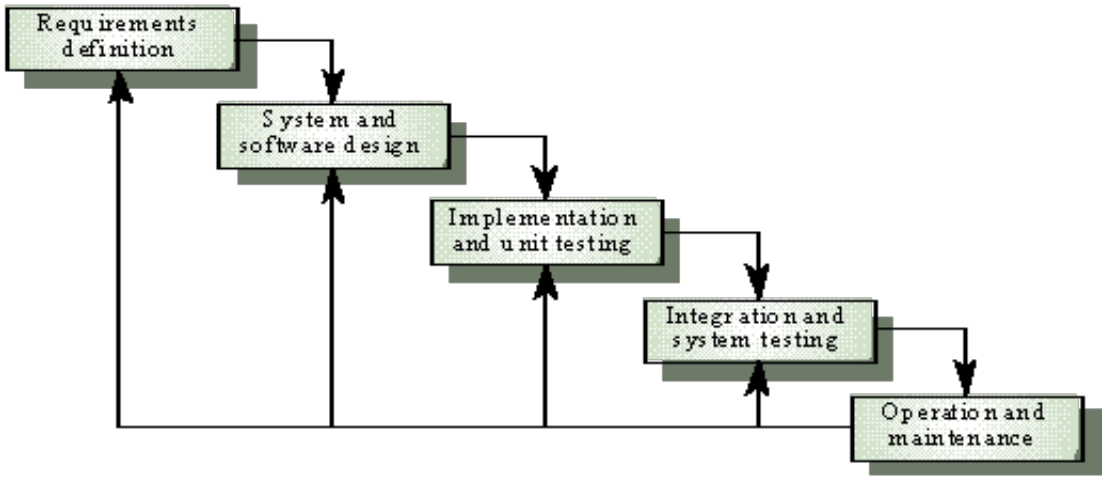
CS 315 - Lecture 2 - Aug 24, 2015

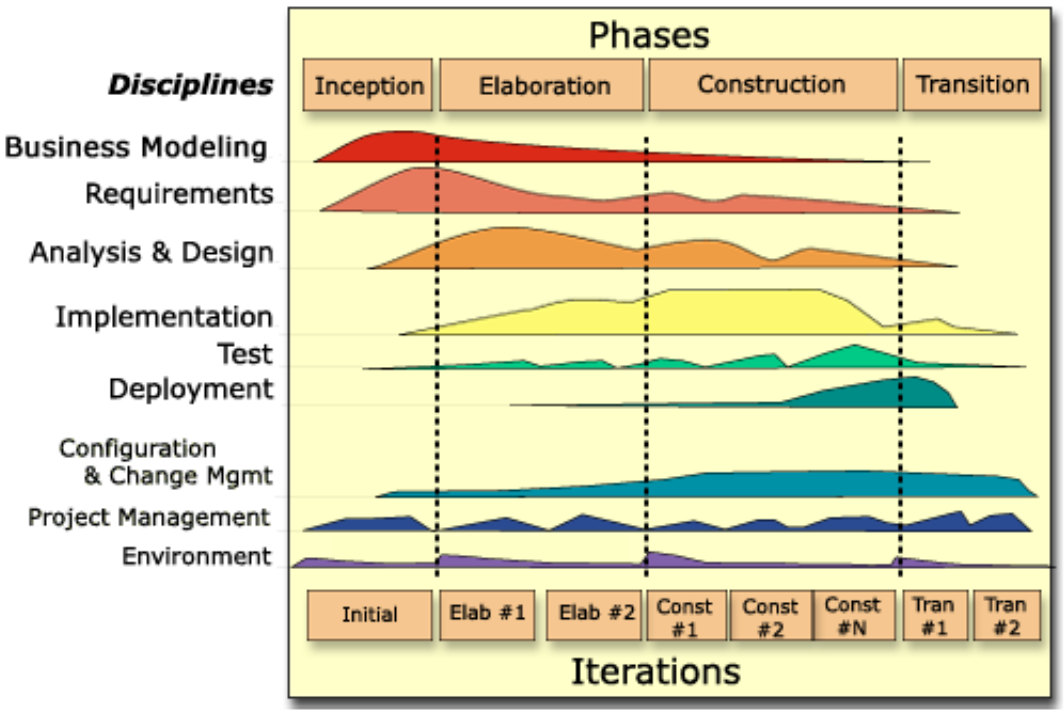
Chapter 2: Software Development Lifecycle Models

- Lecture Notes
- Idealized Model
  - Linear
  - Start from Scratch
  - "Greenfield" situation
  - Got it right the first time

- Waterfall Model
  - Linear with Feedback

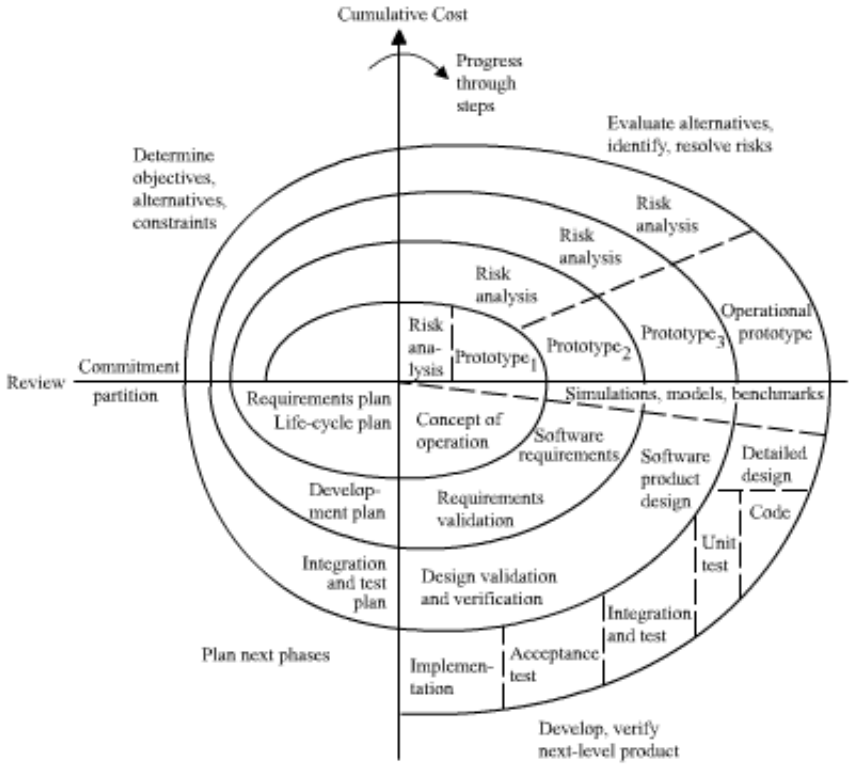


  - Pros
    - Good for contracting
  - Cons
    - Doesn't handle requirements changing well
  - Iterative and Incremental
    - All "flows" are done in each increment but to varying degrees
    - Multiple opportunities to test, receive feedback, and adjust
    - Specific expectations (deliverables) for each increment and each workflow
    - Reduces risk because there is interaction after each step in the process
    - (Rational) Unified Process



-
- Unified Process Site
    - Good resource with a lot of templates

- Evolutionary Development (Build-and-Fix)
    - No design
    - No Specifications
        - Maintenance Nightmare
    - The easiest way to develop software
    - The most expensive way



The Spiral Model

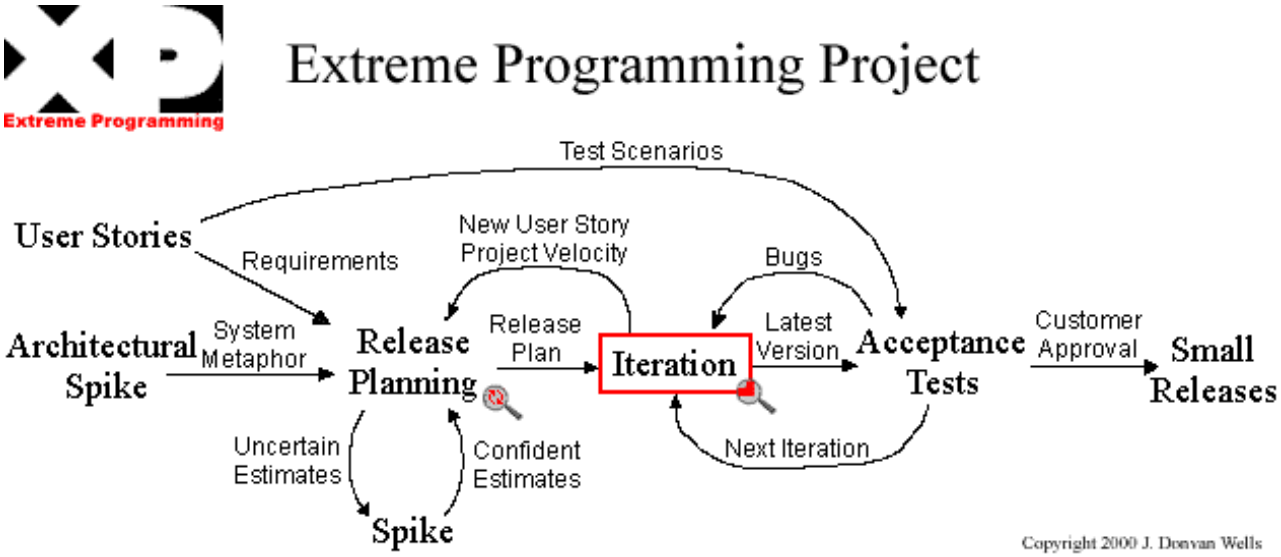
-
- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-sized interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime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

- Rapid Prototyping
  - Prototype to define requirements
  - Explore poorly understood requirements
  - Explore new technology
  - **RAPID**

- Open Source
  - Core group with "Support Group"
  - Quick Releases
  - Lacks documentation and design

- Agile Processes

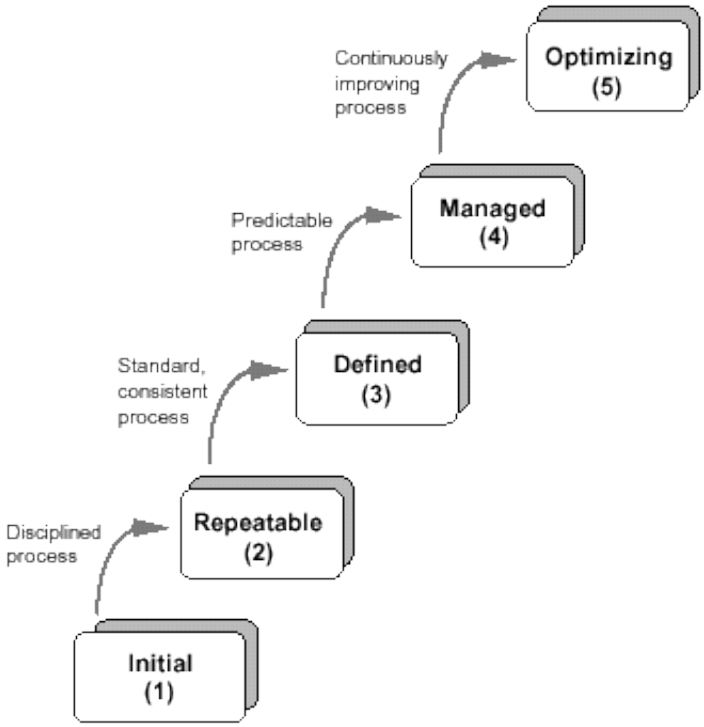
- Driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple 'software increments'
- Adapts as changes occur
- Extreme Programming



  - User Stories are what the user wants
    - Refined into requirements that can be tested

- Scrum
  - All requirements become features in the backlog
  - Features from the backlog are selected for the next 'sprint' (14 days, 30 days, etc.)
  - Daily meeting that covers:
    - What you did yesterday
    - What you are doing today
    - Any issues/blockers
  - New functionality is demonstrated at the end of the sprint
- Other Models
  - Sync-and-Stabilize
    - Microsoft Model
    - Consisted of daily check-outs and check-ins
  - Book contains a chart of various other models on page 67

- Software Process Maturity



- CMM-SW - Capability Maturity Model for Software

- Focus on Quality
  - Begins with the individual
    - Awareness
    - Best Practices
    - Commitment
    - Bottom-up and Top-down
  - Personal Software Process
  - Team Software Process

- The primary goal of any process is **High Quality** software
  - How well does a product meet its specification
  - Problematic
    - Tension between the customer (efficiency, reliability, etc.) and the developer (maintainability, reusability, etc.)
    - Some quality requirements are difficult to specify in an unambiguous way
    - Specifications are usually incomplete