Computers II Lesson 2

2.0 Software Process Models

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

These activities may involve the development of software from scratch in a standard programming language like Java or C. However, business applications are not necessarily developed in this way. New business software is now often developed by extending and modifying existing systems or by configuring and integrating off-the-shelf software or system components.

There are many different software processes but all must include four activities that are fundamental to software engineering:

1. Software specification - the functionality of the software and constraints on its operation must be defined.

2. Software design and implementation - The software to meet the specification must be produced.

3. Software validation - The software must be validated to ensure that it does what the customer wants.

4. Software evolution - The software must evolve to meet changing customer needs.

2.1 Model Examples

1. The waterfall model - This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing, and so on.

2. Incremental development - This approach interleaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.

3. Reuse-oriented software engineering - This approach is based on the existence of a significant number of reusable components. The system development process focuses on integrating these components into a system rather than developing them from scratch.

These models are not mutually exclusive and are often used together, especially for large systems development.

For large systems, it makes sense to combine some of the best features of the waterfall and the incremental development models.

Waterfall Model Stages:

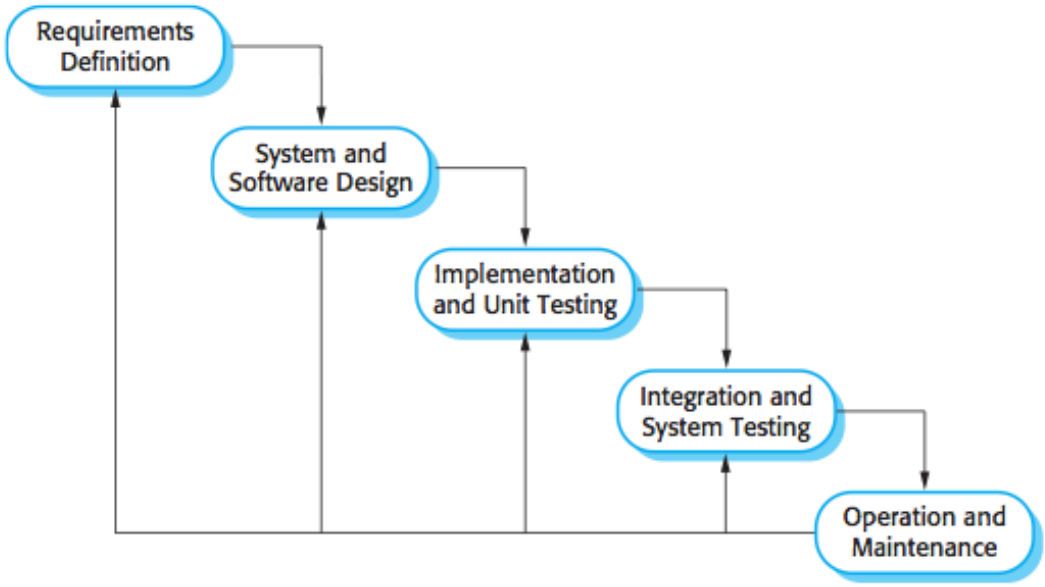
1. Requirements analysis and definition - The system's services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

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 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.



Things to Remember about the Waterfall Model:

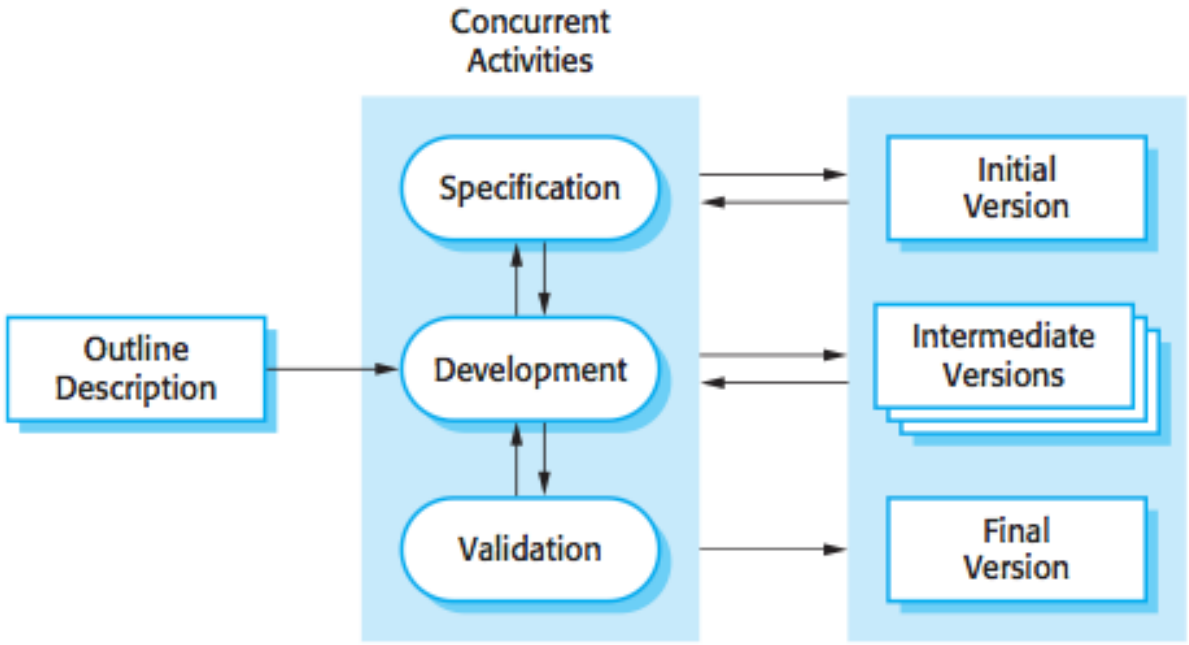
- Documentation is produced at each phase.
- This makes the process visible so managers can monitor progress against the development plan.
- Its major problem is the inflexible partitioning of the project into distinct stages. (Commitments must be made at an early stage in the process, which makes it difficult to respond to changing customer requirements.)
- The waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development.

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 system has been developed

- Incremental software development is better than a waterfall approach for most business, e-commerce, and personal systems.

- Incremental development reflects the way that we solve problems. We rarely work out a complete problem solution in advance but move toward a solution in a series of steps, backtracking when we realize that we have made a mistake.
- Developing the software incrementally is cheaper and easier because we can make changes in the software as it is being developed.



Benefits of Incremental Development:

1. 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.

2. 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. Customers find it difficult to judge progress from software design documents.

3. More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Reuse-oriented software engineering:

- In the majority of software projects, there is some software reuse.

This often happens informally when people working on the project know of designs or code that are similar to what is required. They look for these, modify them as needed, and incorporate them into their system.

Reuse-oriented approaches rely on a large base of reusable software components and an integrating framework for the composition of these components.

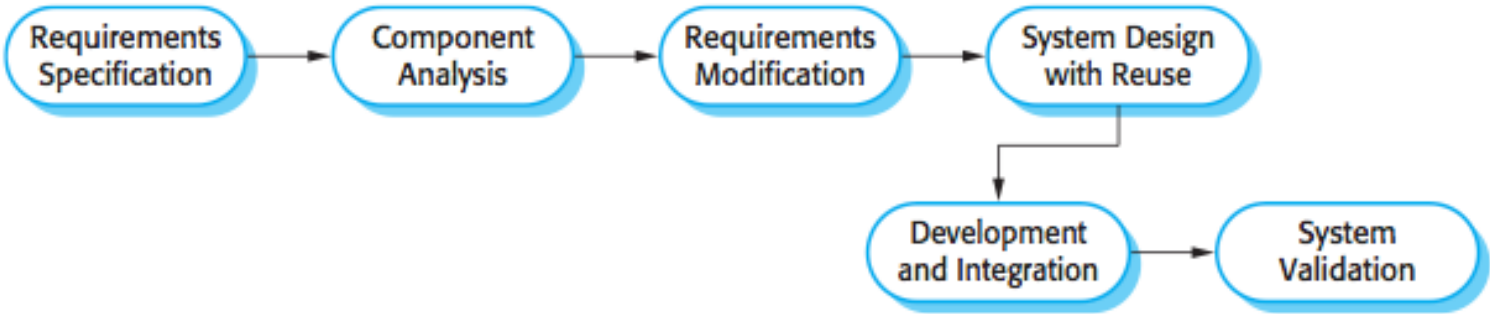
Reuse-oriented stages:

1. Component analysis - Given the requirements specification, a search is made for components to implement that specification. Usually, there is no exact match and the components that may be used only provide some of the functionality required.

2. Requirements modification - The requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components. Where modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.

3. System design with reuse - The framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and organize the framework to cater for this. Some new software may have to be designed if reusable components are not available.

4. Development and integration - Software that cannot be externally procured is developed, and the components are integrated to create the new system. System integration, in this model, may be part of the development process rather than a separate activity.


Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation