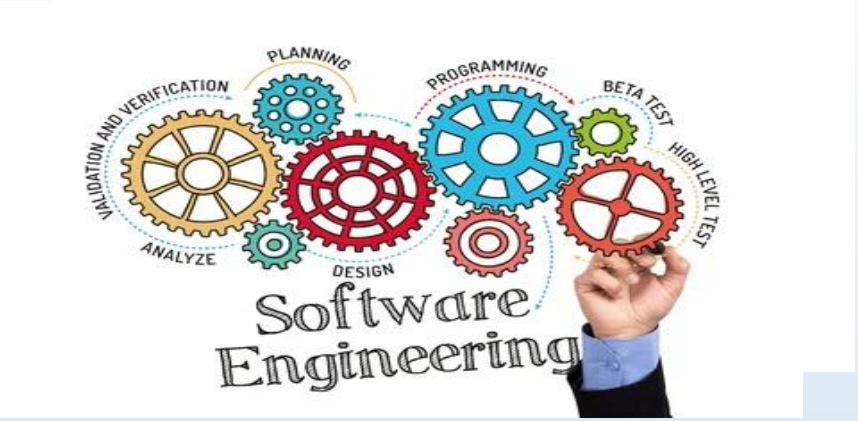
DUT DURBAN UNIVERSITY OF TECHNOLOGY

FACULTY OF ACCOUNTING & INFORMATICS

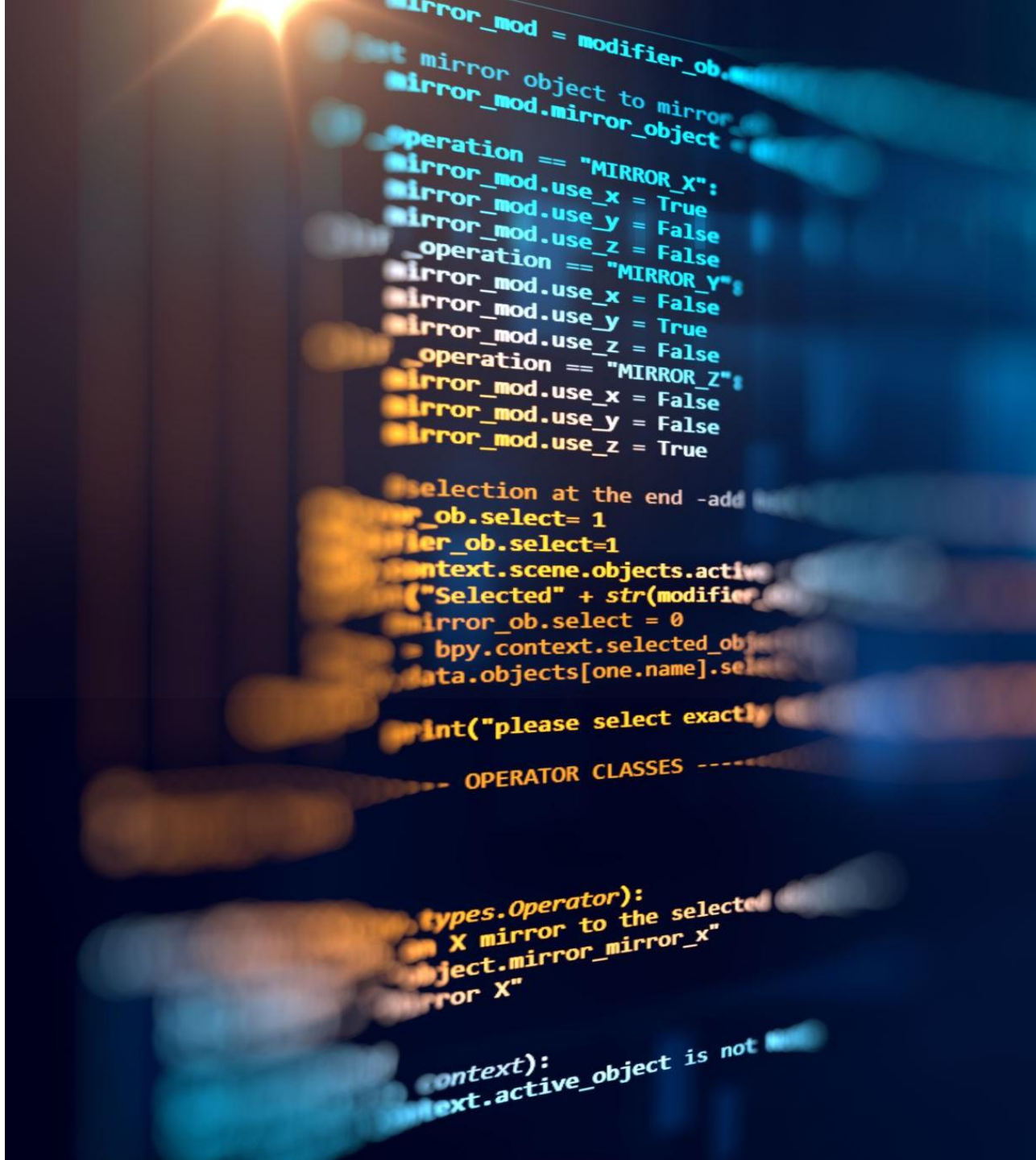


INFORMATION SYSTEMS 3A

SOFTWARE PROCESS

Contents

- Software Process
- Software process models
  - Waterfall model
  - Incremental Development
  - Integration and configuration
- Process activities



Software Process

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

- There are many different software processes, but all must include four activities:
  - Software specification: defining what the system should do.
  - Software design and implementation: defining the organization of the system and implementing the system.
  - Software validation: checking that it does what the customer wants.
  - Software evolution: changing the system in response to changing customer needs.

Software Process

- A software process may also include:
  - Products (outcomes of a process activity)
  - Roles (responsibilities of people)
  - Pre- and post-condition: which are statements that are true before and after a process activity has been enacted or a product produced.

- Software processes are categorized as either **plan-driven** or **agile processes.**

