Life Cycle Models

V. Paúl Pauca

Department of Computer Science
Wake Forest University

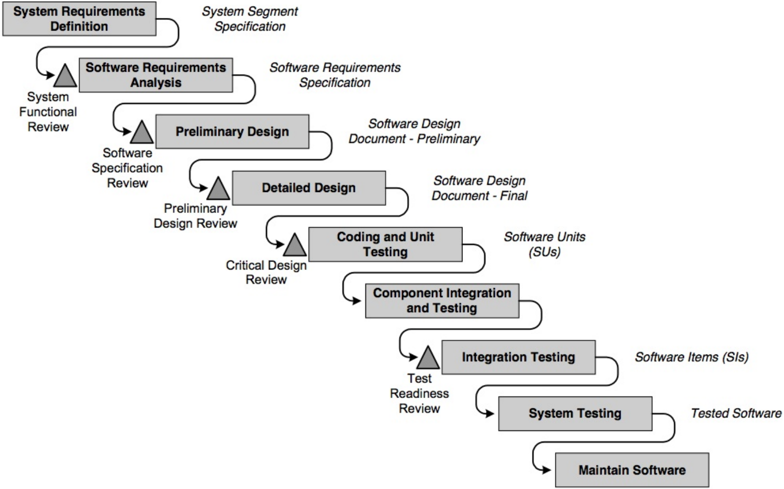
CSC 331-631
Fall 2013

Software Life Cycle

The overall framework in which software is conceived, developed, and maintained.

- Life cycles are also referred to as *models*
- Phases: At its most basic, a life cycle includes:
  - Design
  - Development
  - Maintenance
- Classic life cycle models:
  - Waterfall model
  - Incremental model
  - Spiral model

Waterfall Model I


System Requirements Definition
System Segment Specification
Software Requirements Analysis
Software Requirements Specification
System Functional Review
Preliminary Design
Software Design Document - Preliminary
Software Specification Review
Detailed Design
Software Design Document - Final
Preliminary Design Review
Coding and Unit Testing
Software Units (SUs)
Critical Design Review
Component Integration and Testing
Integration Testing
Software Items (SIs)
Test Readiness Review
System Testing
Tested Software
Maintain Software

From the GSAM Handbook

Waterfall Model II

Characteristics

- First used on DoD projects in the 1970s
- Highly structured sequential development process
- Documentation-driven and document-intensive
- Initial phases document what must be done
- Later phases define how it should be done

Waterfall Model III

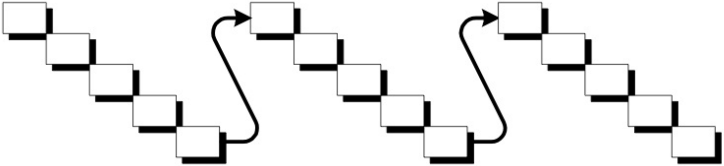
Advantages

- System is well documented
- Corresponding development and management phases
- Relatively accurate cost and schedule estimates

Disadvantages

- Risk dealt within a single cycle
- Local feedback between phase transitions only
- Working product available in the latter stages
- Progress and success hard to observe until later stages
- An early error may be discovered only after delivery
- Fixes must wait until maintenance

Incremental Model I



Iteration:

- Essentially a series of waterfall models
  - Generate release V_1, then revise and generate V_2, and so on.
  - Each successive release V_k is intended to be closer to its target than its predecessor
- Each release adds more functionality

Incremental Model II

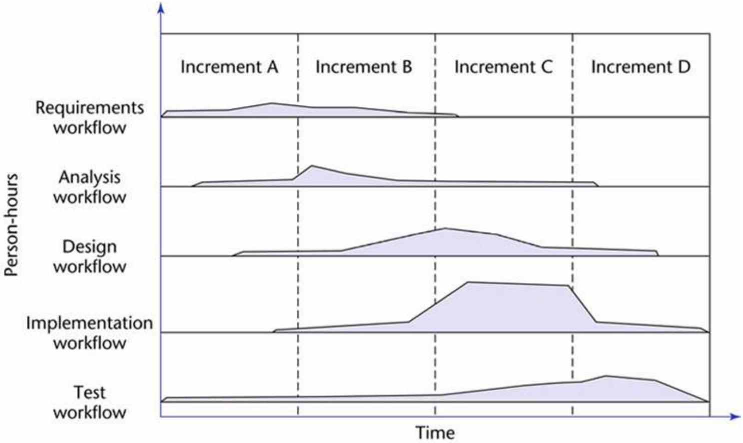
Advantages:

- Provides feedback from one cycle to the next
- Allows modification and addition of requirements
- More responsive to user needs
- Risk spread out over multiple cycles
- Testing may be easier on each iteration

Disadvantages:

- Many requirements still need to be know early
- Interface between cycles must be well defined
- Operations may be impacted as each release is deployed

Incremental Workflow Model I



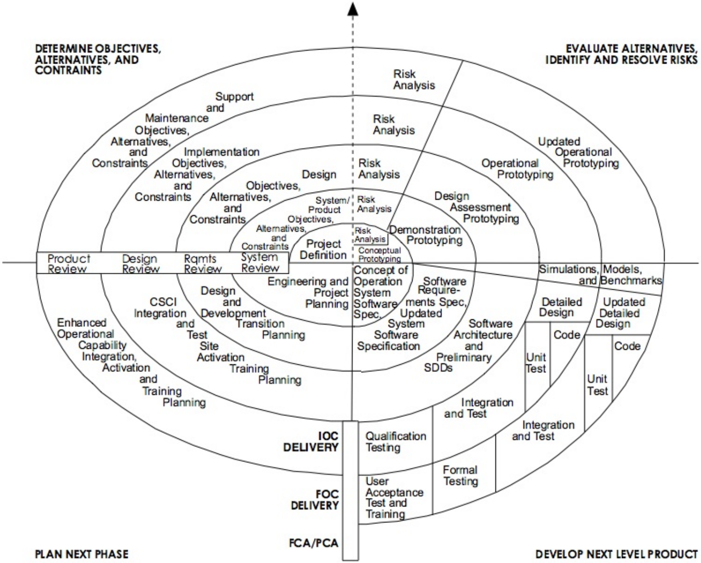
The area under the curves indicate effort spent for each workflow over many increments.

Incremental Workflow Model II

Characteristics

- five core workflows are performed over the entire life cycle
- At most times one workflow predominates
- Examples:
  - At the beginning, the requirements workflow predominates
  - At the end, the implementation and test workflows predominate
- Planning and documentation activities are (should be) performed throughout the life cycle

Spiral Model I


DETERMINE OBJECTIVES, ALTERNATIVES, AND CONTRAINTS
EVALUATE ALTERNATIVES, IDENTIFY AND RESOLVE RISKS
Risk Analysis
Support and Maintenance Objectives, Alternatives, and Constraints
Risk Analysis
Implementation Objectives, Alternatives, and Constraints
Updated Operational Prototyping
Design Objectives, Alternatives, and Constraints
Risk Analysis
Operational Prototyping
System/ Product Objectives, Alternatives, and Constraints
Risk Analysis
Design Assessment Prototyping
Risk Analysis
Demonstration Prototyping
Project Definition
Conceptual Prototyping
Product Review
Design Review
Rqmts Review
System Review
Simulations, Models, and Benchmarks
Concept of Operation
Engineering and Project Planning
System Software Spec.
Software Require- ments Spec, Updated System Software Specification
Detailed Design
Updated Detailed Design
Design and Development Transition Planning
CSCI Integration and Test Site Activation Training Planning
Software Architecture and Preliminary SDDs
Code
Code
Unit Test
Unit Test
Enhanced Operational Capability Integration, Activation and Training Planning
Integration and Test
Integration and Test
IOC DELIVERY
Qualification Testing
Formal Testing
FOC DELIVERY
User Acceptance Test and Training
FCA/PCA
PLAN NEXT PHASE
DEVELOP NEXT LEVEL PRODUCT

