Introduction to Software Engineering

SE-110

Software Process & Process Models

Lecture # 6, 7, 8, 9
22, 23, 25 Feb

Rubab Jaffar
rubab.jaffar@nu.edu.pk

Today's Outline

- Software process models
  - Water Fall Model
  - Evolutionary Development
  - Component base Software Engineering / Reuse-oriented development
- Process iteration
  - Incremental Model
  - Spiral Model
- Process activities

The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - 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 an abstract representation of a process. It presents a description of a process from some particular perspective.

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 Model

- ***Software Process*** : is coherent sets of activities for specifying, designing, implementing and testing software systems.
- ***A software process model*** is an abstract representation of a software process.
- *It is a description of the sequence of activities carried out in an SE project, and the relative order of these activities.* It presents a description of a process from some particular perspective.

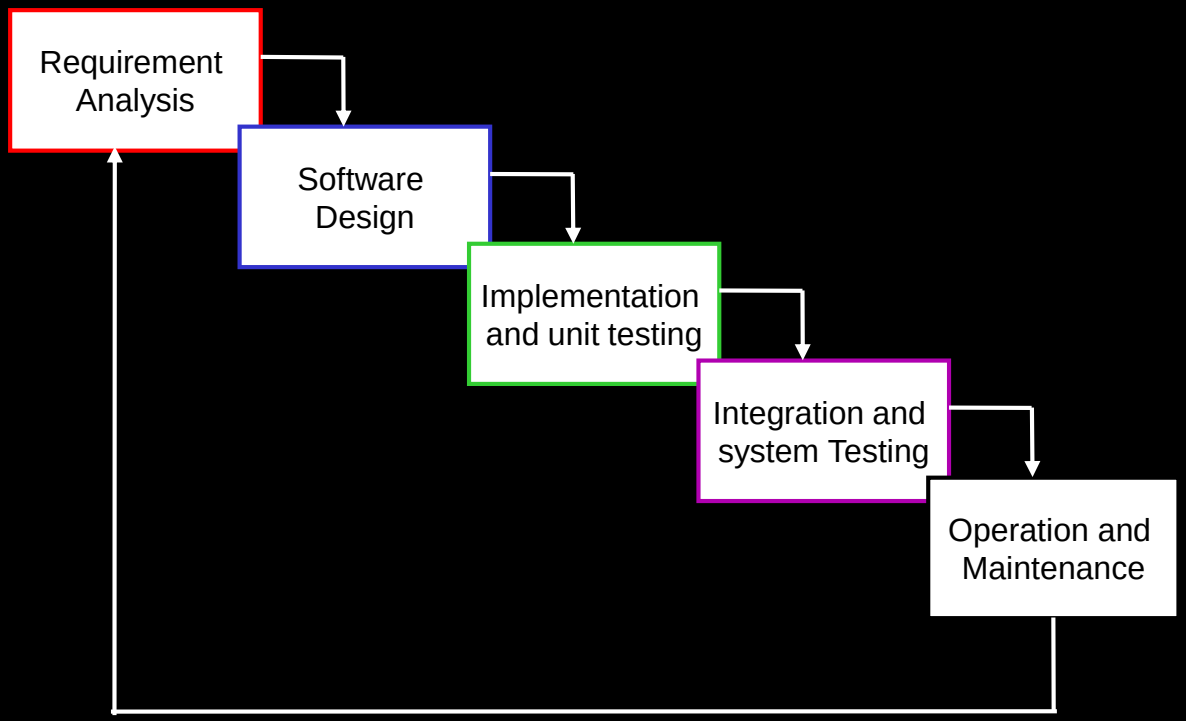
Generic software process models

- The waterfall model
  - Separate and distinct phases of specification and development. It is the oldest paradigm for SE. When requirements are well defined and reasonably stable, it leads to a linear fashion.
- Incremental development
  - Specification, development and validation are interleaved.
- Integration and configuration
  - The system is assembled from existing configurable 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.

Waterfall

Each box represents a **set of tasks** that results in the production of each or more work products.

Each new phase begins when the **work products** of the previous phase as completed, frozen and signed off.



Waterfall (when to use)

- Well suited for projects where requirements can be understood easily and technology decisions are easy
- Has been used widely
- For standard/familiar type of projects it still may be the most optimum.

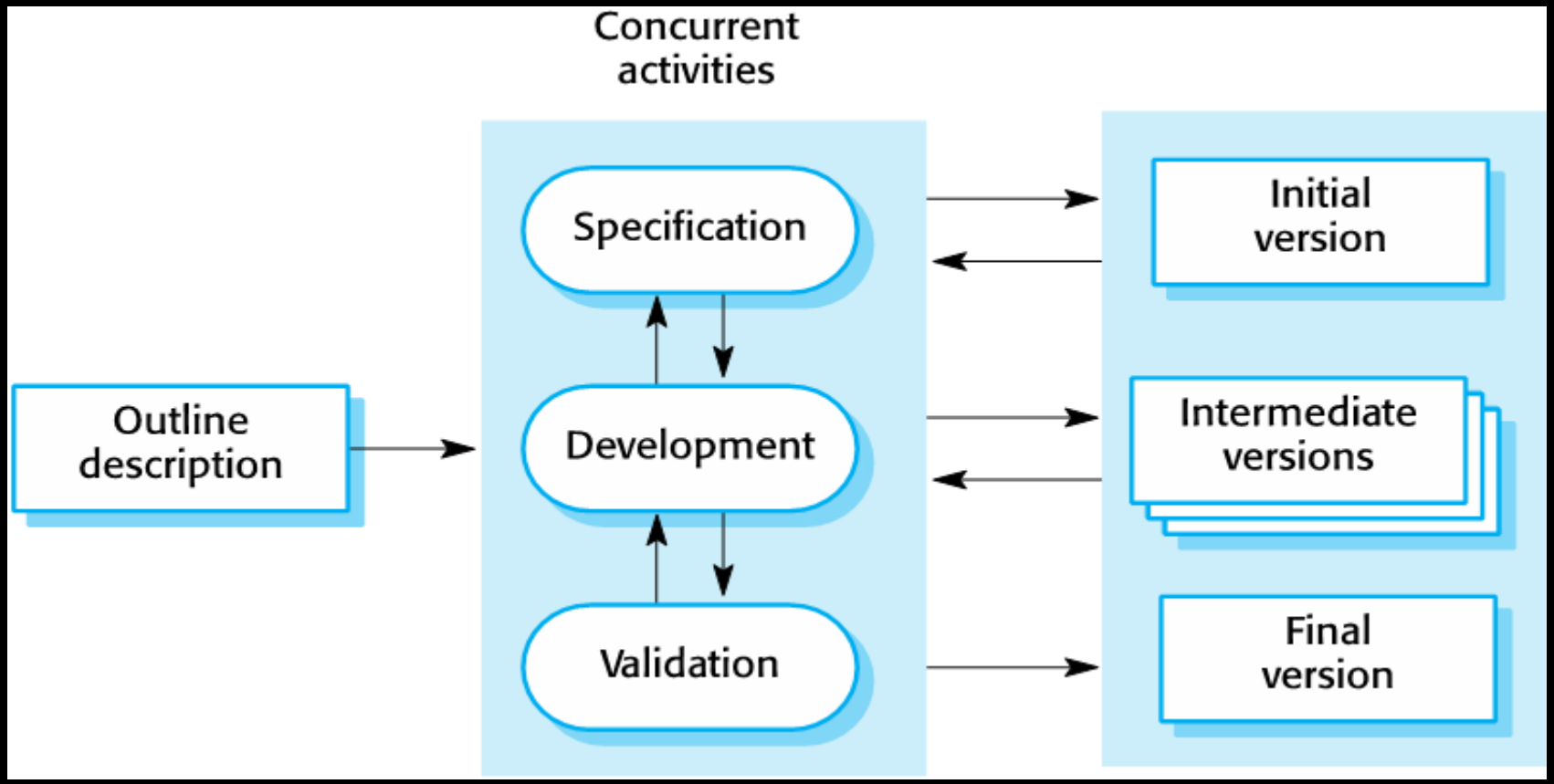
Waterfall model problems

- 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.

Disadvantages of Waterfall Model

- Unable to work where high level of uncertainty is involved
- Requirements need to be stable hence making model rigid
- Unrealistic to state all requirements at the beginning
- Does not handle concurrent events – development teams are delayed waiting for others
- Difficult and expensive to change decisions
- The user is involved only in the beginning phase of requirement gathering and than during acceptance phase

Iterative Development



Iterative 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.

Iterative 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.

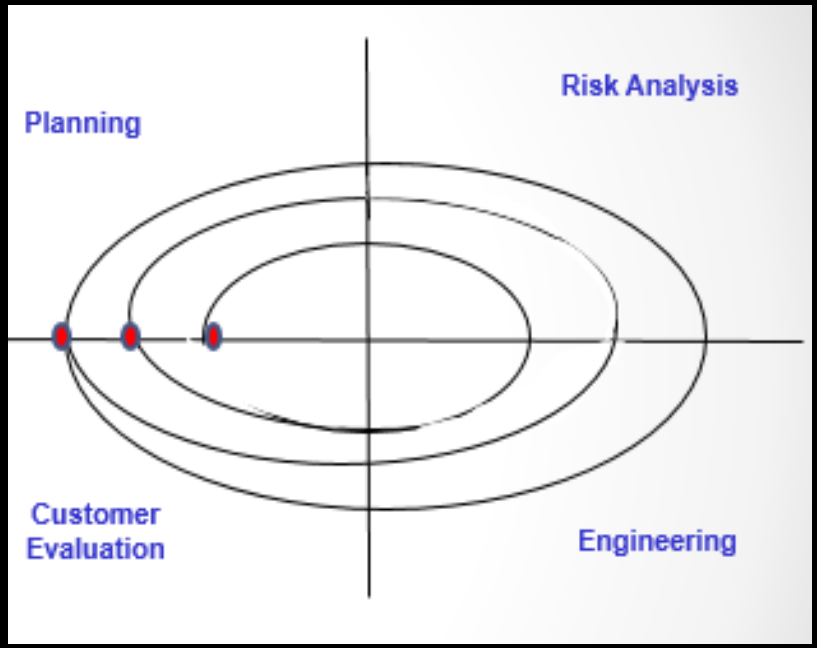
Evolutionary Models: Spiral

- Spiral is primary **Risk Driven** approach. Spiral model consist of different **cycle. In each cycle we try to address some risk elements.**

- Planning: Determines objectives, alternatives and constraints.
- Risk Analysis: Analysis of alternatives as well as an identification and/or resolution of risks.
- Engineering: Development of the next level of product
- Customer evaluation: Assessment of the results of engineering

Spiral

- With each iteration around the spiral progressively more complete versions of the software are built.
- Spiral model enables the developer, and the customer, to understand and ***react to risk*** at each evolutionary level.
- Each loop around the spiral implies that project costs and schedules may be modified.



- This creates problems in fixed-price project.