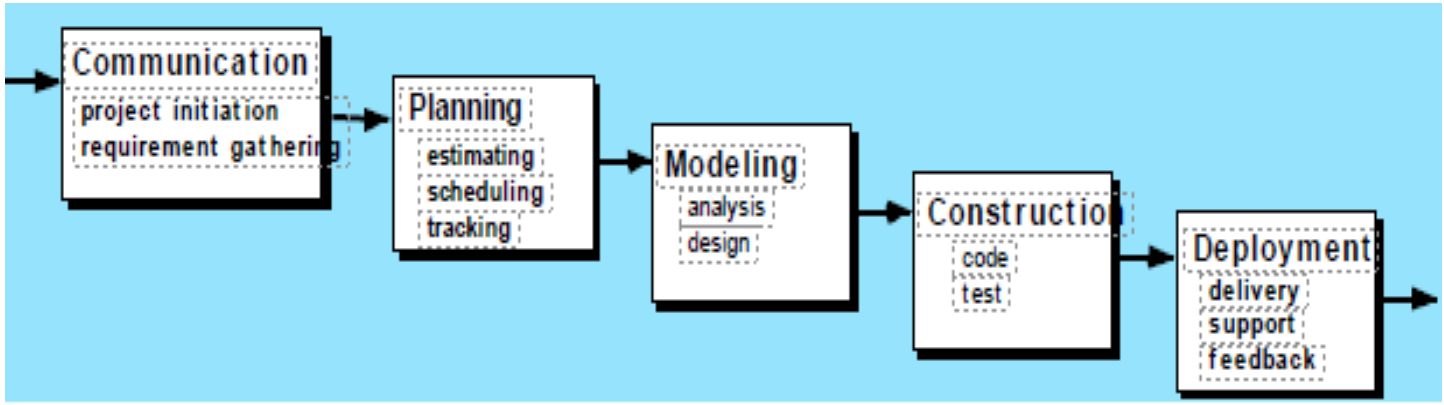
Plan-driven and agile processes

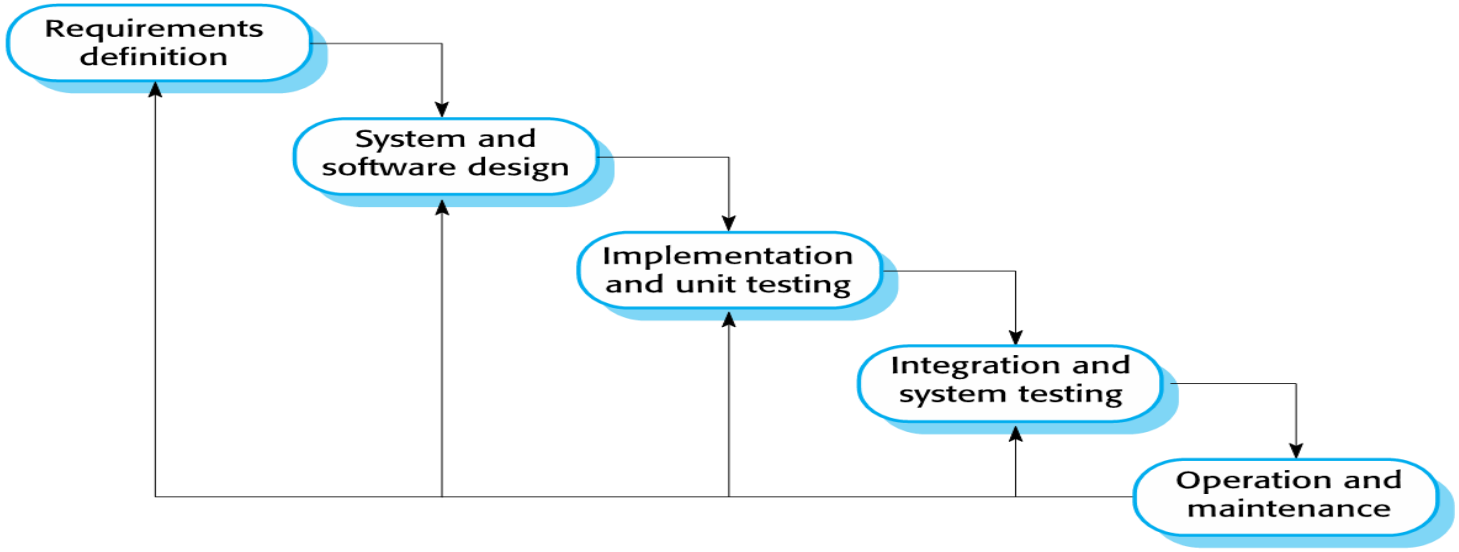
- **Plan-driven processes:** processes where all of the process activities are planned in advance and progress is measured against this plan.
- **Agile processes:** planning is incremental and it is easier to change the process to reflect changing customer requirements

Software Process Models

- A software process model is a simplified representation of a software process.
- Each process model represents a process from a particular perspective and thus provides a partial information about that process.
- Process models covered:
  - The waterfall model
  - Incremental development
  - Reuse-oriented software engineering

The Waterfall Model





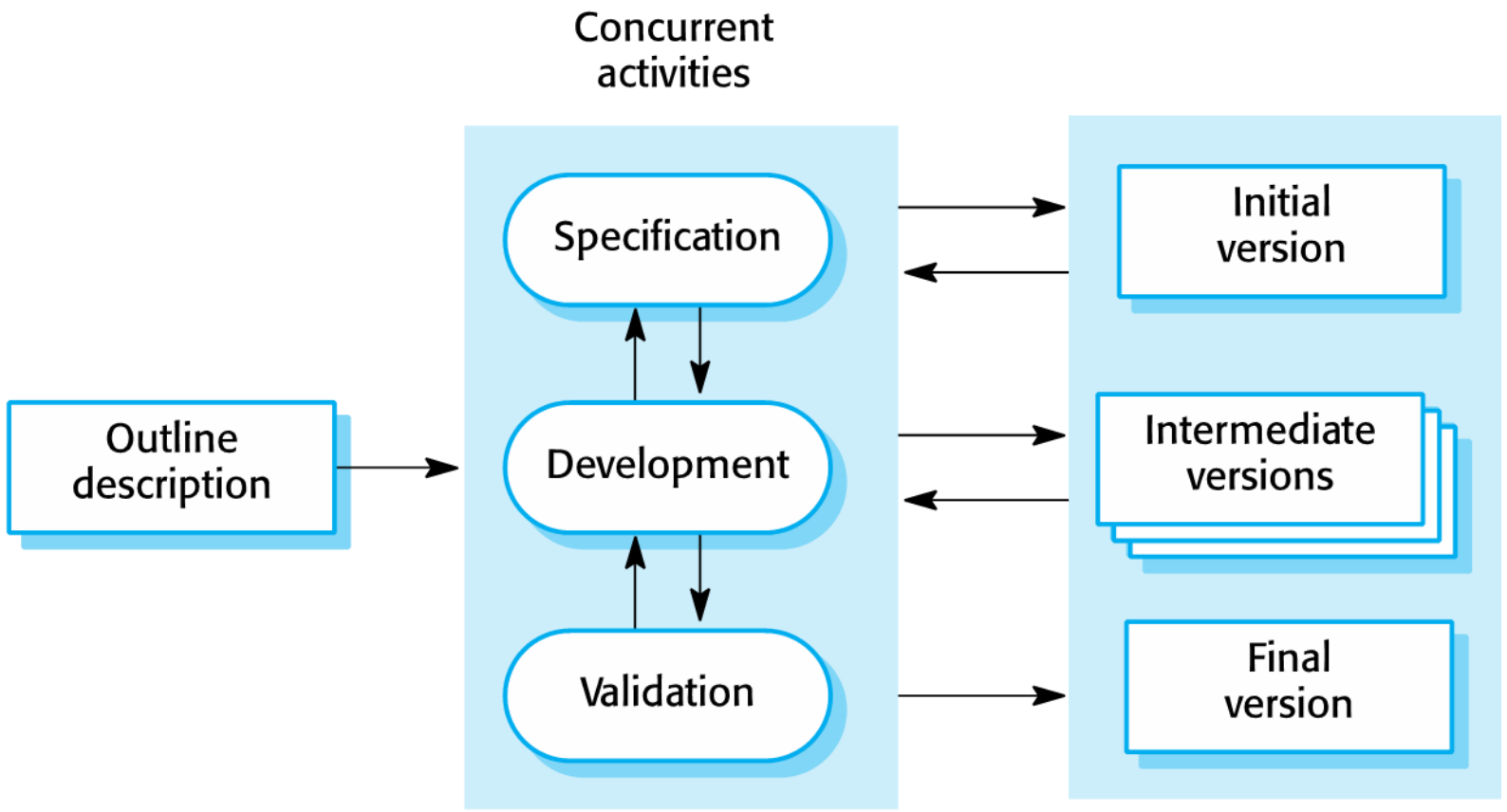
The Waterfall Model

- 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 result of each phase is one or more documents
- The following phase should not start until the previous phase has finished (in practice, they overlap)

The Waterfall Model

- **Benefit**:
- The process is visible
  - Documentation is produced at each phase
- **Problem:**
- 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.

Incremental Development Model



Incremental Development

- Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate (acceptable) system has been developed.
- Generally, the early increments of the system include the most important or most urgently required functionality.
- Can be either plan-driven, agile, or a mixture of both.

Incremental Development

- **Benefits:**
- The cost of accommodating changing customer requirements is reduced
- It is easier to get customer feedback on the development work that has been done key factors for selecting office furniture and creating an optimal working environment
- More rapid delivery and deployment of useful software to the customer is possible

Incremental Development

- **Problems:**
- The process is not visible
  - Managers need regular deliverables to measure progress
- 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.

Characteristics of incremental

- Most common approach
- Used in agile, plan-driven or hybrid
- Allows for discovery of functionality

Integration and configuration

- Based on software reuse where systems are integrated from existing components or 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

