Software Lifecycles

Object-Oriented Analysis and Design

CSCI 6448 - Fall 1998

Kenneth M. Anderson

Software Lifecycle

- A series of steps marking the progress of a software product
- Lifetimes range from days to years
- Consists of
  - people!
  - overall process
  - intermediate products
  - stages of the process

Software Production Personnel

- Client
  - individual or organization that commissions the development of a product
- Developer
  - organization producing the product
- User(s)
  - person who authorizes the client to contract the developer
  - person(s) who will utilize the software in operation

Examples

- internal software development

  client = developer
- contract software development

  client ≠ developer

What is a process?

- Device for producing a product
- An instance of a process description
  - Each description describes a wide class of instances
  - Process descriptions are used to solve classes of problems
- Thus
  - software processes are devices for creating and evolving software products

Intermediate Software Products

- Objectives
  - Demarcate end of phases
  - Enable effective reviews
  - Specify requirements for next phase
- Form
  - Rigorous
  - Machine processible (highly desirable)
- Content
  - Specifications, Tests, Documentation

Phases of a Software Lifecycle

- Standard Phases
  - Requirements Analysis & Specification
  - Design
  - Implementation and Integration
  - Operation and Maintenance
  - Change in Requirements
  - Testing throughout!
- Phases promote manageability and provide organization

Requirements Analysis and Specification

- Problem Definition —> Requirements Specification
  - determine exactly what client wants and identify constraints
  - develop a contract with client
  - Specify the product's task explicitly
- Difficulties
  - client asks for wrong product
  - client is computer/software illiterate
  - specifications may be ambiguous, inconsistent, incomplete
- Validation
  - extensive reviews to check that requirements satisfy client needs
  - look for ambiguity, consistency, incompleteness
  - check for feasibility, testability
  - develop system/acceptance test plan

Design

- Requirements Specification —> Design
  - develop architectural design (system structure)
    - decompose software into modules with module interfaces
  - develop detailed design (module specifications)
    - select algorithms and data structures
  - maintain record of design decisions
- Difficulties
  - miscommunication between module designers
  - design may be inconsistent, incomplete, ambiguous
- Verification
  - extensive design reviews (inspections) to determine that design conforms to requirements
  - check module interactions
  - develop integration test plan

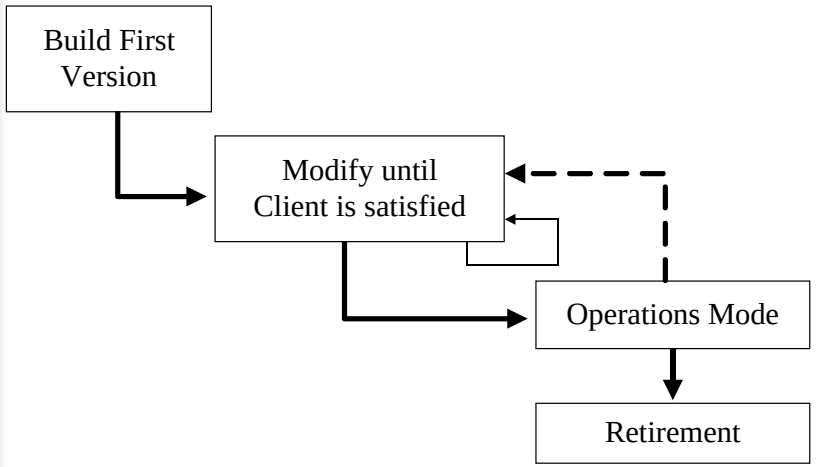
Implementation and Integration

- Design —> Implementation
  - implement modules and verify they meet their specifications
  - combine modules according to architectural design
- Difficulties
  - module interaction errors
  - order of integration has a critical influence on product quality
- Verification and Testing
  - code reviews to determine that implementation conforms to requirements and design
  - develop unit/module test plan: focus on individual module functionality
  - develop integration test plan: focus on module interfaces
  - develop system test plan: focus on requirements and determine whether product as a whole functions correctly

