization of the system and implementing the system**;
  - ***Validation*** – checking that it **does what the customer wants;**
  - ***Evolution*** – **changing the system** in **response to changing customer needs.**

- **A software process model** is a ***simplified representation of a process.*** It presents a description of a process from some particular perspective.

Software process descriptions

- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities

- Process descriptions may also include:
  - **Products,** which are the ***outcomes of a process activity***;
  - **Roles**, which reflect the ***responsibilities of the people involved in the process; Ex: Project manager, configuration manager, programmer, etc***
  - **Pre- and post-conditions,** which are ***statements that are true before and after a process activity*** has been enacted or a product produced.

Software Process Categories

Plan-driven and agile processes

- **Plan-driven processes** are processes where all of the process activities are ***planned in advance*** and ***progress is measured against this plan***.

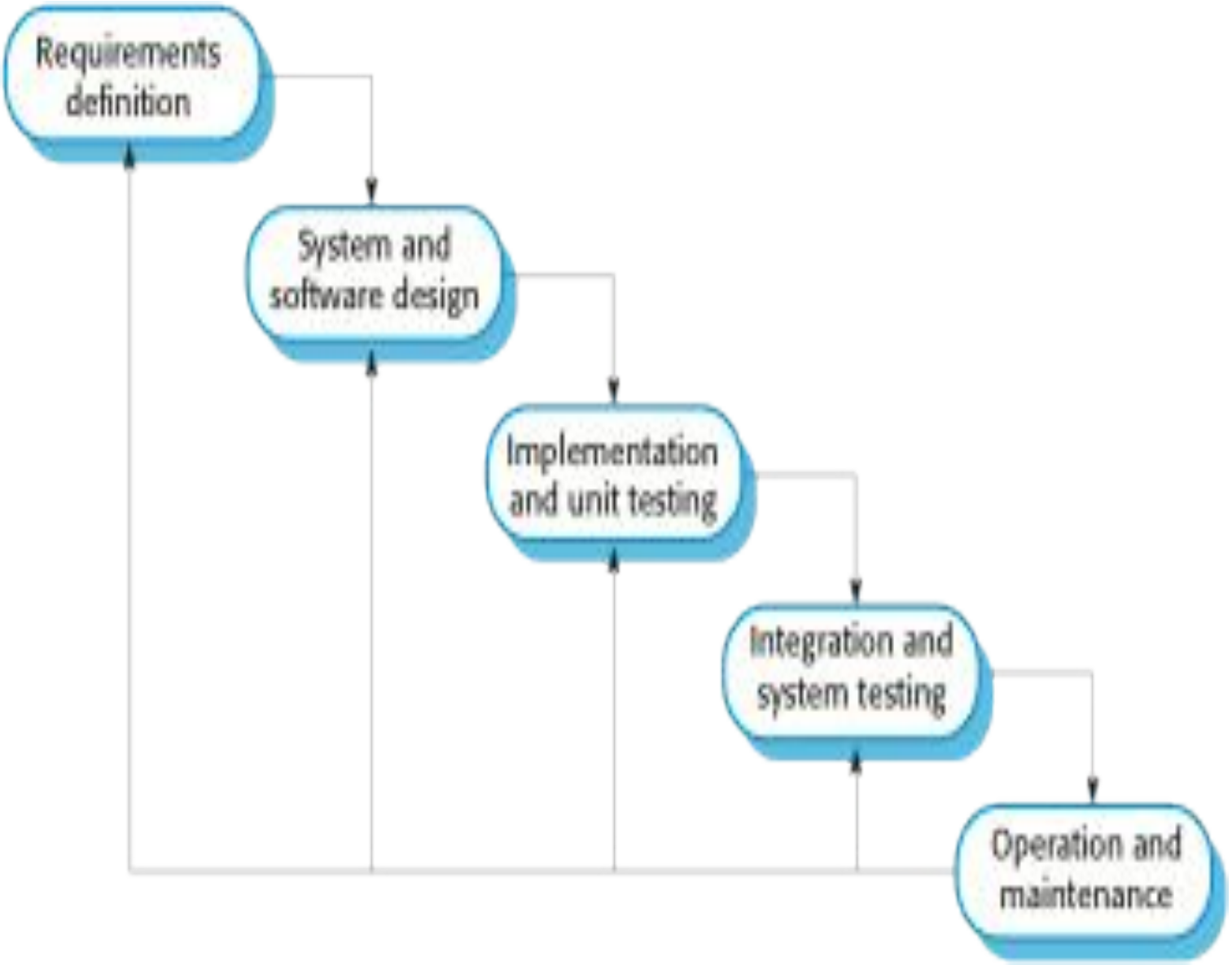
- **In agile processes**, ***planning is incremental*** and ***it is easier to change the process*** to reflect ***changing customer requirements***.

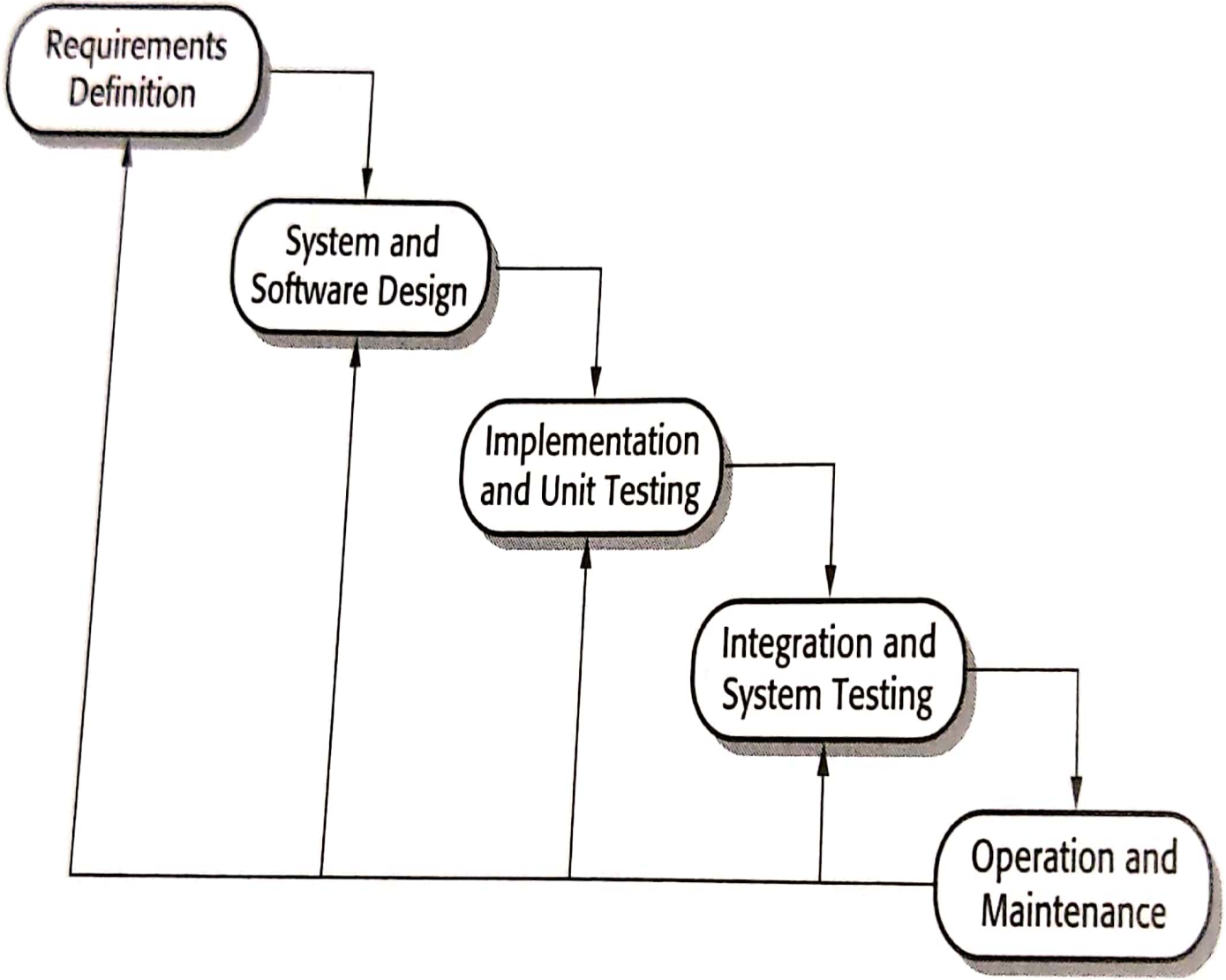
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

- The waterfall model (software life cycle)
  - Plan-driven model. This takes fundamental process activities of specification, development, validation and evolution and represent them as process phases such as ***requirements specification , software design, implementation and testing and so on***
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Reuse-oriented software engineering
  - The system is assembled from existing components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model




Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

- The main **drawback** of the waterfall model is the ***difficulty of accommodating change after the process is underway.*** In principle, ***a phase has to be complete before moving onto the next phase***.

Water fall model phases (software life cycle..)