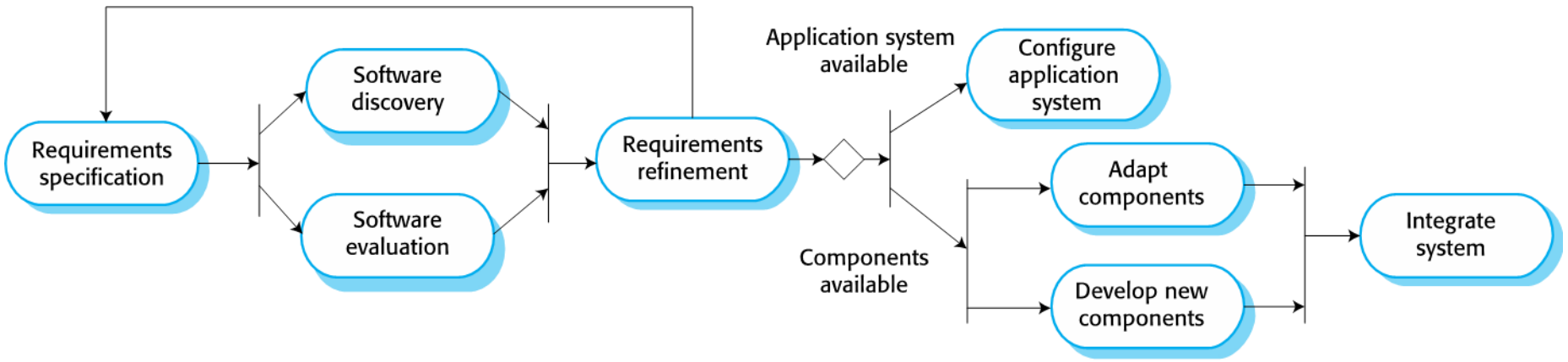
Types of reusable software

- 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 JEE.
- Web services that are developed according to service standards and which are available for remote invocation.

Advantages and disadvantages

- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements

Integration and configuration



Process activities

- The activities needed to develop software

➢Specification

➢Development

➢Validation

➢ Evolution

The process model organizes these activities

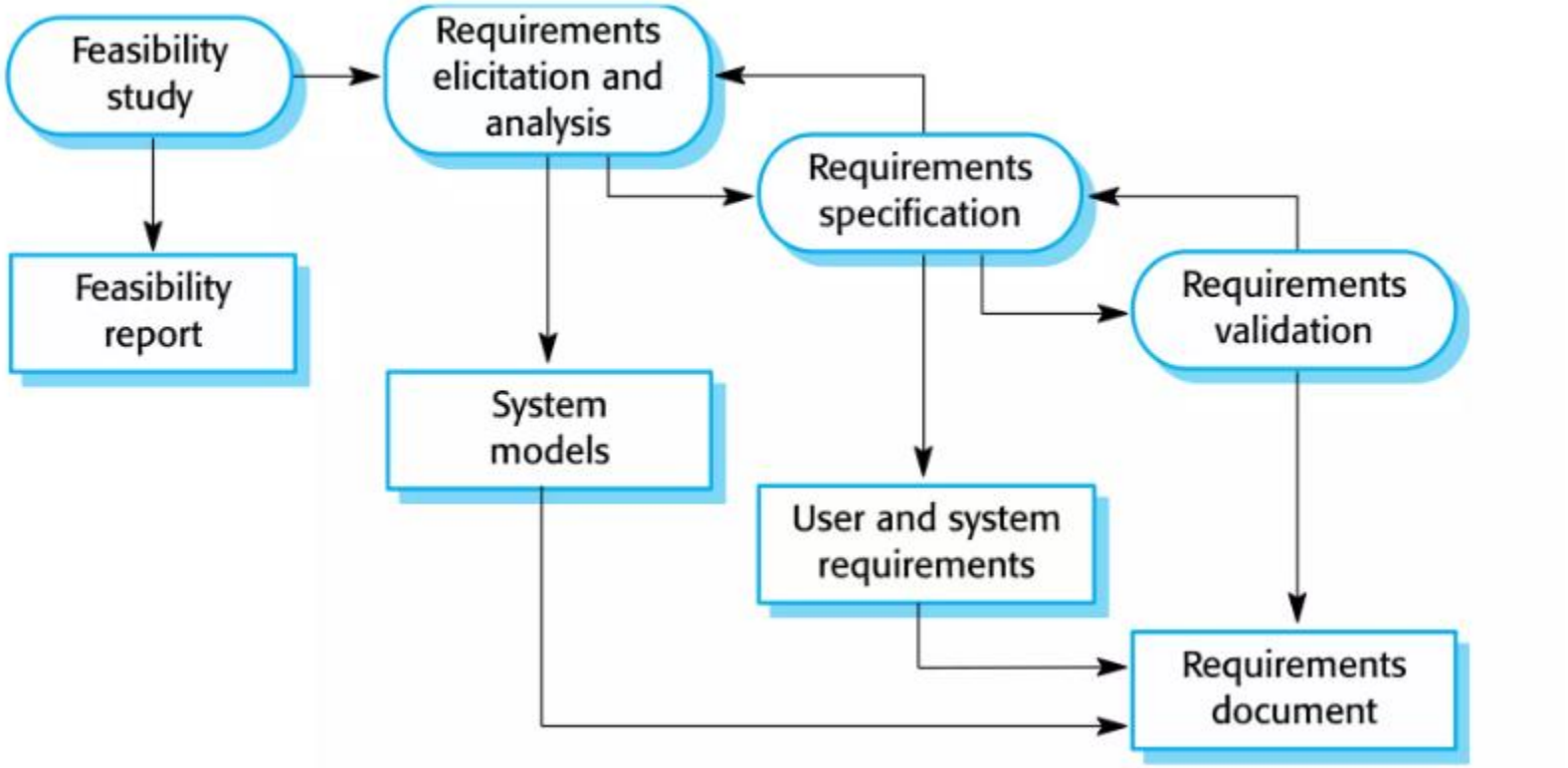
Process activities

- The four basic process activities of **specification**, **development, validation**, and **evolution** 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.
  - Requirements validation: checking the validity of the requirements.