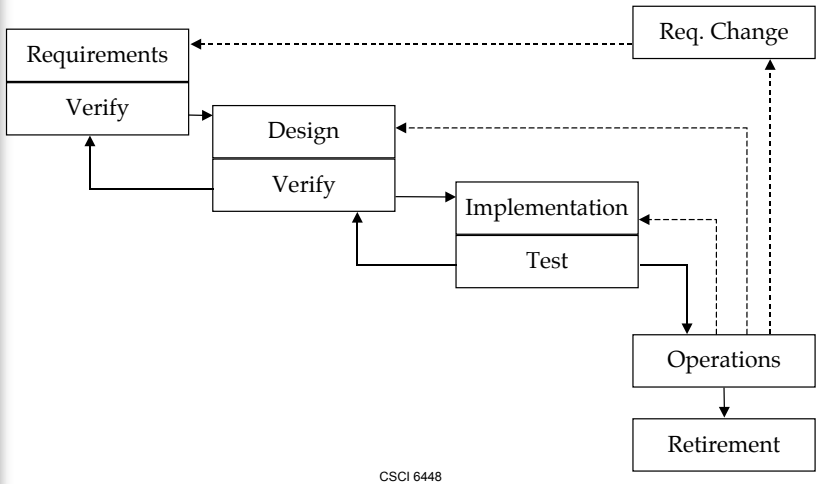
Operation and Maintenance

- Operation —> Change
  - maintain software after (and during) user operation
  - determine whether product as a whole still functions correctly
- Difficulties
  - design not extensible
  - lack of up-to-date documentation
  - personnel turnover
- Verification and Testing
  - review to determine that change is made correctly and all documentation updated
  - test to determine that change is correctly implemented
  - test to determine that no inadvertent changes were made to compromise system functionality (check that no affected software has regressed)

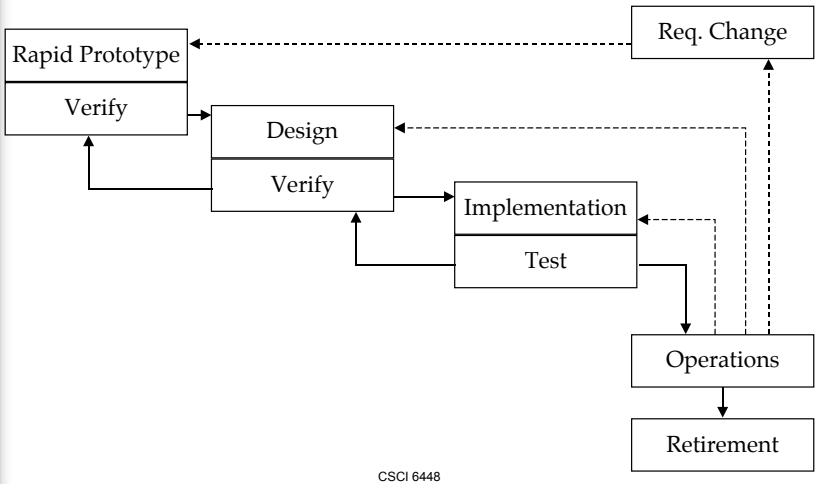
Build-and-Fix



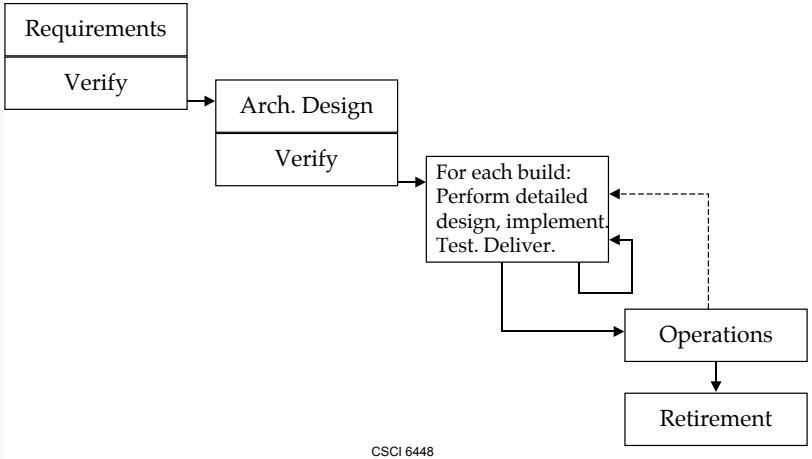
Waterfall Model (in essence)



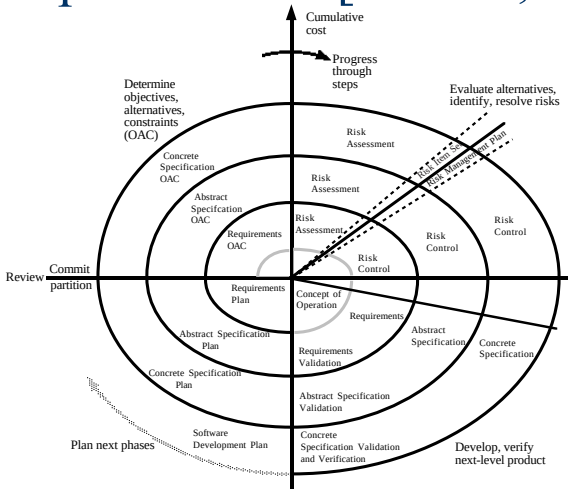
Rapid Prototyping



Incremental



The Spiral Model [Boehm,1988]



Next Lecture

- Object-Oriented Software Lifecycles
  - In particular, Objectory
    - Process being developed by the “three amigos”
- Relate phases to
  - Techniques
  - UML notations

First Team Assignment

- Produce an informal description of the requirements for your team project
- Describe
  - Project Background and Scope
  - Expected Users
  - Skills of Team Members
- See website for more information