Waterfall model

1. *Requirements analysis and definition* The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

2. *System and software design* The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

3. *Implementation and unit testing* During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

4. *Integration and system testing* The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

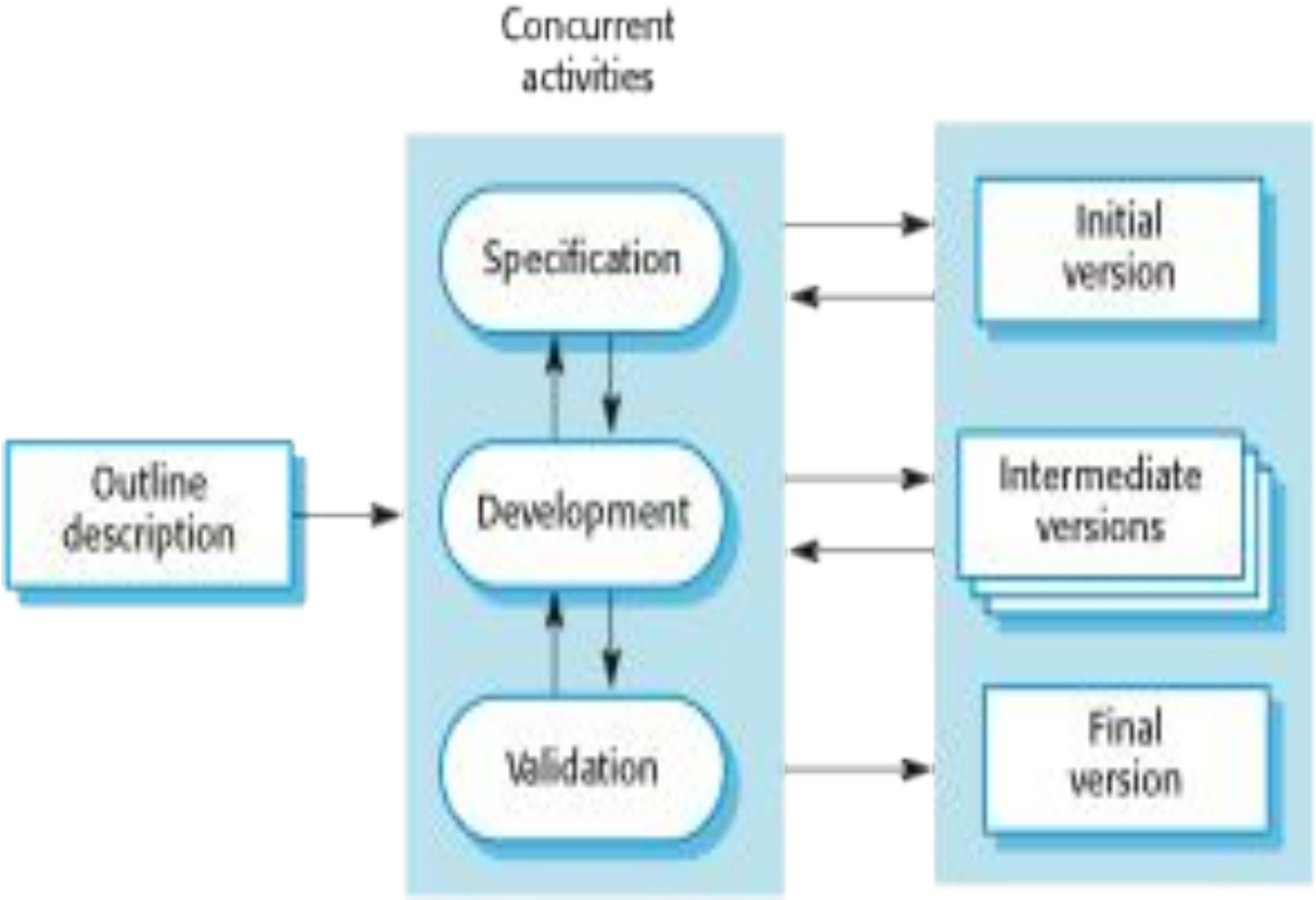
5. *Operation and maintenance* Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

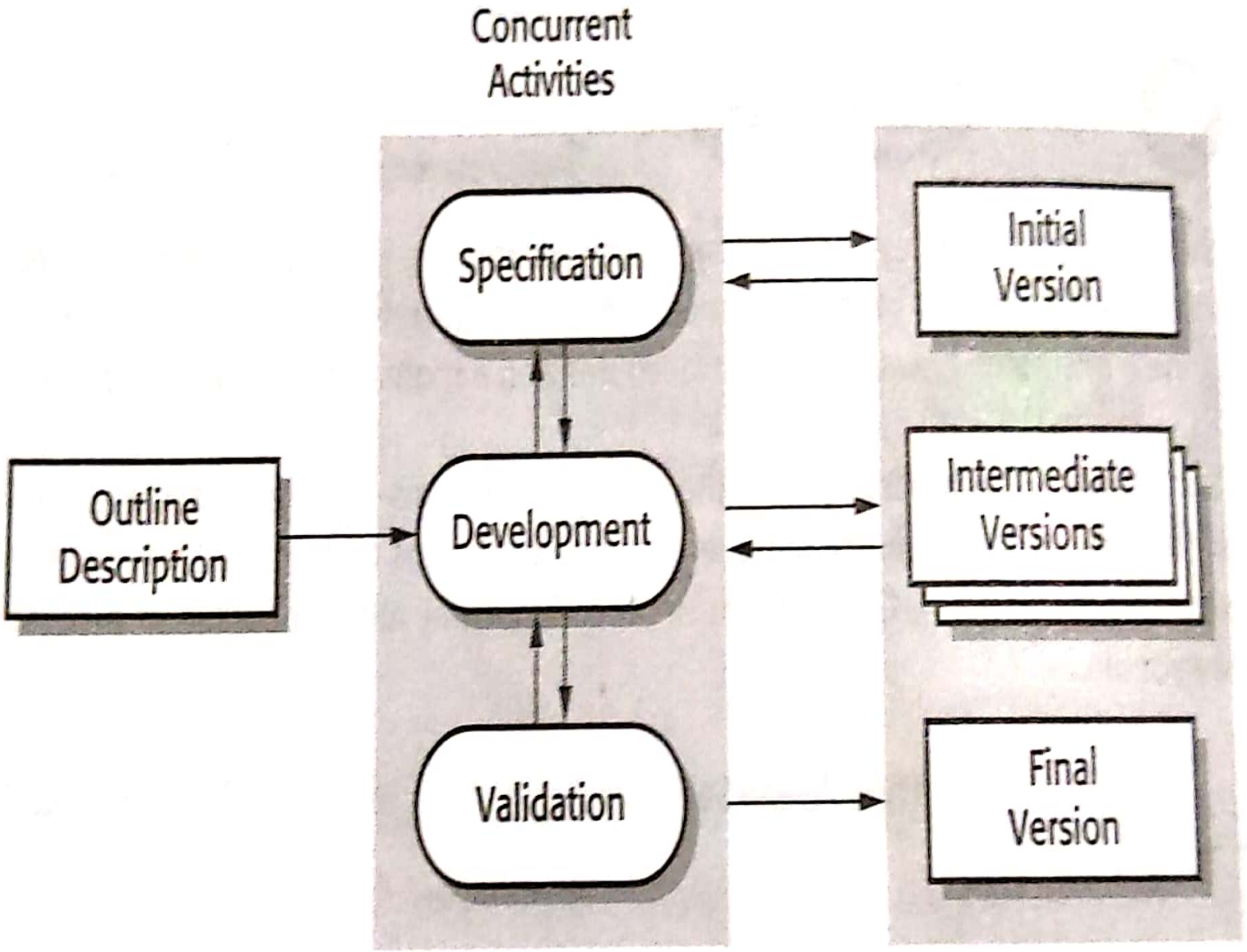
Waterfall model problems

- ***Inflexible partitioning*** of the project into ***distinct stages*** makes it ***difficult*** to respond to ***changing customer requirements.***
  - Therefore, this model is only appropriate when the requirements are **well-understood** and **changes will be fairly limited during the design process.**
  - Few business systems have **stable requirements.**

- The waterfall model is mostly used for ***large systems engineering projects where a system is developed at several sites.***
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental development




Concurrent Activities
Specification
Development
Validation
Outline Description
Initial Version
Intermediate Versions
Final Version

Incremental development benefits

- The ***cost*** of accommodating ***changing customer requirements*** is ***reduced***.
  - The ***amount*** of ***analysis and documentation*** that has to be ***redone*** is much ***less*** than is required with the waterfall model.

- It is easier to get ***customer feedback*** on the ***development work*** that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.

- More ***rapid delivery*** and ***deployment of useful software*** to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- The process is not visible.
  - Managers need *regular deliverables* to measure *progress*. If systems are developed quickly, it is *not cost-effective* to produce documents that reflect every version of the system.
- ***System structure tends to degrade as new increments are added***
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented software engineering

- **Based on systematic reuse where *systems are integrated from existing components* or *COTS (Commercial-off-the-shelf) systems*.**

- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.

- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 16.

Reuse-oriented software engineering

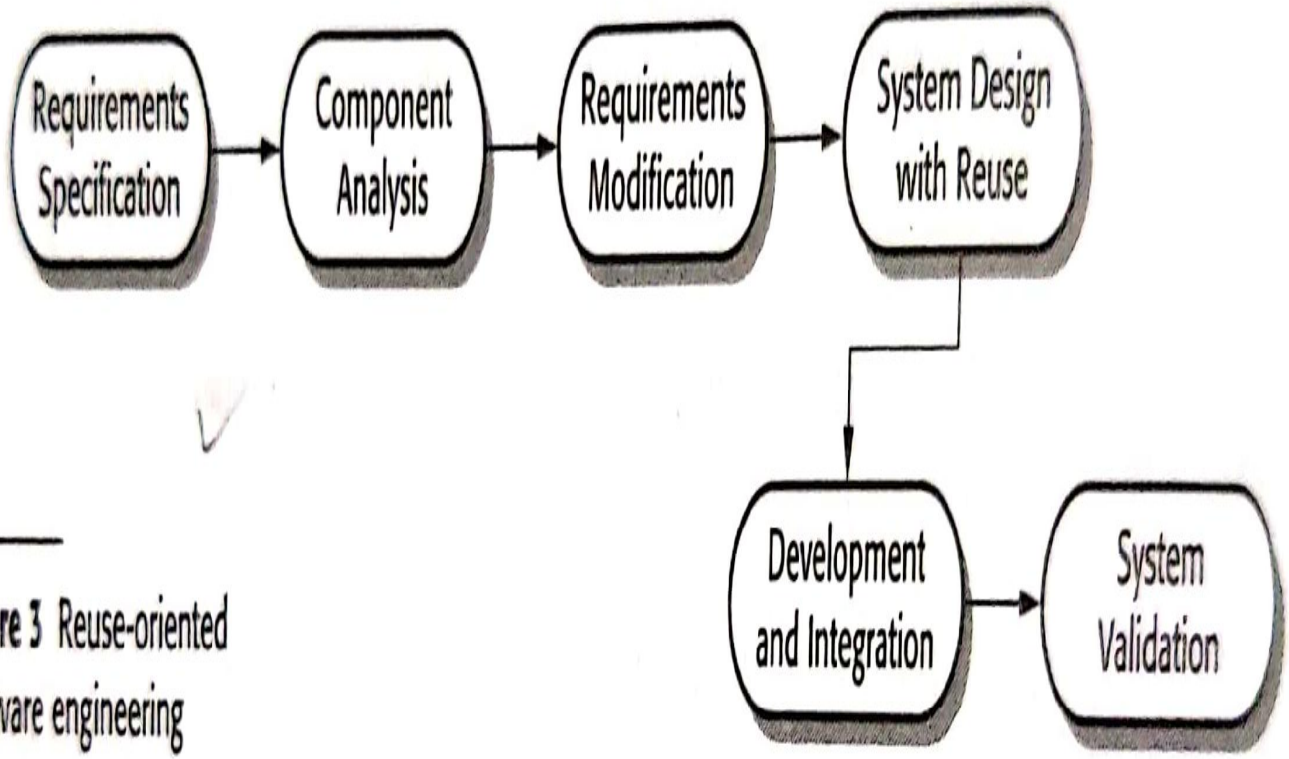


Figure 3 Reuse-oriented software engineering

Types of software component used in reuse-oriented process

- ***Web services*** that are developed according to service standards and which are available for remote invocation.
- ***Collections of objects*** that are developed as a ***package*** to be integrated with a component framework such as ***.NET or J2EE.***
- ***Stand-alone software systems*** (COTS) that are configured for use in a particular environment.

Process activities

- **Real software processes** are ***inter-leaved sequences*** of ***technical, collaborative*** and ***managerial activities*** with the overall goal of *specifying*, *designing*, *implementing* and *testing* a software system.

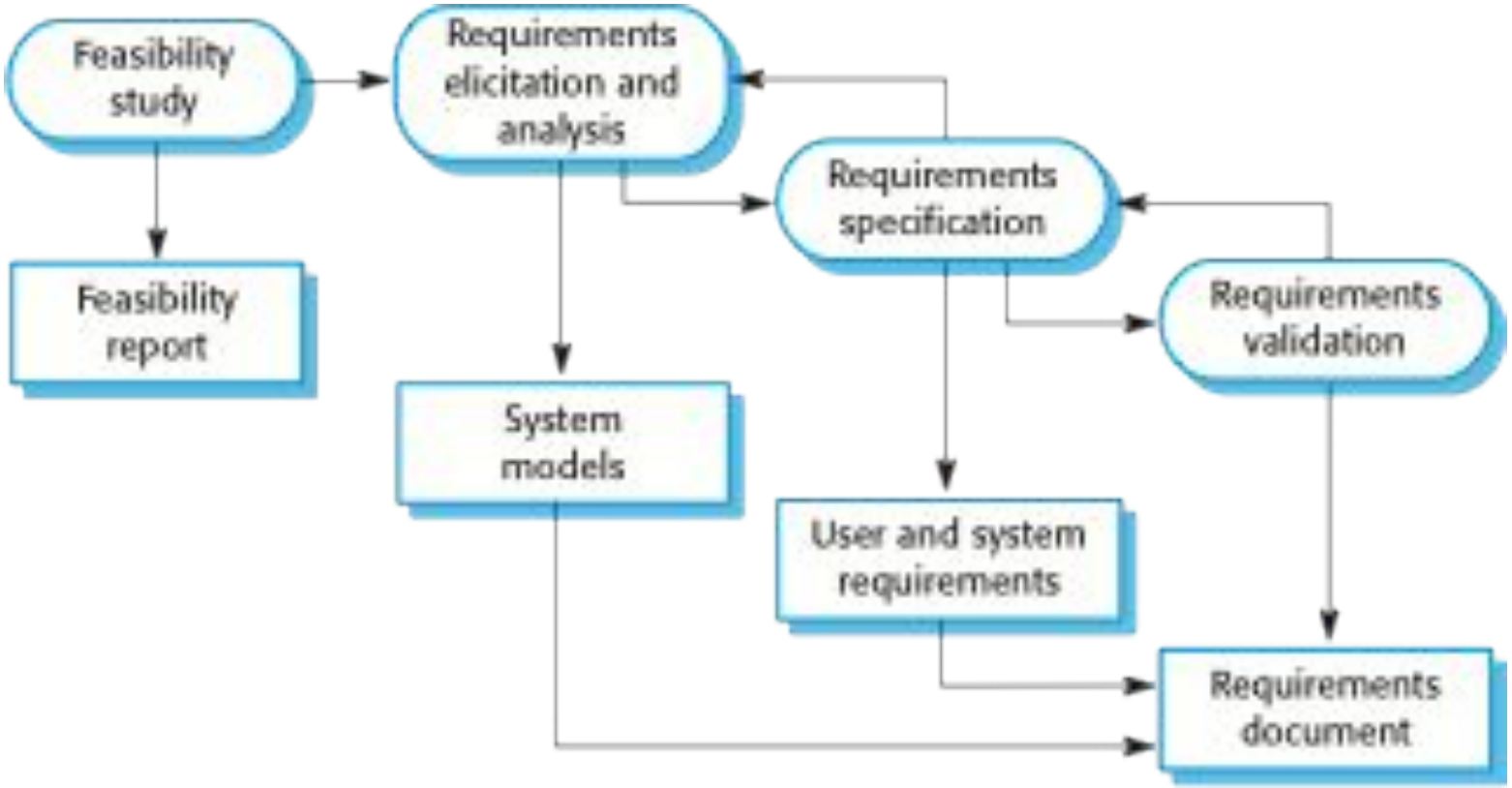
- The four basic process activities of **s**pecification, **d**evelopment, **v**alidation and **e**volution are organized differently in different development processes.

  - In the **waterfall model,** they are organized in ***sequence,*** whereas in **incremental development** they are ***inter-leaved.***

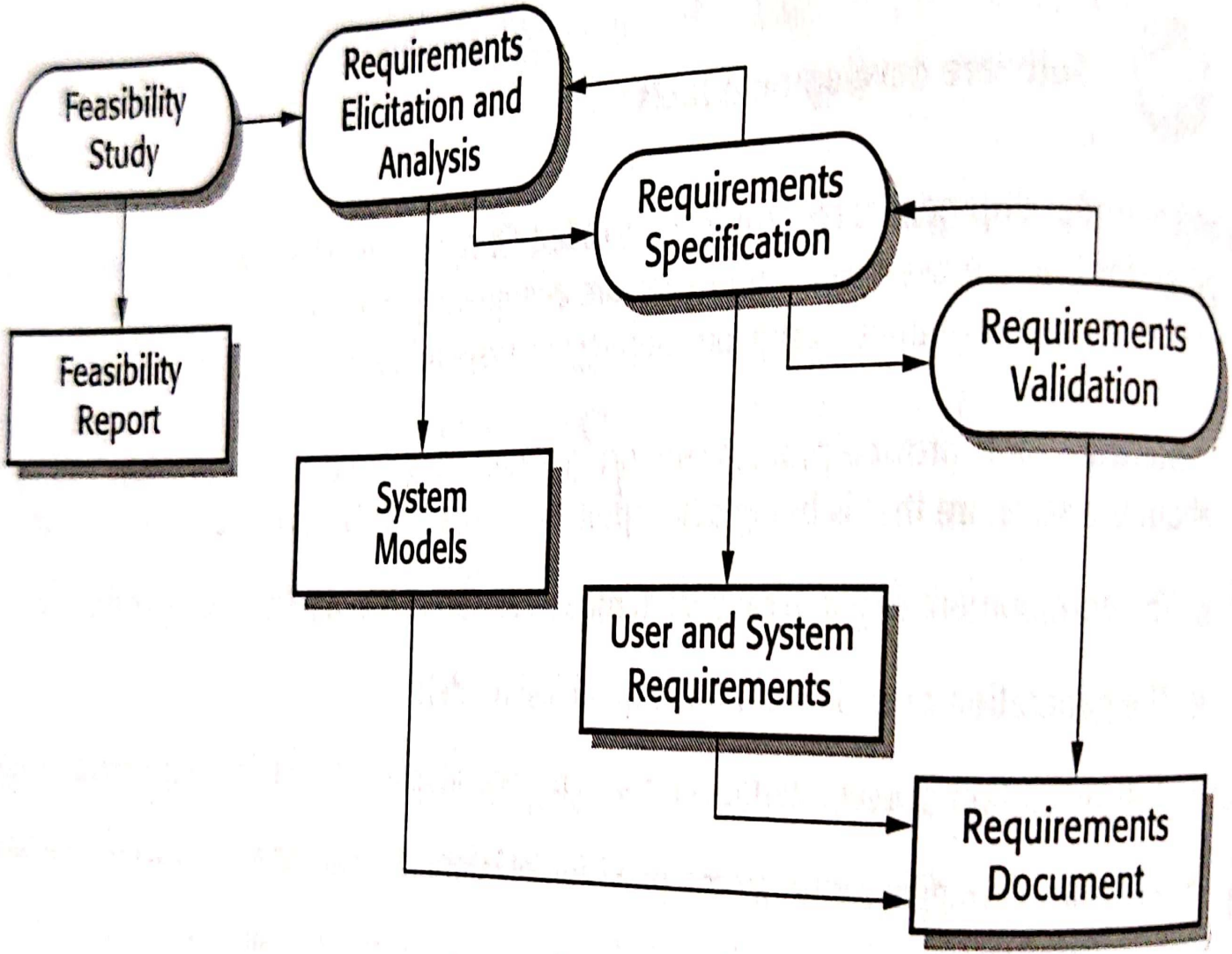
Software specification

- The process of establishing ***what services are required*** and the ***constraints on the system's operation*** and ***development.***
- **Requirements engineering process**
  - Feasibility study
    - Is it **technically and financially** feasible to build the system?
  - Requirements elicitation and analysis
    - What do the **system stakeholders** require or expect from the system?
  - Requirements specification
    - **Defining the requirements in detail**
    - **User requirements** - abstract statements of the system requirements for the customer and end-user of the system.
    - **System requirements** – more detailed description of the functionality to be provided.
  - Requirements validation
    - Checking the validity of the requirements
    - **Checks the requirements for realism, consistency and completeness.**

The requirements engineering process



The requirements engineering process



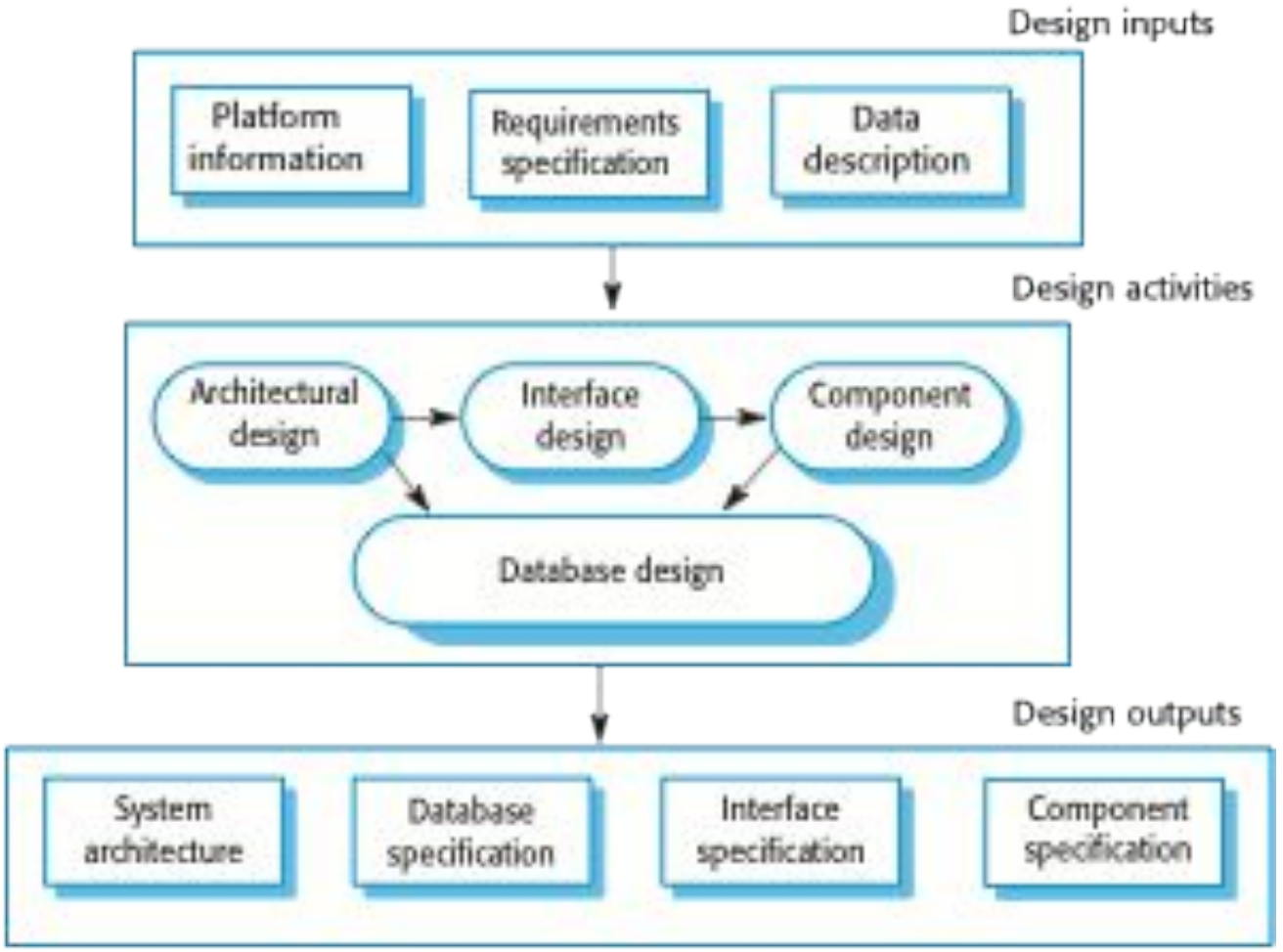
Software design and implementation

- The process of converting the ***system specification*** into an ***executable system***.

- **Software design**
  - Design a ***software structure*** that realises the ***specification***;
- **Implementation**
  - Translate this ***structure*** into an ***executable program***;

- The activities of design and implementation are closely related and may be ***inter-leaved***.

A general model of the design process



A general model of the design process

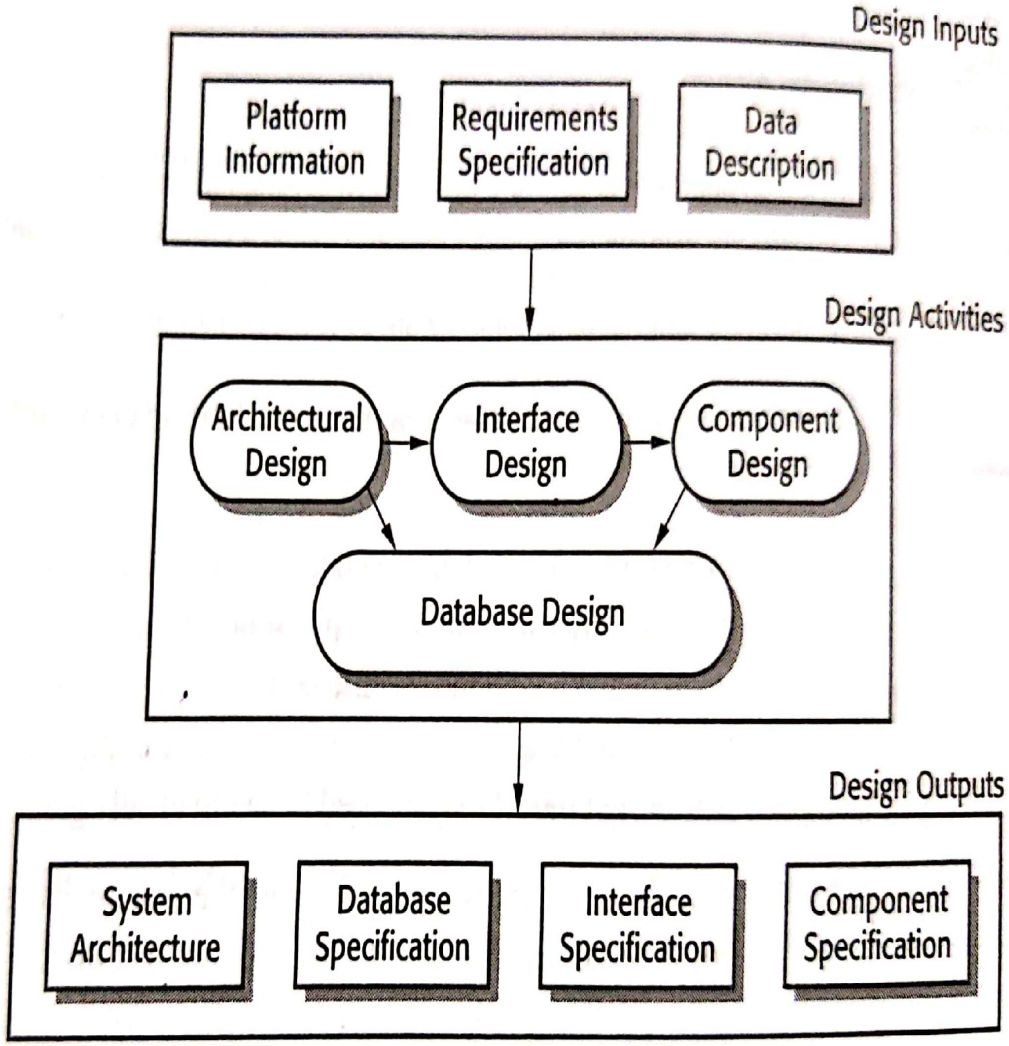


Figure 5 A general model of the design process

Design activities

- ***Architectural design*****,** where you identify the **overall structure of the system**, the **principal components** (sometimes called sub-systems or modules), their **relationships** and how they are **distributed**.

- ***Interface design,*** where you define the ***interfaces*** between ***system components.***

- ***Component design,*** where you take **each system component** and **design how it will operate**.

- ***Database design*****,** where you design the **system data structures** and **how these are to be represented in a database.**

Software validation

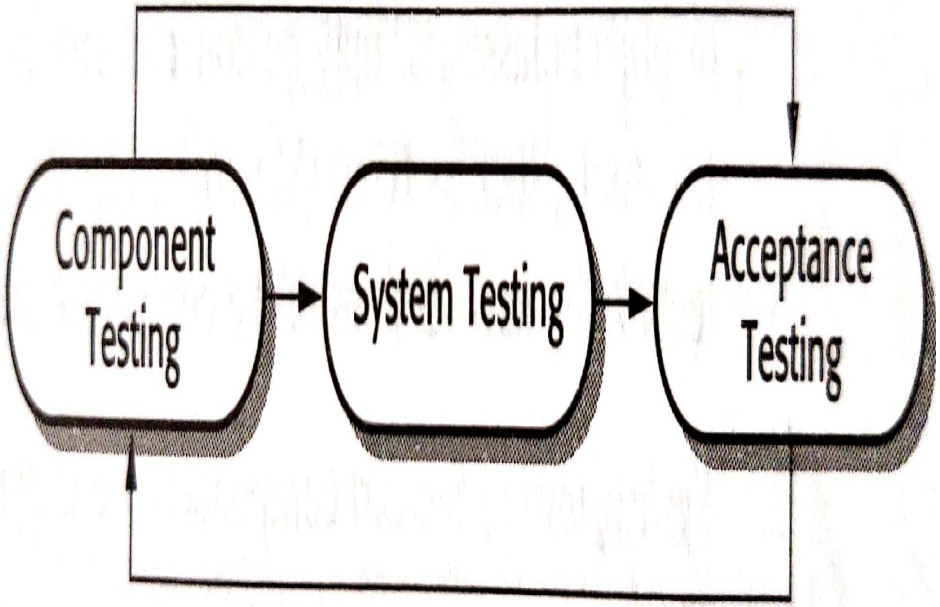
- **Verification and validation (V & V)** is intended to show that a ***system conforms to its specification*** and ***meets the requirements of the system customer.***

- Involves ***checking*** and ***review processes*** and ***system testing***.

- **System testing** involves ***executing the system*** with ***test cases*** that are derived from the **specification of the real data** to be processed by the system.

- **Testing** is the most commonly used **V & V activity**.

Stages of testing



Stages of testing

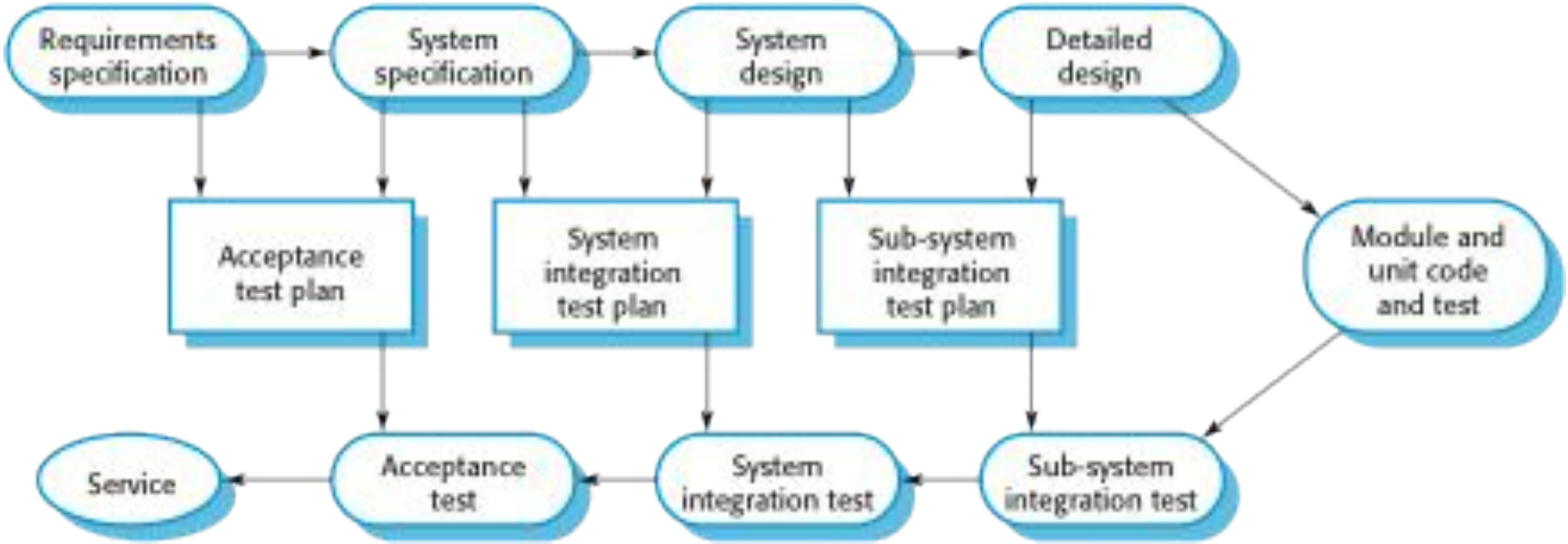
Testing stages

- **Development or component testing**
  - ***Individual components*** are tested ***independently;***
  - Components may be *functions* or ***objects*** or coherent groupings of these entities.

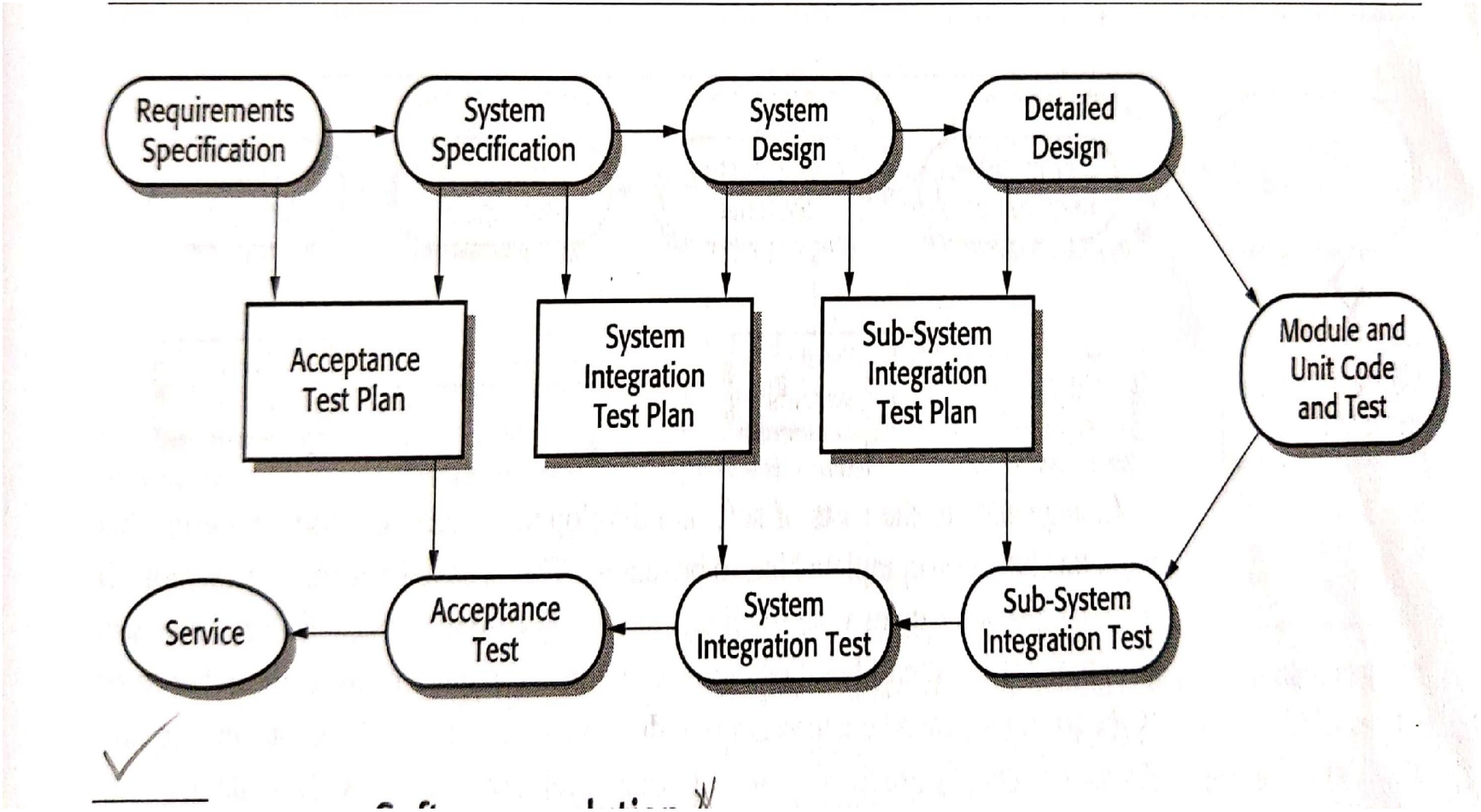
- **System testing**
  - Testing of the ***system as a whole***. Testing of emergent properties is particularly important.

- **Acceptance testing**
  - ***Testing with customer data*** to check that the ***system meets the customer's needs.***

Testing phases in a plan-driven software process



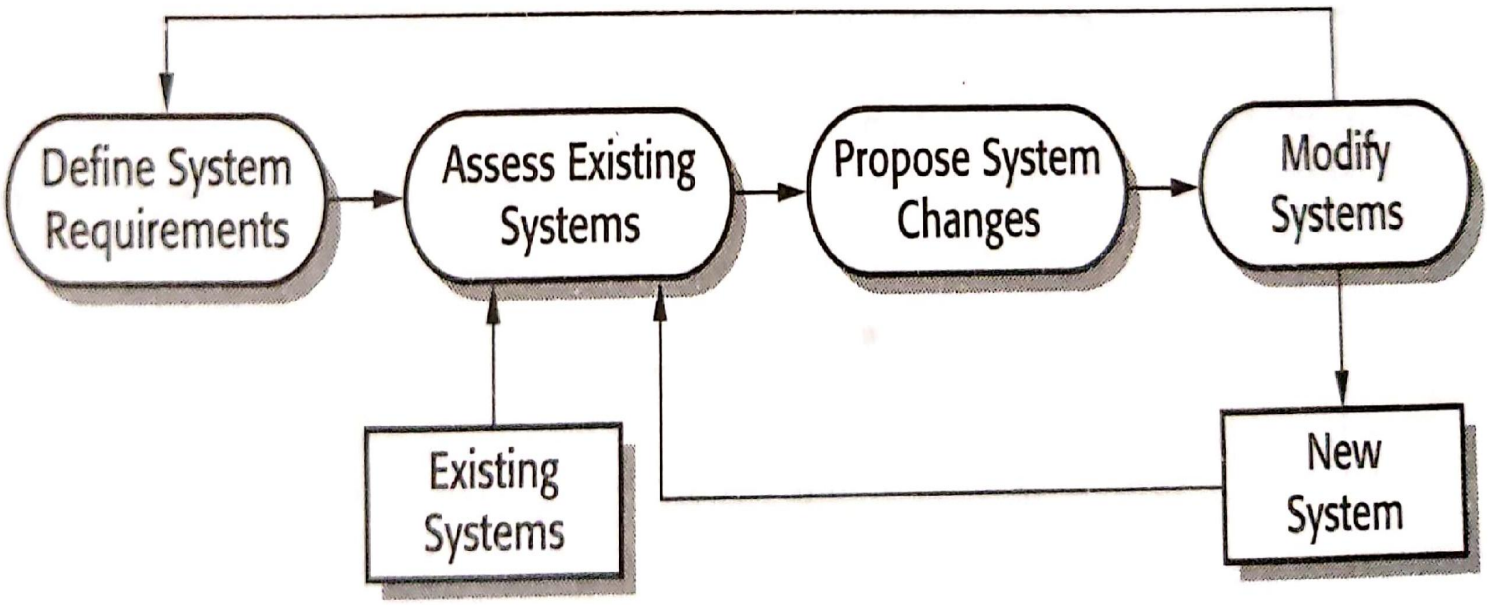
Testing phases in a plan-driven software process



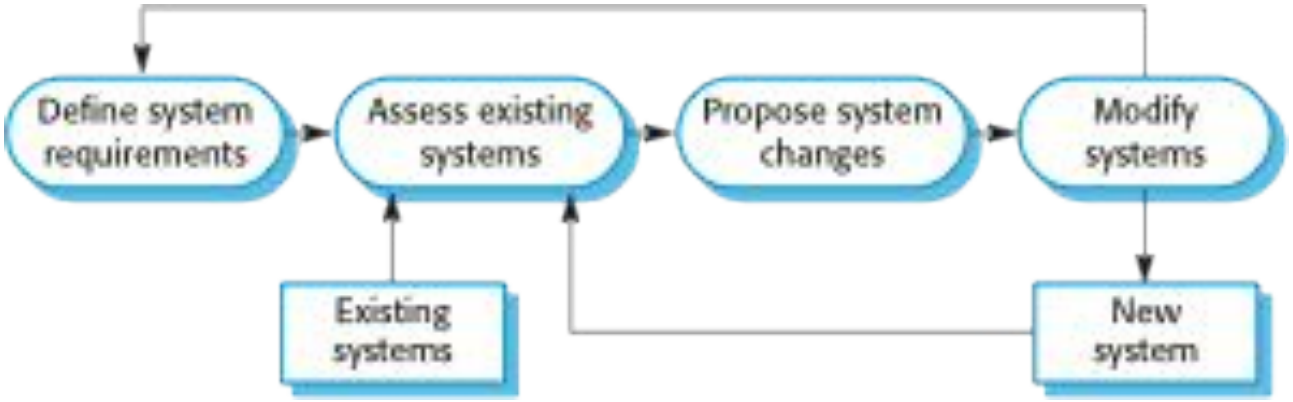
Software evolution

- Software is i**nherently flexible** and **can change.**
- As requirements change through ***changing business circumstances,*** the ***software that supports the business*** must also ***evolve*** and ***change.***
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



System evolution



Coping with change

- Change is **inevitable** in all **large software projects**.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes

- Change leads to **rework** so the **costs** of change include both rework (e.g. ***re-analysing requirements***) as well as the ***costs of implementing new functionality***

Reducing the costs of rework

Two Approaches

- **Change avoidance,** where the software process includes activities that can ***anticipate possible changes before significant rework is required***.
  - For example, a ***prototype system*** may be developed to show some key features of the system to customers.

- **Change tolerance**, where the ***process is designed*** so that ***changes can be accommodated*** at ***relatively low cost***.
  - This normally involves some form of ***incremental development.*** Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment ***(a small part of the system) may have be altered to incorporate the change***.

Software prototyping

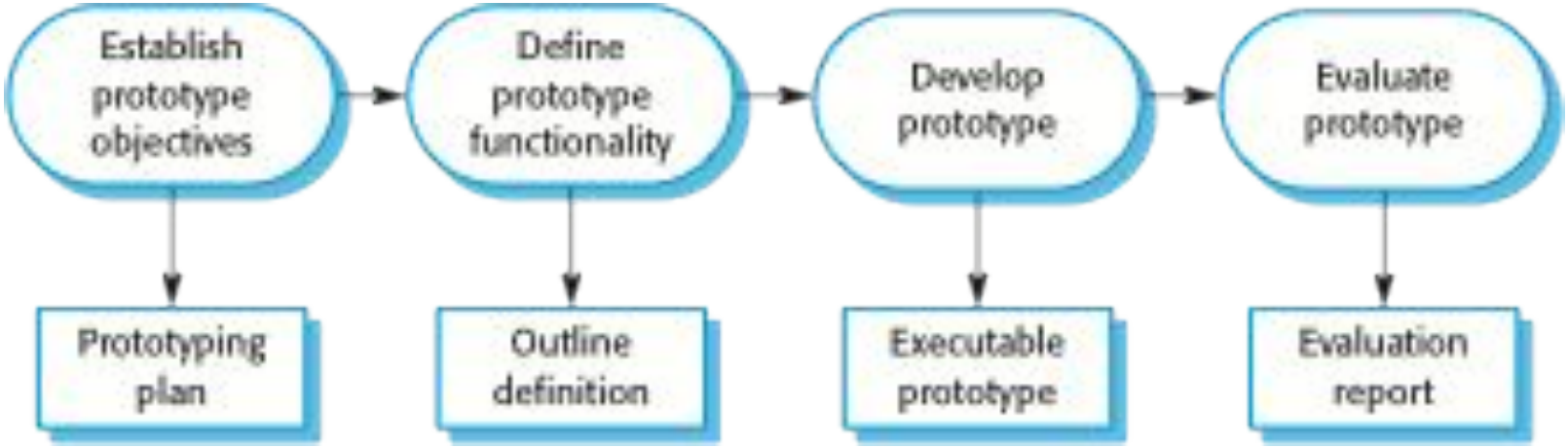
- A prototype is an ***initial version of a system*** used to demonstrate ***concepts*** and ***try out design options.***

- 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.

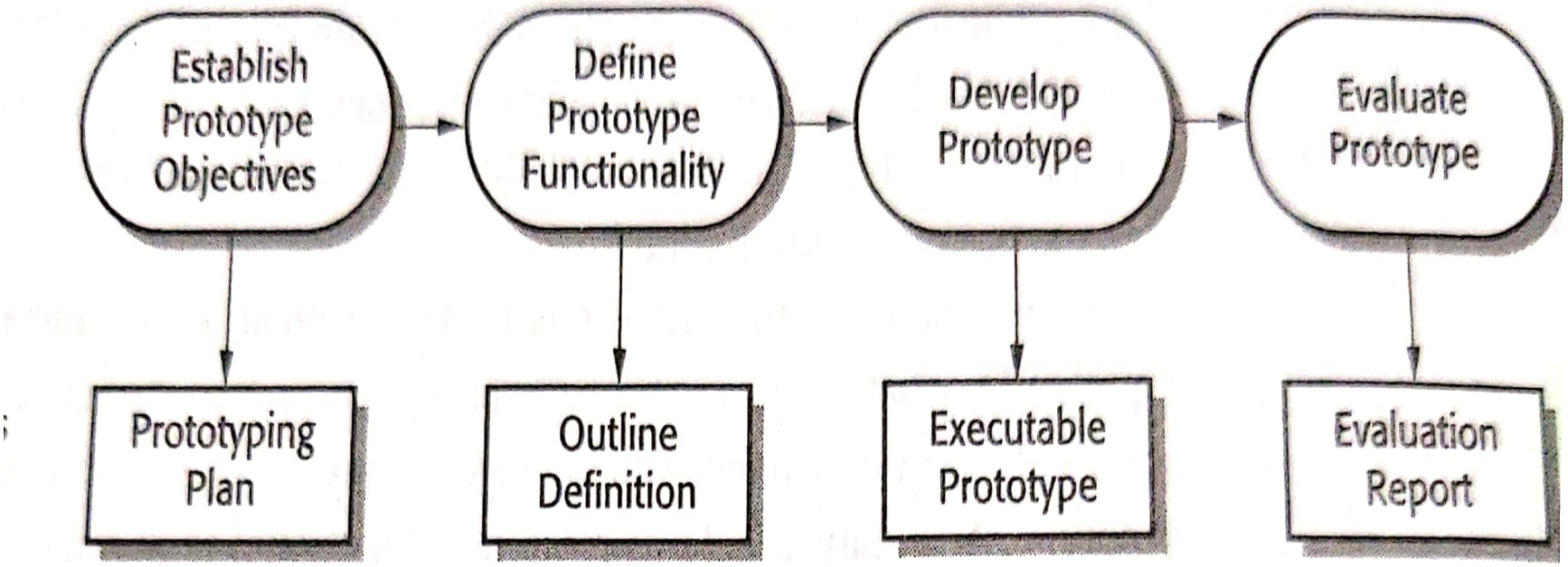
Benefits of prototyping

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

The process of prototype development



The process of prototype development



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***

Throw-away prototypes

- Prototypes should be ***discarded after development*** as they are ***not a good basis*** for ***a production system:***
  - It may be **impossible** to tune the system to meet ***non-functional requirements***;
  - Prototypes are normally ***undocumented***;
  - The prototype ***structure*** is usually through rapid change; ***degraded***
  - The prototype probably ***will not meet*** **normal** ***organisational quality standards.***

Incremental delivery

- ***Rather than deliver the system as a single delivery***, the ***development and delivery is broken down into increments*** with ***each increment*** delivering ***part of the required functionality.***

- User requirements are **prioritised** and the **highest priority requirements** are included in ***early*** *increments*.