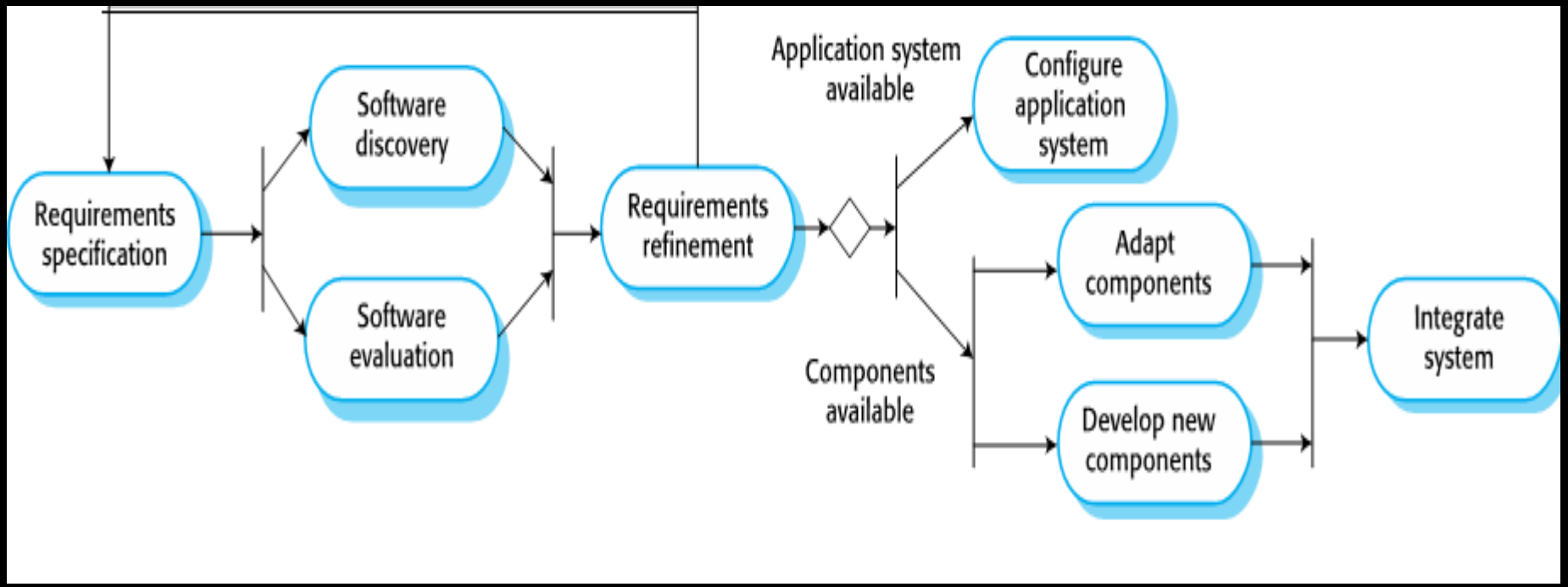
Integration and Configuration

- Based on software reuse where systems are integrated from existing components or application systems (sometimes called COTS -Commercial-off-the-shelf- systems).
- Reused elements may be configured to adapt their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 15.