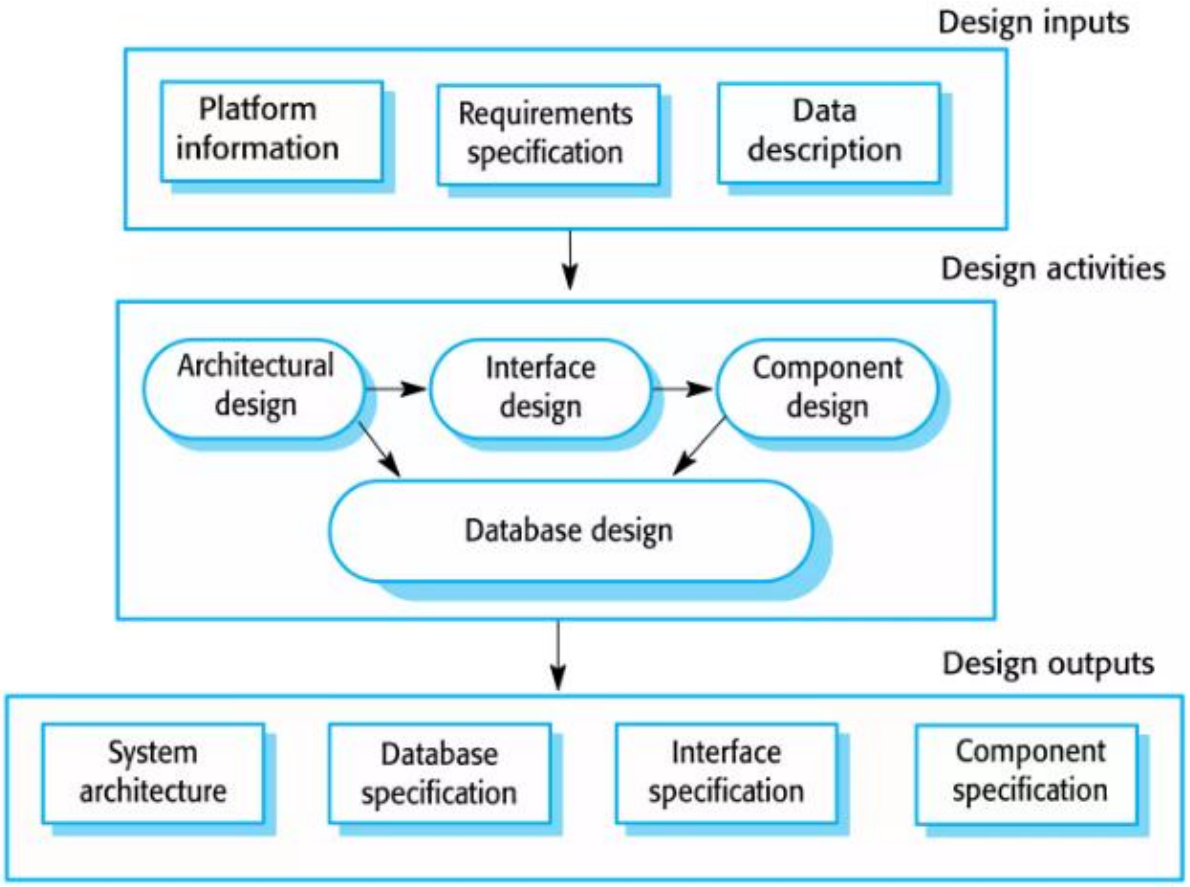
The requirements engineering process



Software design and implementation

- The process of converting the system specification into an executable system.
- **Software design**: design a software structure that realizes 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

- **Component design**, where you take each system component and design how it will operate.
- **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.
- **Database design**, where you design the system data structures and how these are to be represented in a database.

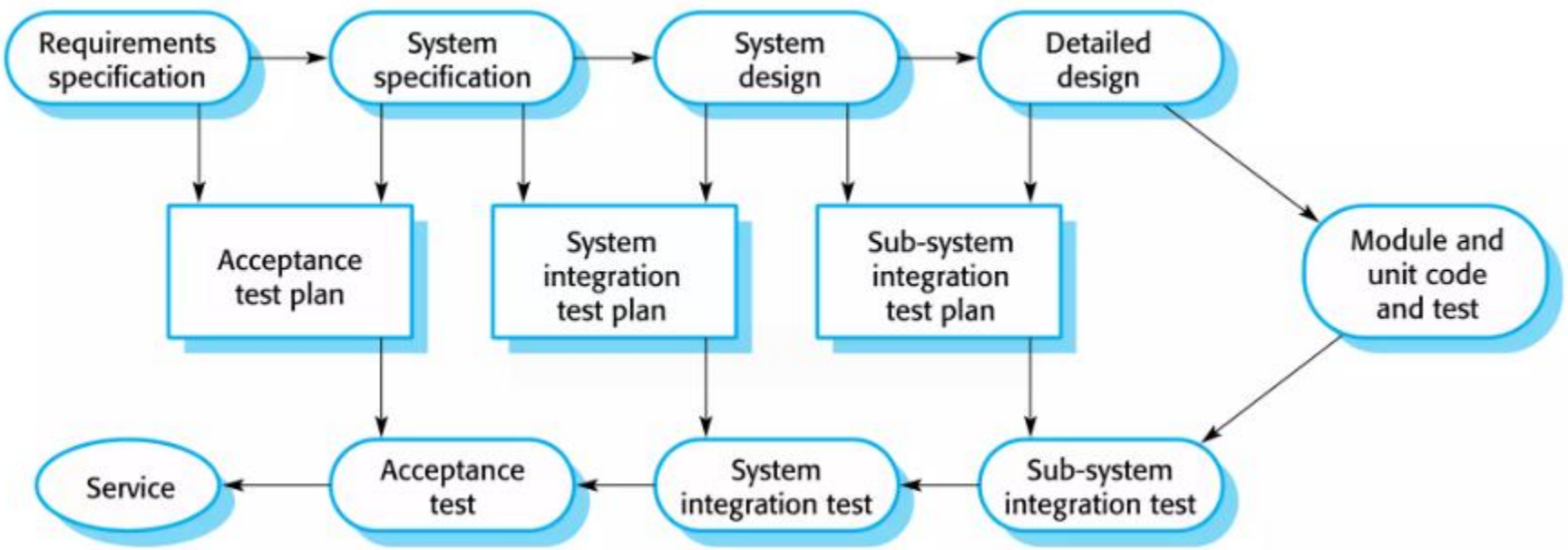
Software validation

- Software Validation is a process of evaluating software product so as to ensure that the software meets the pre defined and specified business requirements as well as the end users/customers' demands and expectations.
- **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.
- Testing is the most commonly used V & V activity.

Software validation Activities (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 plan driven software process



Software evolution

- The software is modified to adapt it to changing customer and market requirement
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.

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.

Two related approaches may be used to reduce the costs of rework:

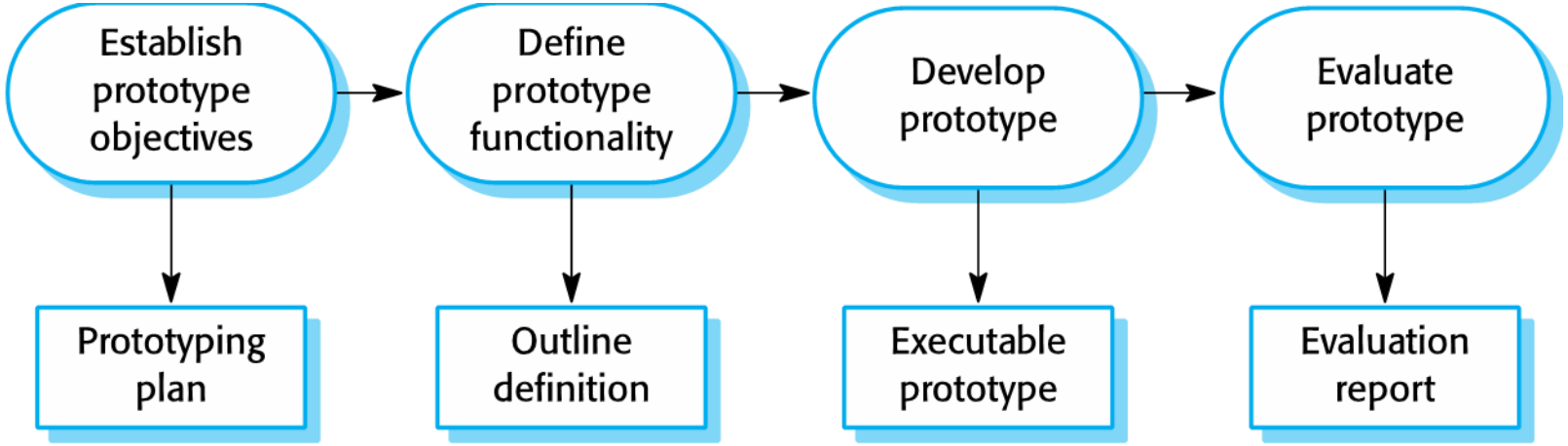
1. Change anticipation: where the software process includes activities that can anticipate possible changes before significant rework is required.
2. Change tolerance : where the process is designed so that changes can be accommodated at relatively low cost.

- **Coping with changing requirements:**

➢ System prototyping

➢ Incremental delivery

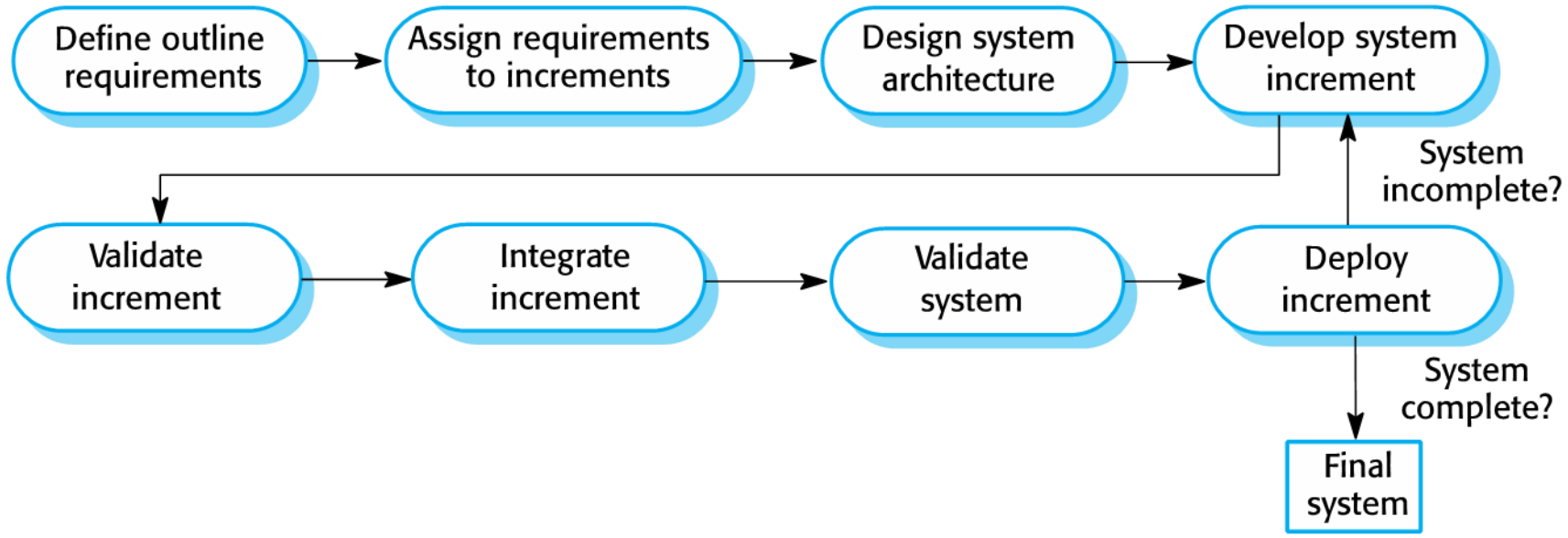
Prototyping



Benefits of prototyping

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

Incremental delivery



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.