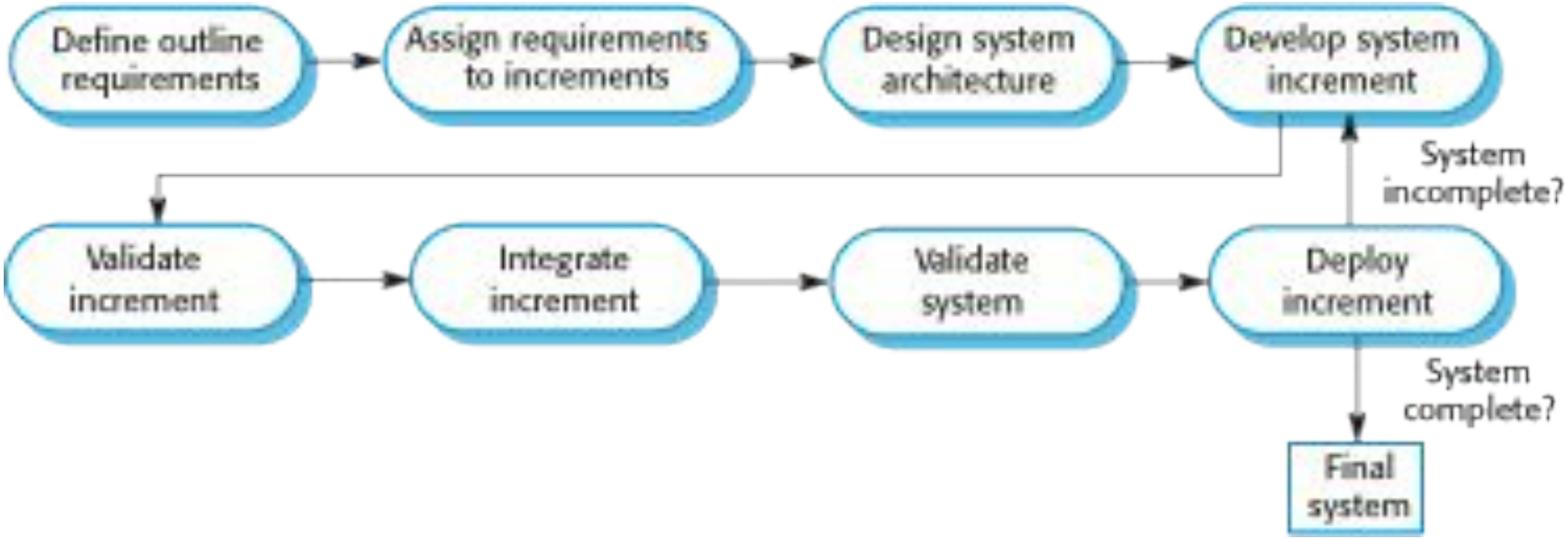
- Once the development of an increment is started, the requirements are ***frozen*** though requirements for later increments can continue to evolve.

Incremental development and delivery

- Incremental development
  - ***Develop the system in increments*** and ***evaluate each increment*** before proceeding to the development of the next increment;
  - Normal approach used in **agile methods**;
  - **Evaluation** done by **user/customer proxy.**

- Incremental delivery
  - ***Deploy*** an increment for ***use by end-users;***
  - More ***realistic evaluation*** about ***practical use of software;***
  - Difficult to implement for replacement systems as increments have less functionality than the system being replaced.

Incremental delivery



Incremental delivery

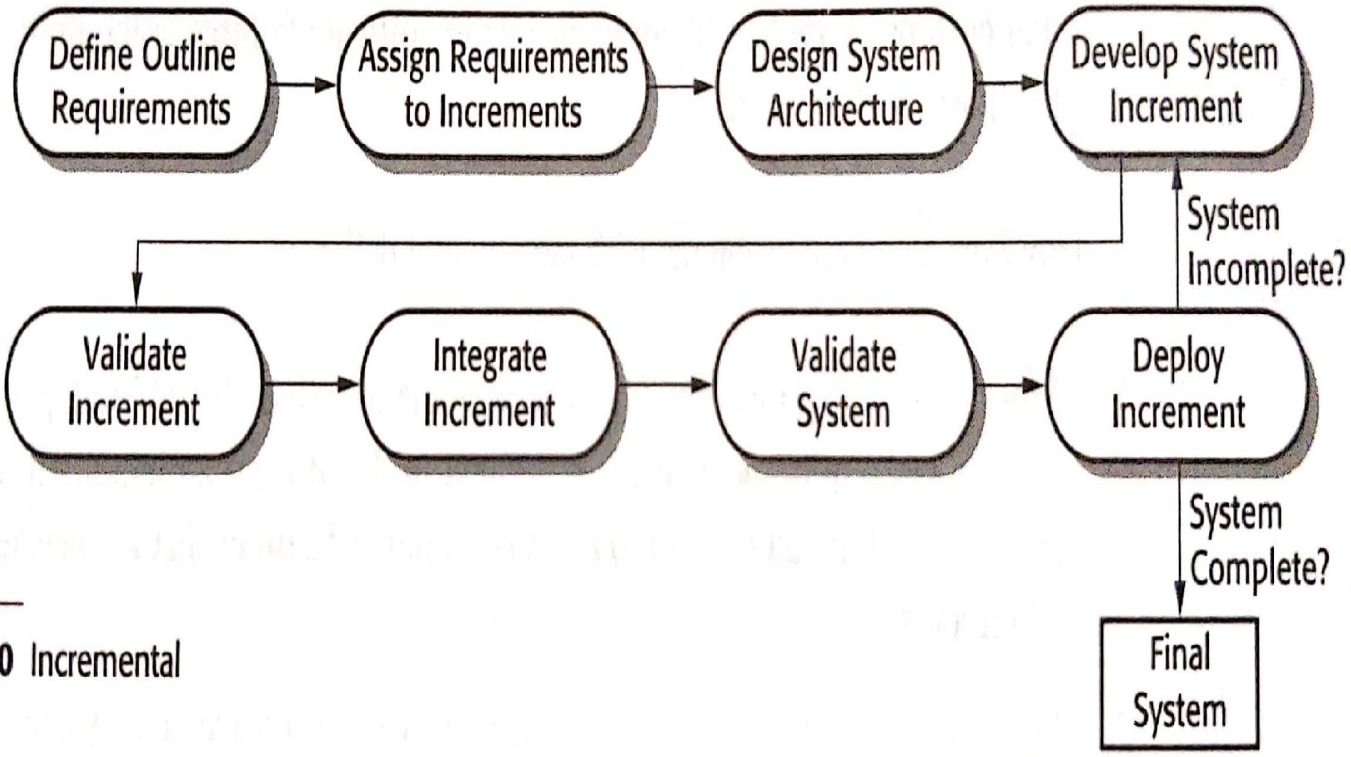


Figure 10 Incremental delivery

Incremental delivery advantages

- ***Customer value*** can be delivered with each increment so system functionality is available earlier.
- ***Early increments*** act as a ***prototype*** to ***help elicit requirements for later increments***.
- ***Lower risk*** of ***overall project failure***.
- The ***highest priority system services*** tend to receive the ***most testing.***

Incremental delivery problems

- Most systems require a ***set of basic facilities that are used by different parts of the system***.
  - As requirements are not defined in detail until an increment is to be implemented, ***it can be hard to identify common facilities that are needed by all increments.***

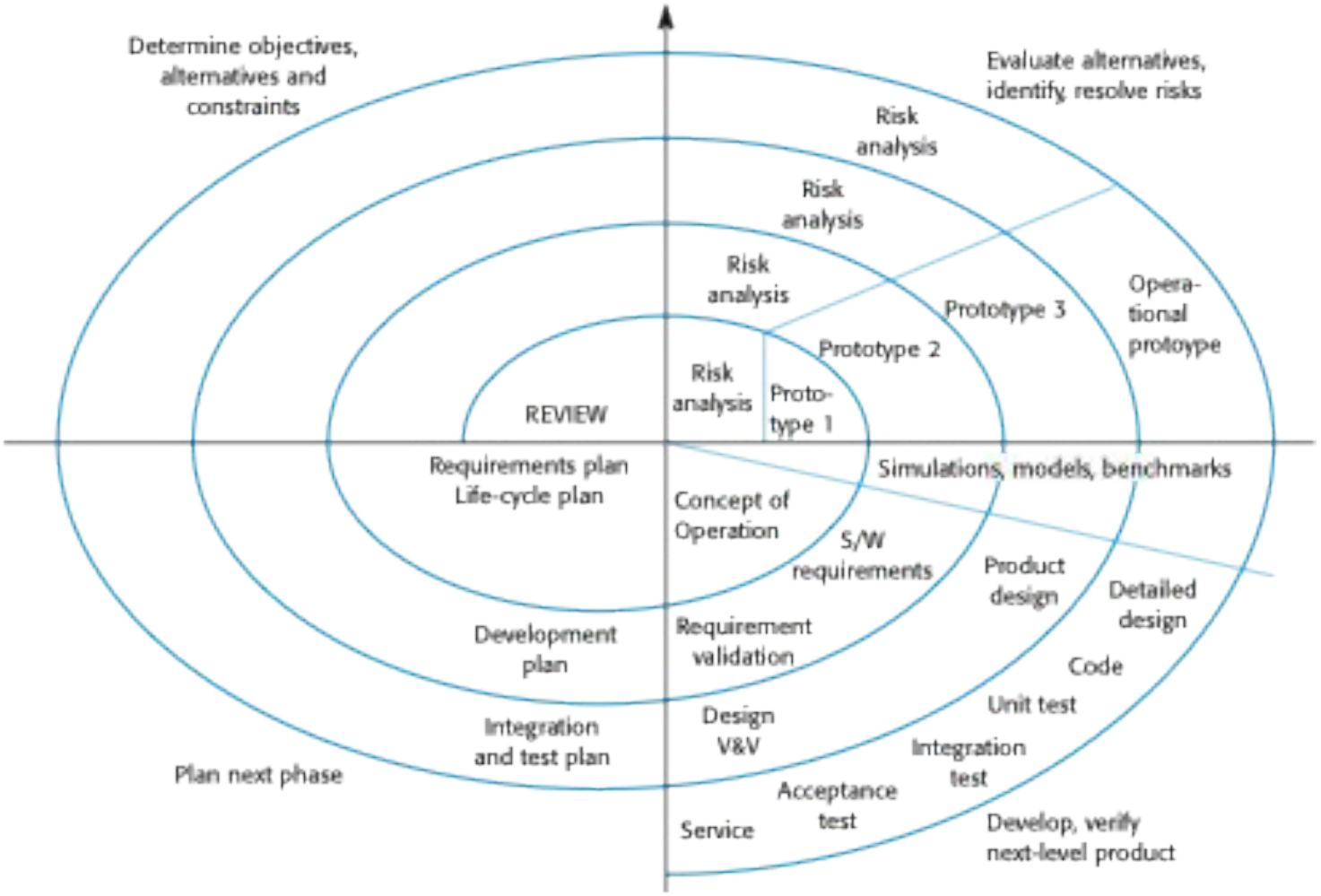
- The essence of iterative processes is that the ***specification*** is developed ***in conjunction*** with the ***software.***
  - However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