Types of Reusable Software

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

Reuse-oriented Software Engineering



Process Activities

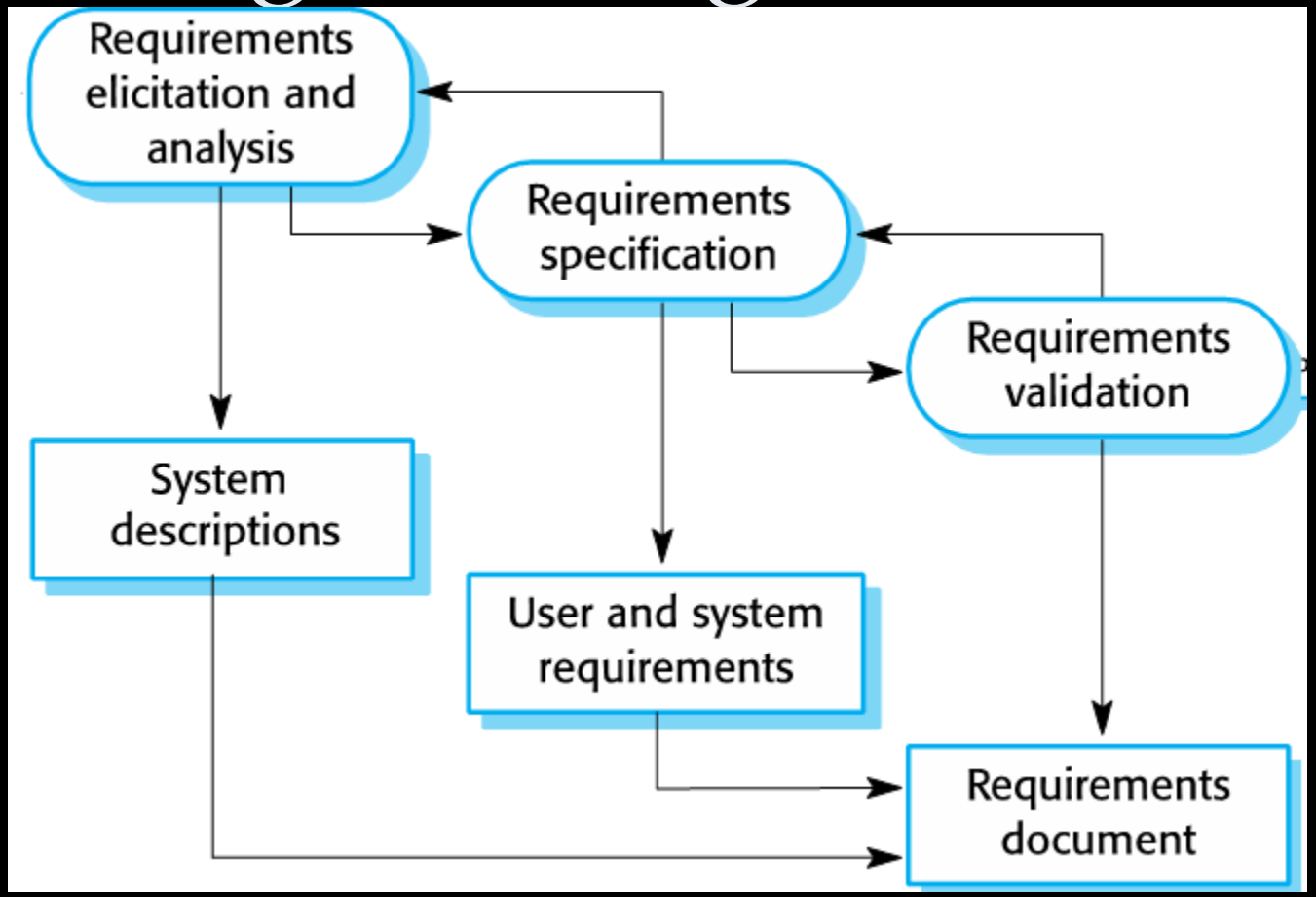
Process Activities

- 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 specification, development, validation and evolution are organized differently in different development processes.
- For example, in the waterfall model, they are organized in sequence, whereas in iterative development they are interleaved.

Software Specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

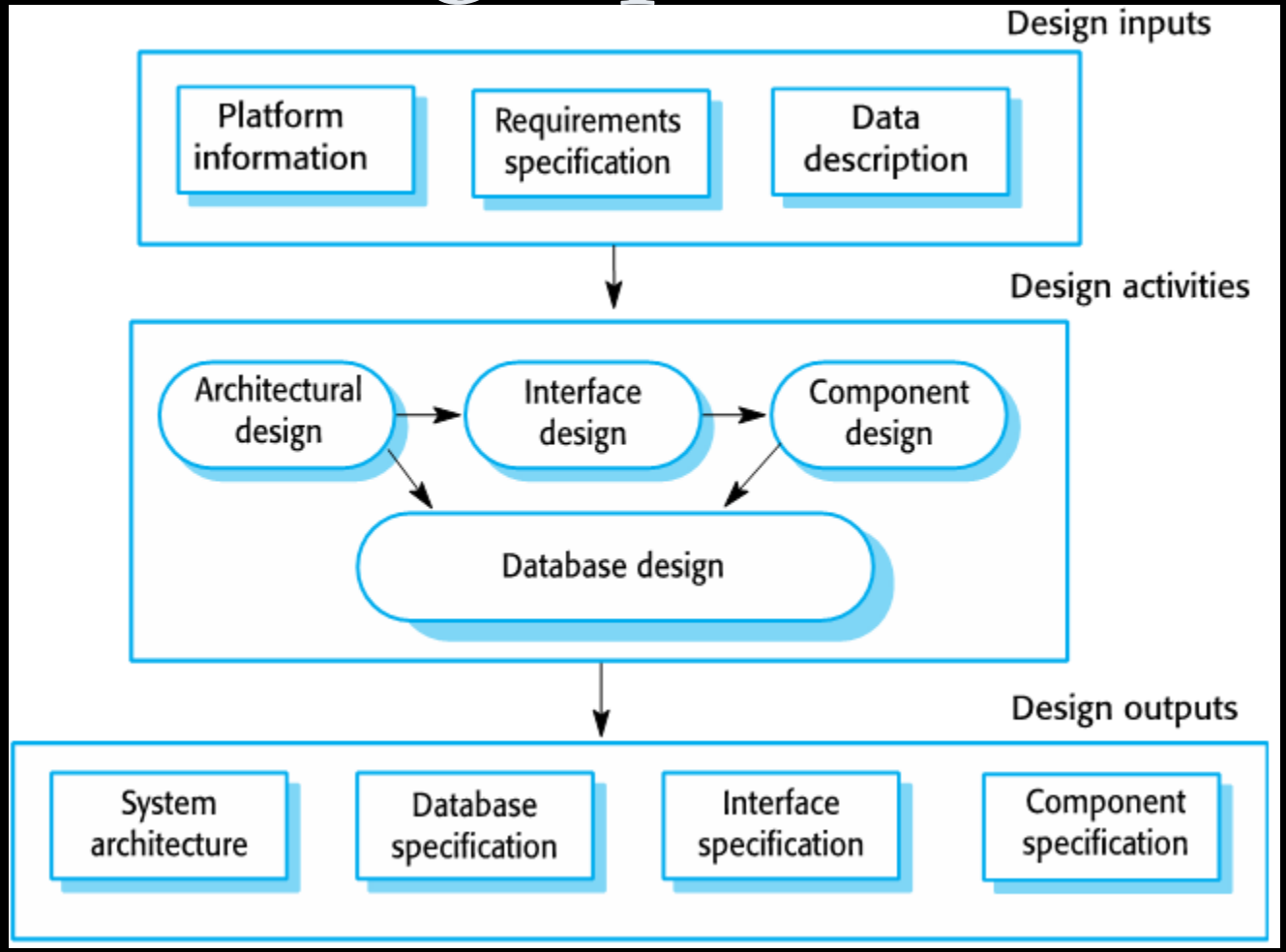
The Requirements Engineering Process



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



Design Activities

- *Architectural design,* where you identify the overall structure of the system, the principal components (subsystems or modules), their relationships and how they are distributed.
- *Database design,* where you design the system data structures and how these are to be represented in a database.
- *Interface design,* where you define the interfaces between system components.
- *Component selection and design,* where you search for reusable components. If unavailable, you design how it will operate.

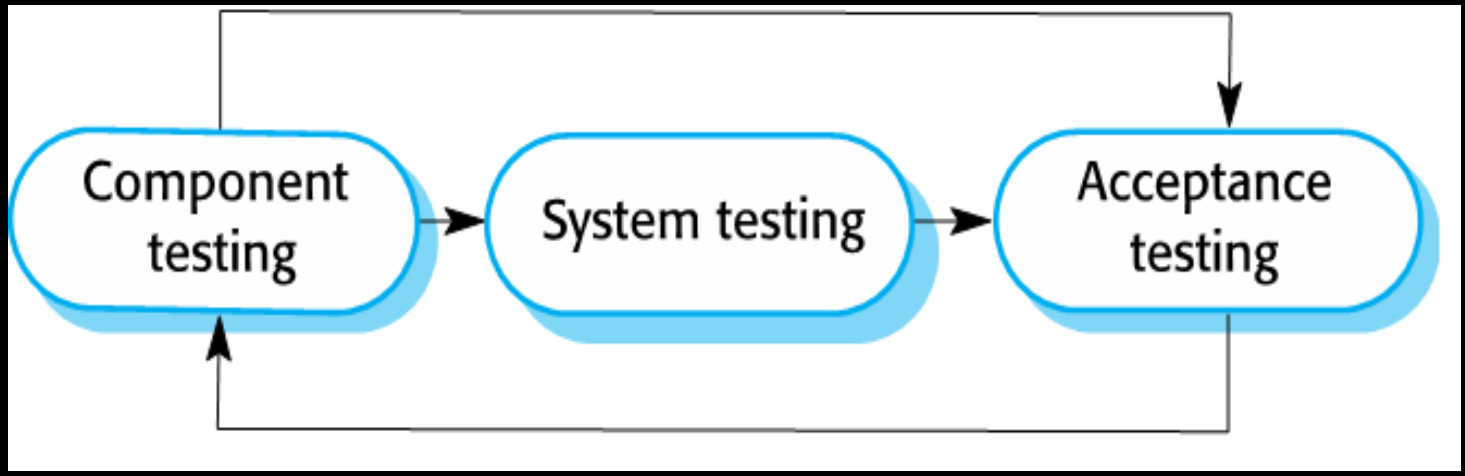
System Implementation

- The software is implemented either by developing a program or programs or by configuring an application system.
- Design and implementation are interleaved activities for most types of software system.
- Programming is an individual activity with no standard process.
- Debugging is the activity of finding program faults and correcting these faults.

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.

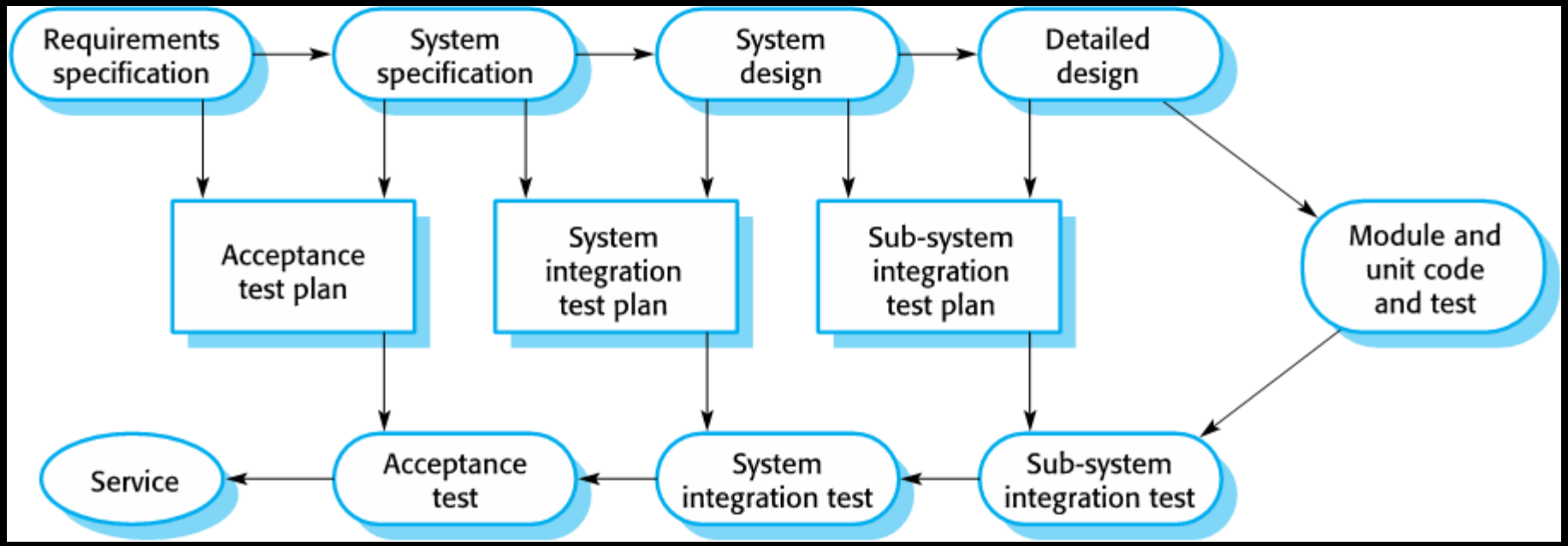
Stages of Testing



Testing Stages

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

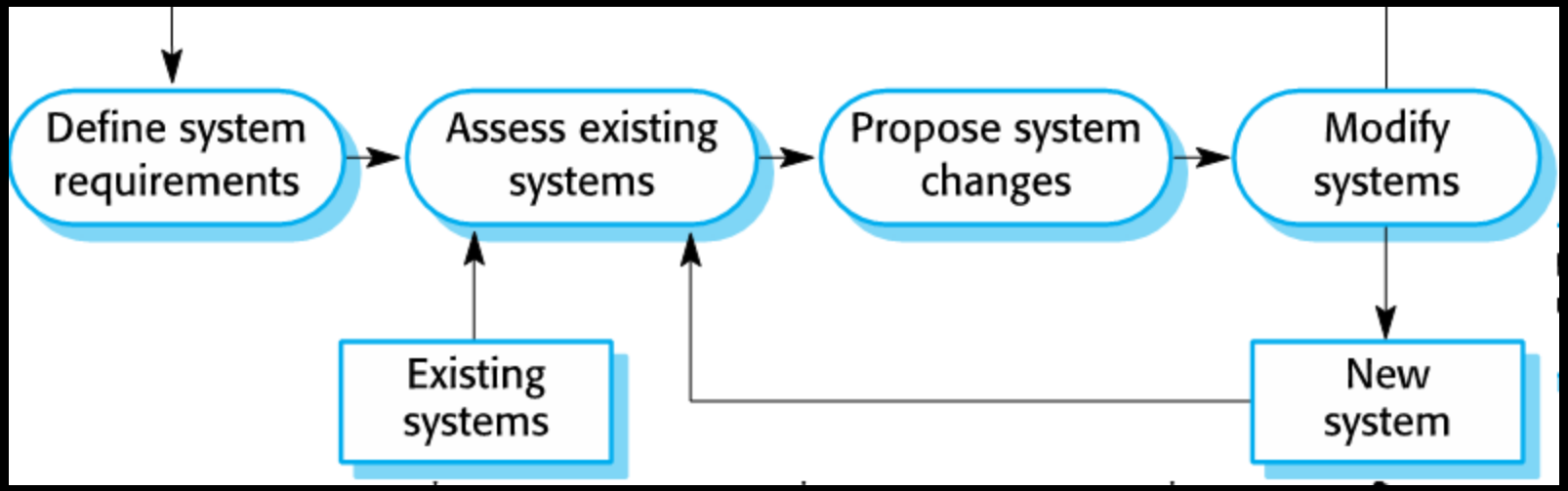
Testing phases in a plan-driven software process (V-model)



Software Evolution

- Software is inherently 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



That is all