Spiral Model II

- Proposed by Boehm in 1988.
- Goal: minimizing risk
- Quadrants:
  - Determine objectives, alternatives, and constraints
  - Evaluate alternatives, identify, and resolve risks
  - Develop, verify next-level product
  - Plan the next phase
- If all risks cannot be mitigated, the project is immediately terminated

Spiral Model III

- Actions before each phase
  - Analyze alternatives
  - Risk analysis
- Actions after each phase
  - Evaluation
  - Planning of the next phase
- Radial dimension: cumulative cost to date
- Angular dimension: progress through development phases

Spiral Model IV

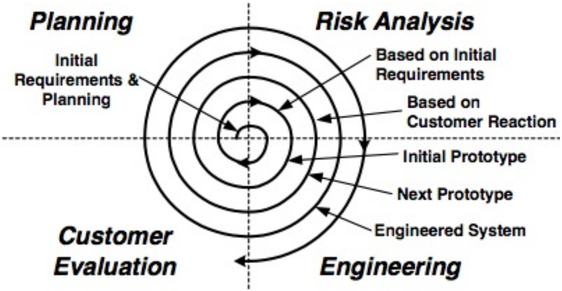
Advantages

- Provides better risk management than other models
- Requirements are better defined
- System is more responsive to user needs

Disadvantages

- More complex and harder to manage
- Usually increases development costs and schedule

Evolutionary or Prototyping Model I



Characteristics

- Develops a product in multiple cycles
- Produces a more refined prototype system at each iteration
- Specification, development and testing occur concurrently
- Design decisions made to get prototype working
- General requirements must be know early

Evolutionary or Prototyping Model II

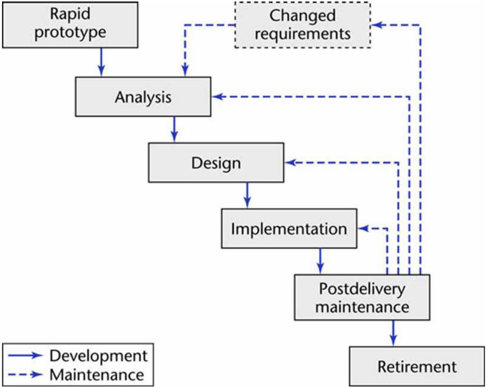
Advantages

- Project can begin without fully understanding requirements
- Final requirements are evolved
- Risk spread over various iterations
- Emphasis on early operational capability

Disadvantages

- Usually increased costs and schedule over waterfall
- Increased management
- Users can mistake a prototype for the final system
- Risk may be increased in various areas

Rapid Prototyping Model I



Rapid prototype

Changed requirements

Analysis

Design

Implementation

Postdelivery maintenance

Retirement

Rapid Prototyping Model II

Characteristics

- Early development of a working rapid prototype
- Interaction with client/users with rapid prototype to validate the product, i.e. determine whether product is what client wanted.

Rapid Prototyping Model III

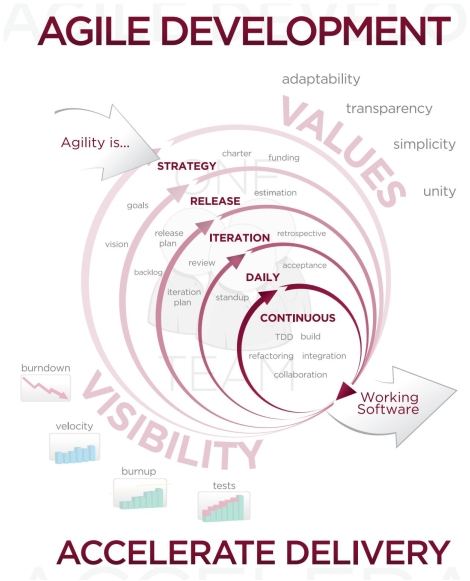
Advantages

- Rapid prototype used to generate the specification document
- Design phase aided by existence of the prototype
- Reduced effect of regression faults in implementation
- Development feedback loops less likely to be needed

Disadvantages

- Insufficient analysis can lead to improper solutions
- User confusion over prototype vs. final system
- Excessive development time of prototype
- Higher start up costs for building a development team

Agile Development Models I

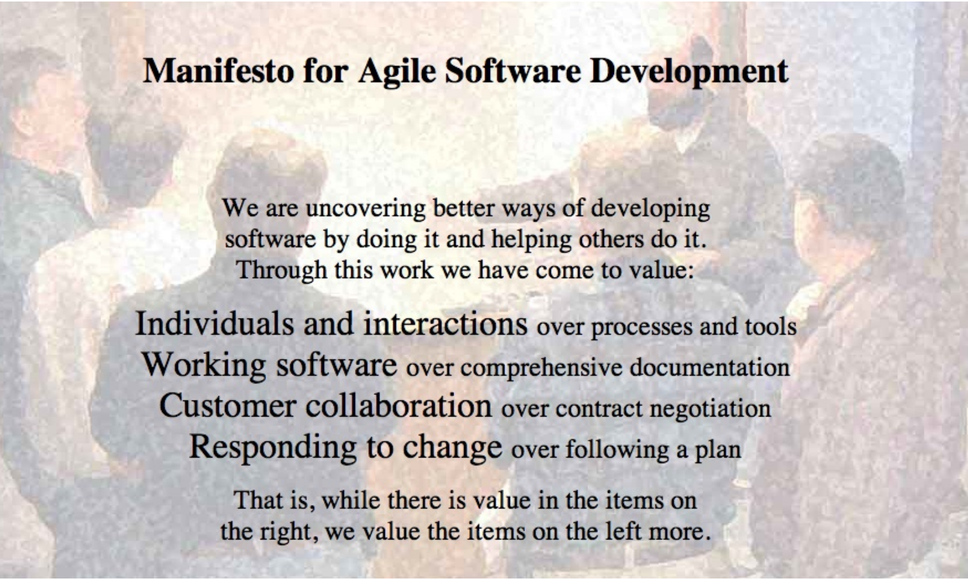


Agile Development Models II

Characteristics

- Based on iterative and incremental development
- Requirements and solutions evolve through self-organizing, cross-functional teams
- Defined in the Agile Manifesto published in 2001

Agile Development Models III



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Agile Development Models IV

Twelve Principles

- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily co-operation between business people and developers
- Face-to-face conversation is the best form of communication (co-location)
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

Extreme Programming I

Characteristics

- Created by Ken Beck in the mid 1990s
- Focuses on features or stories the client wants
- Estimate duration and cost of each story
- Test-driven development: Test cases for a task are drawn up first
- Pair programming: two programmers implement a task, ensuring test cases work correctly
- Task integration: tasks continuously integrated into current version
- The design is modified while the product is built – refactoring

Extreme Programming II

Advantages

- XP successful with small-scale software development
- Responsive to user needs
- Less emphasis on documentation

Disadvantages

- Lack of structure and necessary documentation
- May require major cultural shift to be adopted
- Scope creep – due to lack of detailed requirements
- Non-functional quality attributes are not user stories

SCRUM is an agile development framework, following many of the ideas in XP

Synchronize-and Stabilize Model - Microsoft I

- A version of the iterative and incremental model [Cusumano and Selby, 1997].
- Characteristics:
  - Requirements analysis: interview potential customers, extract key features of interest.
  - Draw up specifications.
  - Divide project into 3 or 4 builds by critical features
  - Each build is developed by small teams working in parallel
  - Synchronize at end of day: integrate partially completed components, test and debug
  - Stabilize at end of the build: maintenance, fix faults. Build is freezed.
- Requirements can be modified during course of a build.