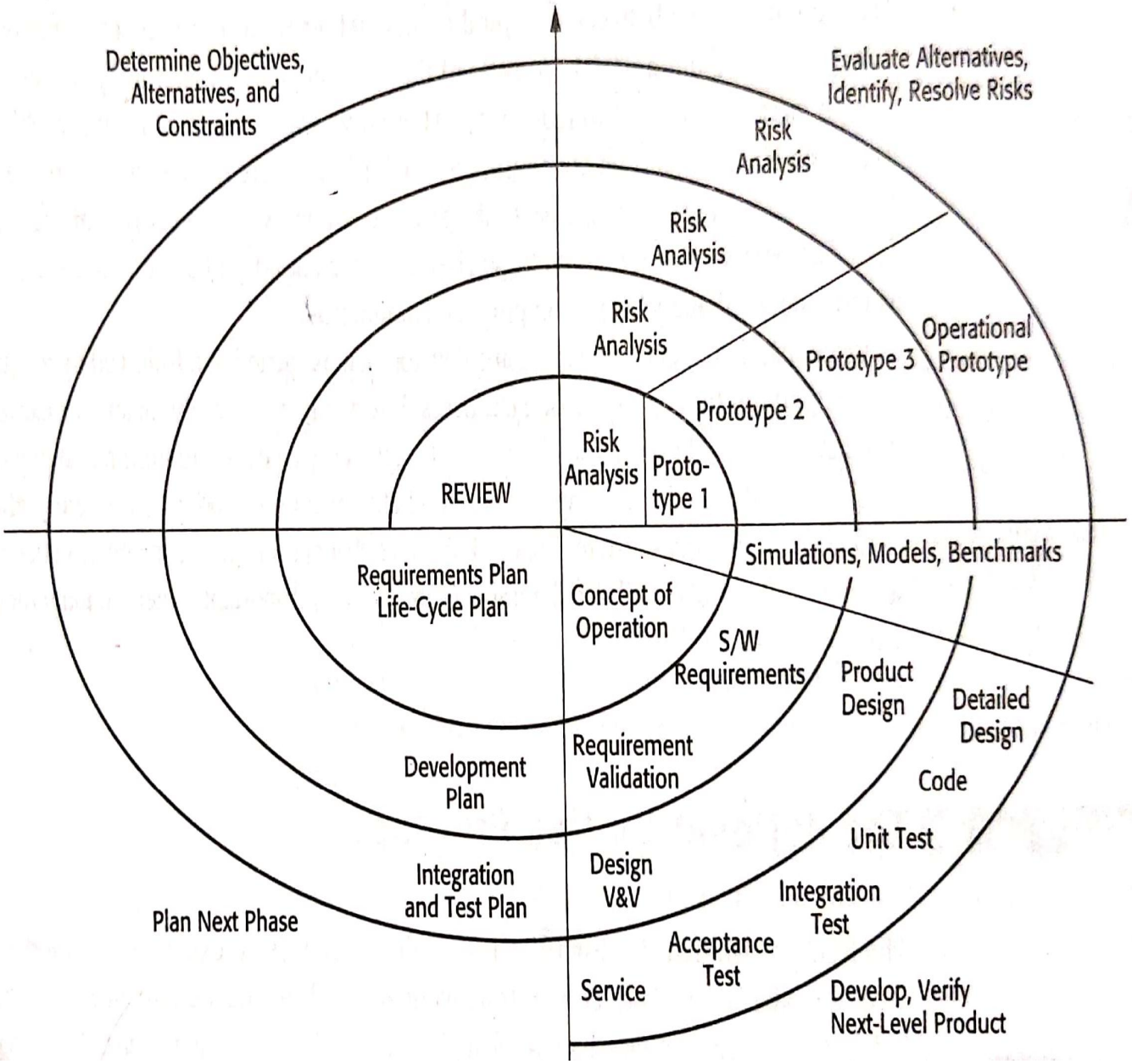
Boehm's spiral model

- **Process** is represented as a **spiral** rather than as a sequence of activities with backtracking.
- **Each loop** in the spiral represents a **phase in the process.**
- **No fixed phases** such as specification or design - ***loops in the spiral are chosen depending on what is required.***
- **Risks** are **explicitly assessed** and **resolved** throughout the process.

- Thus, the **innermost loop** might be concerned with **system feasibility**, the **next loop** with **requirements definition**, the next loop with system **design,** and so on.
- The spiral model **combines change avoidance** with **change tolerance.**
- It assumes that **changes** are a result of **project risks** and includes **explicit risk management activities** to **reduce these risks.**
- **Each loop** in the spiral is split into **four sectors**

Boehm's spiral model of the software process




Determine Objectives, Alternatives, and Constraints
Evaluate Alternatives, Identify, Resolve Risks
Risk Analysis
Risk Analysis
Risk Analysis
Risk Analysis
Operational Prototype
Prototype 3
Prototype 2
Proto-type 1
REVIEW
Simulations, Models, Benchmarks
Requirements Plan Life-Cycle Plan
Concept of Operation
S/W Requirements
Product Design
Detailed Design
Code
Unit Test
Integration Test
Acceptance Test
Develop, Verify Next-Level Product
Requirement Validation
Design V&V
Integration and Test Plan
Development Plan
Plan Next Phase
Service

Spiral model sectors

- Objective setting
  - Specific objectives for the phase are identified.
- Risk assessment and reduction
  - Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
  - A development model for the system is chosen which can be any of the generic models.
- Planning
  - The project is reviewed and the next phase of the spiral is planned.

- The **main difference** between the spiral model and other software process models is its **explicit recognition of risk.**
- A cycle of the spiral begins by elaborating objectives such as performance and functionality. Alternative ways of achieving these objectives, and dealing with the constraints on each of them, are then enumerated.
- Each alternative is assessed against each objective and sources of project risk are identified.
- The next step is to resolve these risks by information-gathering activities such as more detailed analysis, prototyping, and simulation.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

Banking Application Problem Statement : Assume that you are a software development Company and you are required to develop a software for a Bank to automate all the major processes. A typical bank will have a head office and many branches spread over a city/state/Country and each branch is managed by a few employees and a branch manager. Every branch will have many customers and as per bank rules they can have many accounts(but limited to 3) and also can share an account. The customer accounts can be of different types such as savings account(s/b), recurring deposit account(rd), current account(c/a). The bank generates its revenue from lending loans of different types such as corporate loan, housing loan, vehicle loan and personal loan. The interest rates are different for each of these types and there will be one time fixed processing fees. The bank derives its corpus from customer deposits, share capital or loan taken from RBI. All customers must open an account with a minimum balance by providing details like Identity proof(Adhar Card), Address proof(Ratio Card/Electricity bill receipt) and nominee and introducer details. This step is necessary to avail all the facilities of the bank such as ATM card, Credit Card, Loan, Deposits, Demand Draft, NEFT etc. The customers after opening the account, can do various transactions such as deposit, withdraw, transfer amount etc. The bank will disburse salary to all the employees of the bank as per the RBI/Bank rules. The bank should have online portal for carrying out most of the activities and must support 200 users at any given point in time and must be available 24×7 and the data-passed over net must be secure.

Given the above problem statement, you are required to identify the functional and data requirements and prepare a draft SRS document.

UNIT-1: SOFTWARE PROCESSES

1. What is software process model? Explain the types of software process model? 6M
2. Compare agile methodology with waterfall methodology.? L4 6M
3. Differentiate between Waterfall model and Incremental development with relevant example.? L4 8M
4. Explain Reuse oriented developmental model with neat diagram? Also discuss the benefits of this model as compared to waterfall model? L2 10M
5. With a neat diagram explain waterfall model? Explain the problems involved in waterfall model.? L2 10M
6. With a neat diagram explain Incremental development model also mention the benefits and problems involved in Incremental development model.? L2 10M

7. Explain why Boehm's spiral model is an adaptable model that can support both change avoidance and change tolerance activities. In practice, this model has not been widely used. Suggest why this might be the case. ? L2 8M
8. Describe on spiral model advantages and disadvantages ?L2 5M
9. What is prototype? Explain the process of prototype development with diagram? Mention the benefits of using prototype? L2 10M
10. What are the two approaches that may be used to reduce the cost of rework?L2 5M
11. Explain why incremental development is the most effective approach for developing business software systems. Why is this model less appropriate for real-time systems engineering? L2 8M
12. Explain why systems developed as prototypes should not normally be used as production systems? L2 7M