Chapter 2

Software Processes

The Software Process

1. Structured set of activities to develop software systems.
2. Key activities:
    - Specification: Defining system requirements.
    - Design and implementation: Designing and building the system.
    - Validation: Ensuring the system meets customer needs.
    - Evolution: Modifying the system to meet changing requirements.

Software Process Models

- Abstract representations of software processes.
- Describe processes from specific perspectives.

Software Process Descriptions

Include:

- Activities (e.g., specifying data models, designing user interfaces)
- Ordering of activities
- Products (outcomes of activities)
- Roles (responsibilities of people involved)
- Pre- and post-conditions (statements about states before and after activities)

Plan-Driven and Agile Processes

- Plan-driven processes: All activities planned in advance, progress measured against plan.
- Agile processes: Incremental planning, easier to adapt to changing requirements.
- Most practical processes combine elements of both approaches.
-

Process Selection

1. No right or wrong software processes.
2. Choice depends on factors such as:
    - Type of system being developed
    - Size and criticality of the system
    - Experience and skills of the development team

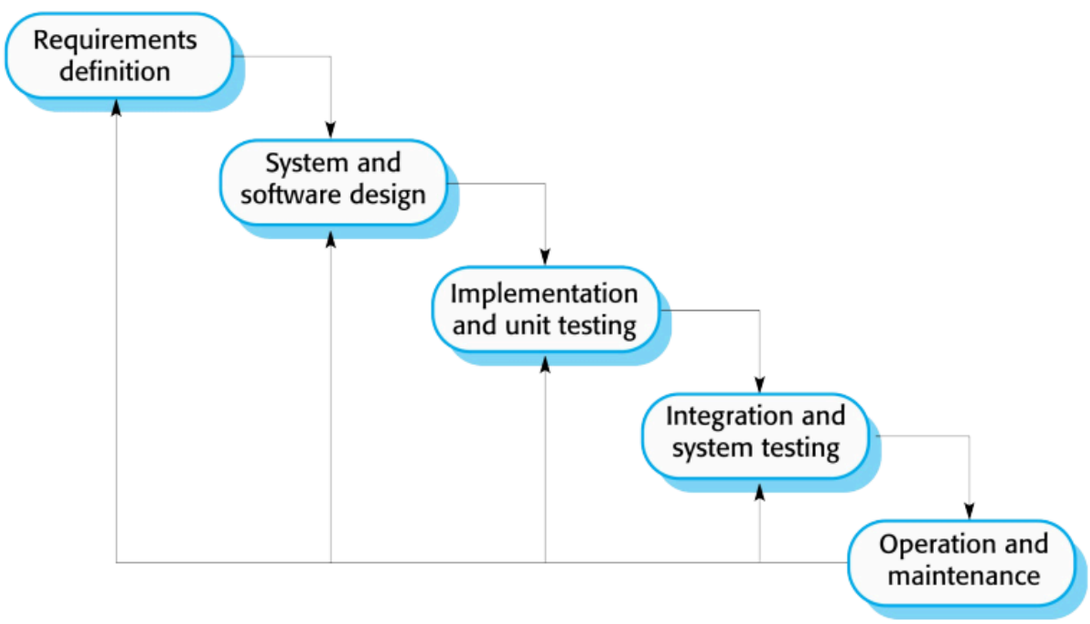
Waterfall Model

Waterfall model is a plan driven model There are separate identified phases in the waterfall model:

- **Requirements analysis and definition:** Gathering and analyzing user needs.
- **System and software design:** Creating a blueprint for the system.
- **Implementation and unit testing:** Building and testing individual software components.
- **Integration and system testing:** Combining components and testing the complete system.
- **Operation and maintenance:** Deploying the system and addressing any issues that arise.

Waterfall Model Problems

1. **Inflexible:** Difficult to accommodate changes once a phase is complete.
2. **Only suitable for projects with stable requirements:** Few business systems have stable requirements.
3. **Mostly used for large systems engineering projects:** Coordinates work at multiple sites.



Incremental Development

A software development approach where the system is built and delivered in increments, with each increment adding new functionality or fixing defects.

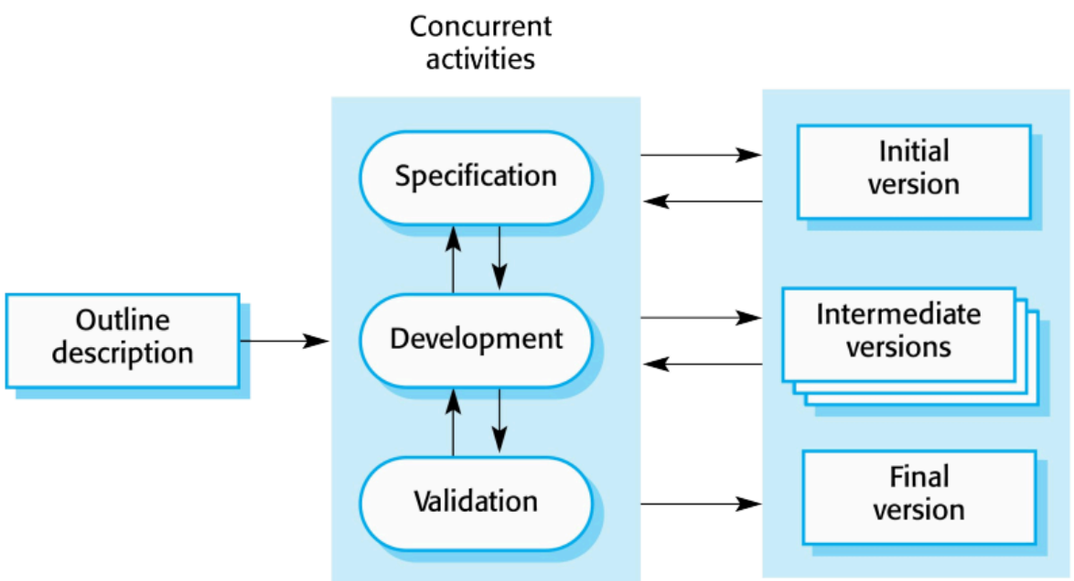
- Interleaved specification, development, and validation.
- May be plan-driven or agile.

Incremental Development Benefits

- **Reduced cost of change:** Less analysis and documentation to redo.
- **Improved customer feedback:** Customers can comment on demos and see progress.
- **Rapid delivery:** Customers can use software earlier.

Incremental Development Problems

- **Lack of visibility:** Managers need regular deliverables to measure progress.
- **Degraded system structure:** New increments can corrupt the system structure if not refactored.



Integration and Configuration

A software development approach based on software reuse, where systems are integrated from existing components or application systems **COTS** (Commercial-off-the-shelf-Systems) Reused elements can be configured to adapt their behavior and functionality to user requirements.

Types of Reusable Software

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

Integration Key Process Stages

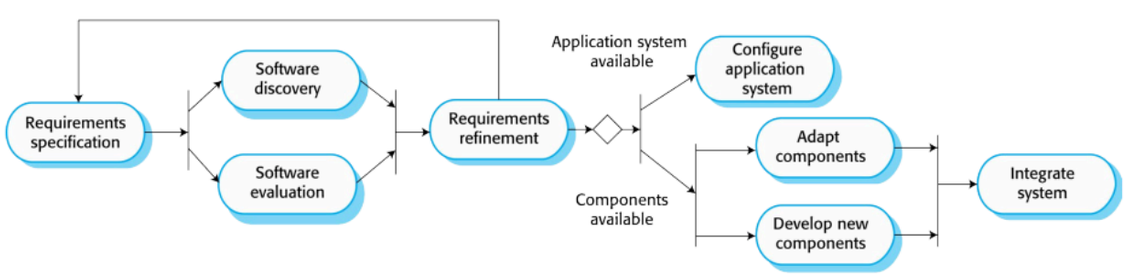
- Requirements specification
- Software discovery and evaluation
- Requirements refinement
- Application system configuration
- Component adaptation and integration

Advantages

1. Reduced costs and risks
2. Faster delivery and deployment

Disadvantages

1. Requirements compromises
2. Loss of control over evolution of reused components



Process Activities

Real software processes involve interleaved technical, collaborative, and managerial activities. Basic process activities:

I. **Specification**
II. **Development**
III. **Validation**
IV. **Evolution.**

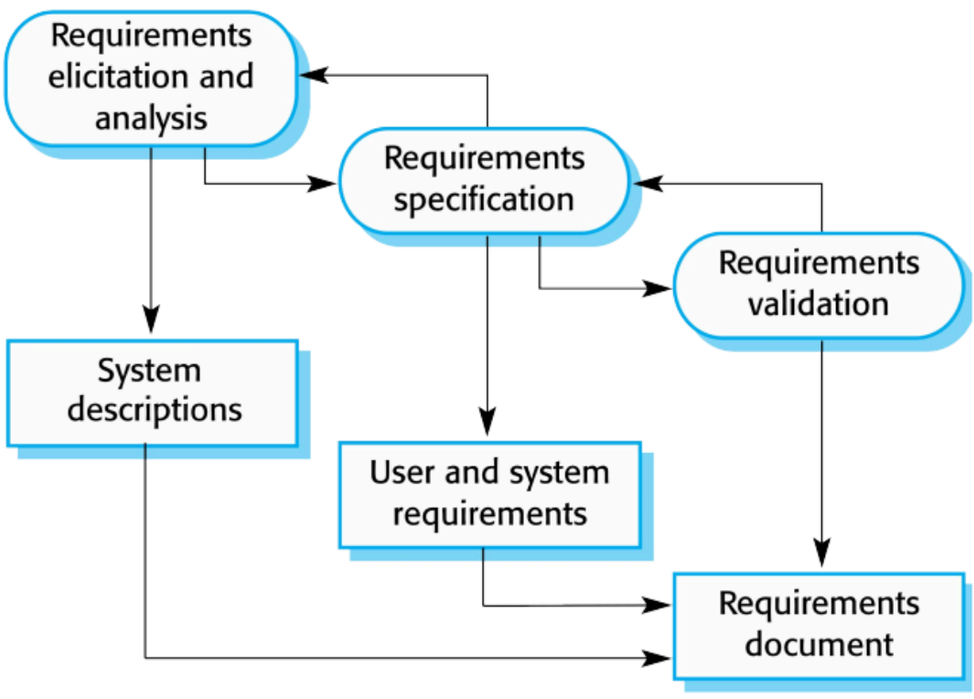
For example, in the waterfall model, they are organized in sequence, whereas in incremental development they are interleaved.

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: Determine stakeholder needs and expectations.
- Requirements specification: Define requirements in detail.
- Requirements validation: Check validity of requirements.



Development

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 interleaved.

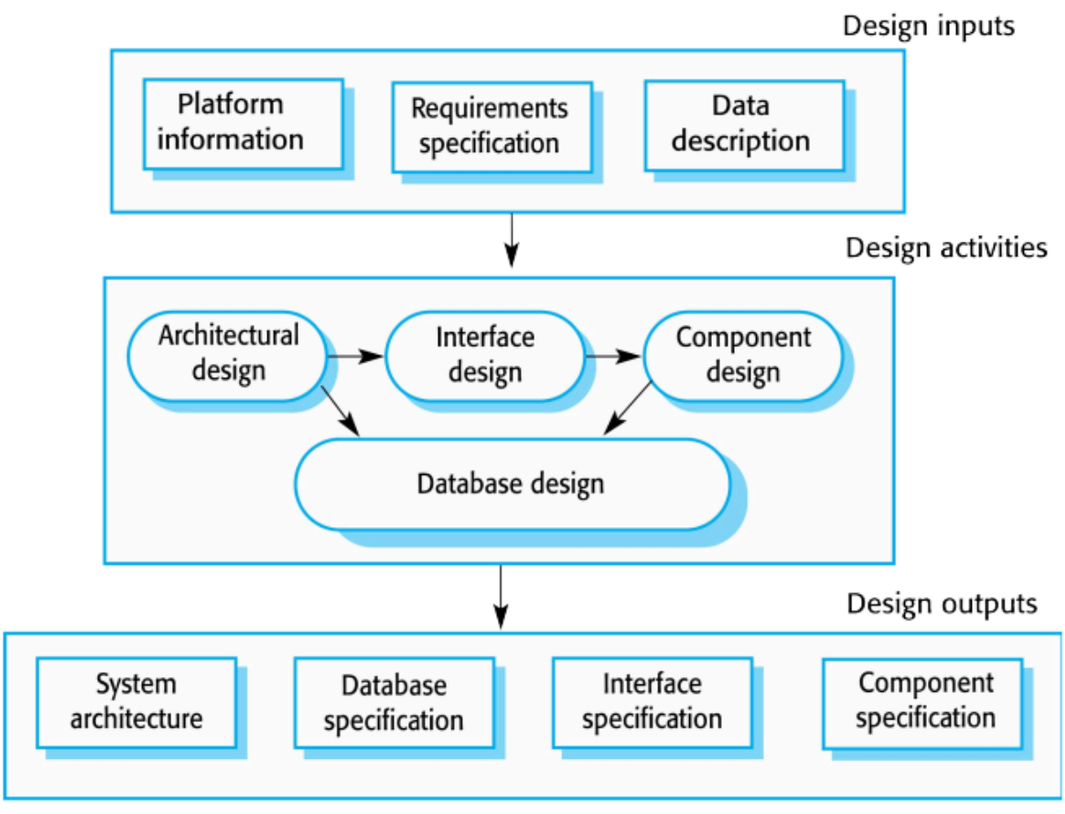
Design Activities

1. **Architectural design:** Overall system structure, components, relationships, distribution
2. **Database design:** System data structures and database representation
3. **Interface design:** Interfaces between system components
4. **Component selection and design:** Search for and design reusable components

System Implementation

Implement software by developing programs or configuring an application system

- **Programming:** Individual activity with no standard process
- **Debugging:** Finding and correcting program faults



Validation

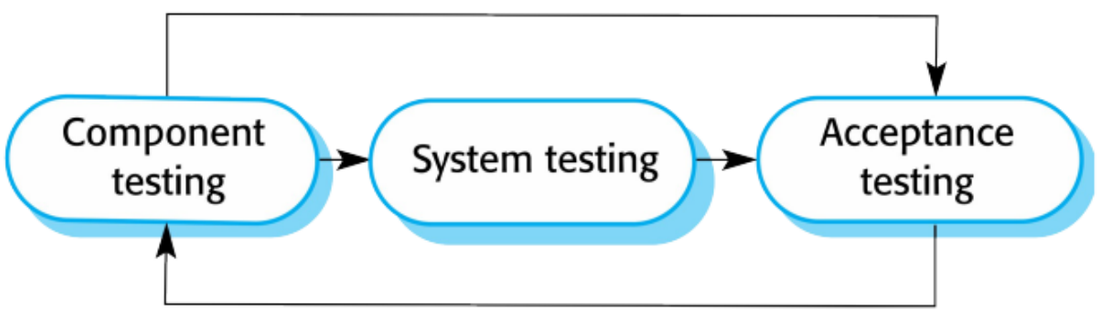
Verification and validation (V&V) is the process of ensuring that a software system conforms to its specification and meets the requirements of the system customer.

Key Points:

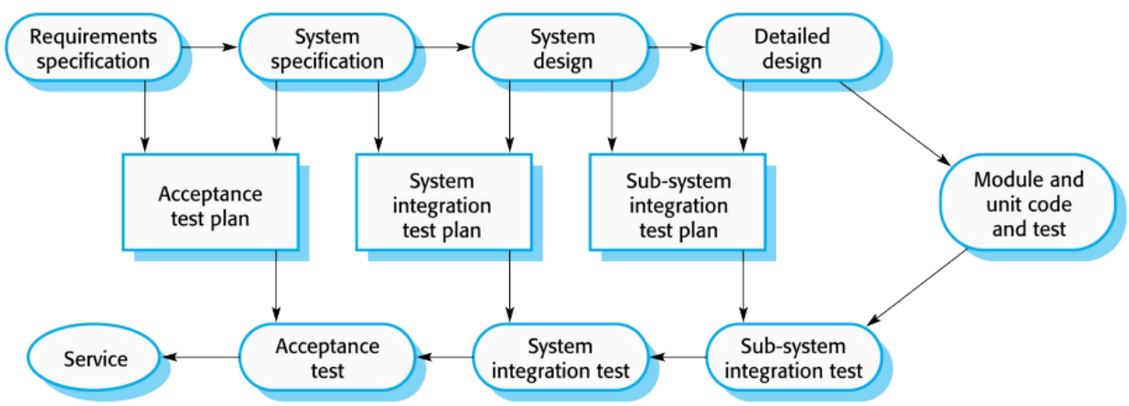
- V&V involves checking and reviewing processes and system testing.
- System testing is the most commonly used V&V activity.
- Testing is essential for ensuring the quality and reliability of software systems.

Testing Stages

1. **Component testing:** Testing individual components independently
2. **System testing:** Testing the system as a whole
3. **Customer testing:** Testing with customer data to ensure the system meets their needs



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



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.

Key Points:

- Software is not static, but rather evolves over time.
- Evolution is necessary to keep software systems up-to-date with changing requirements.
- The distinction between development and evolution is becoming increasingly irrelevant.