

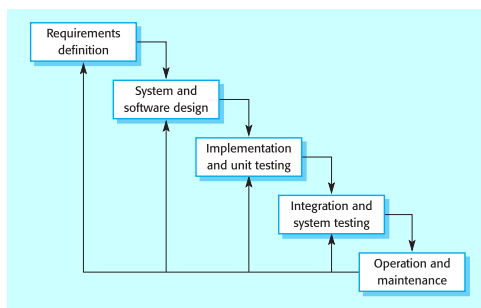
The software process

- A structured set of activities required to develop a software system
 - Specification;
 - Design;
 - Validation;
 - Evolution.
- A software process model is an abstract representation of a process. 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.
- Evolutionary development
 - Specification, development and validation are interleaved.
- Component-based software engineering
 - The system is assembled from existing components.
- There are many variants of these models e.g. formal development where a waterfall-like process is used but the specification is a formal specification that is refined through several stages to an implementable design.

Waterfall model



Waterfall model phases

- 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. One phase has to be complete before moving onto the next phase.

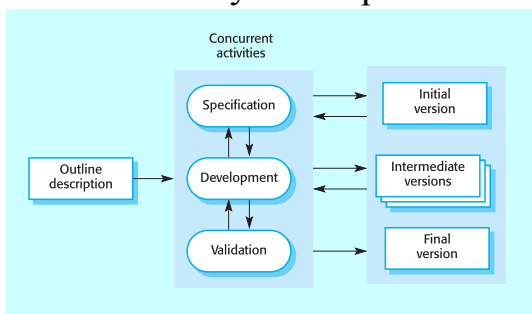
Waterfall model problems

- responding to changes in customer requirements
- difficult to define the problem up-front
- getting customer feedback -> sometimes only until end
 - documents produced at end of stage may not be meaningful to customer
- Appropriate
 - requirements are well-understood
 - changes will be fairly limited during the design process.
 - mostly used *as is* for large systems engineering projects where a system is developed at several sites.

Evolutionary development

- Exploratory development
 - Objective is to work with customers and to evolve a final system from an initial outline specification. Should start with well-understood requirements and add new features as proposed by the customer.
- Throw-away prototyping
 - Objective is to understand the system requirements. Should start with poorly understood requirements to clarify what is really needed.

Evolutionary development



Evolutionary development

- Problems
 - Lack of process visibility;
 - Systems are often poorly structured;
 - Special skills (e.g. in languages for rapid prototyping) may be required.
- Applicability
 - For small or medium-size interactive systems;
 - For parts of large systems (e.g. the user interface);
 - For short-lifetime systems.

Component-based 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.
- This approach is becoming increasingly used as component standards have emerged.

Process iteration

- System requirements ALWAYS evolve in the course of a project so process iteration where earlier stages are reworked is always part of the process for large systems.
- Iteration can be applied to any of the generic process models.
- Two (related) approaches
 - Incremental delivery;
 - Spiral development.

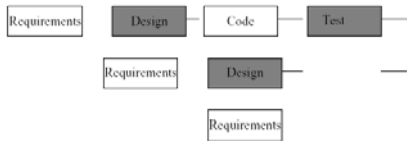
Incremental delivery

- 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 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.

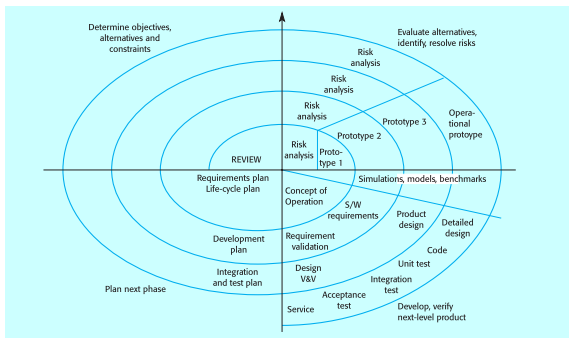
Incremental Waterfall



Spiral development

- 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.

Spiral model of the software process



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.
 - main advantage of spiral model: explicit risk analysis
- 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.

- How do you choose the right model?
- Capability Maturity Model

Extreme programming

- An approach to development based on the development and delivery of very small increments of functionality.
- Relies on constant code improvement, user involvement in the development team and pairwise programming.
- Test driven
- Covered in Chapter 17...