CS301 Software Development

Instructor: Peter Kemper

Software Processes

Software Processes

"Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves."

Alan C. Kay, Source: ACM Queue A Conversation with Alan Kay Vol. 2, No. 9 - Dec/Jan 2004-2005 (from wikiquote)

Photo: wikipedia.org

… well, this is obviously not what software is supposed to be!

What issues stick out?

Software Process

- Most software development projects follow recognized stages from inception to completion
- Useful definitions:

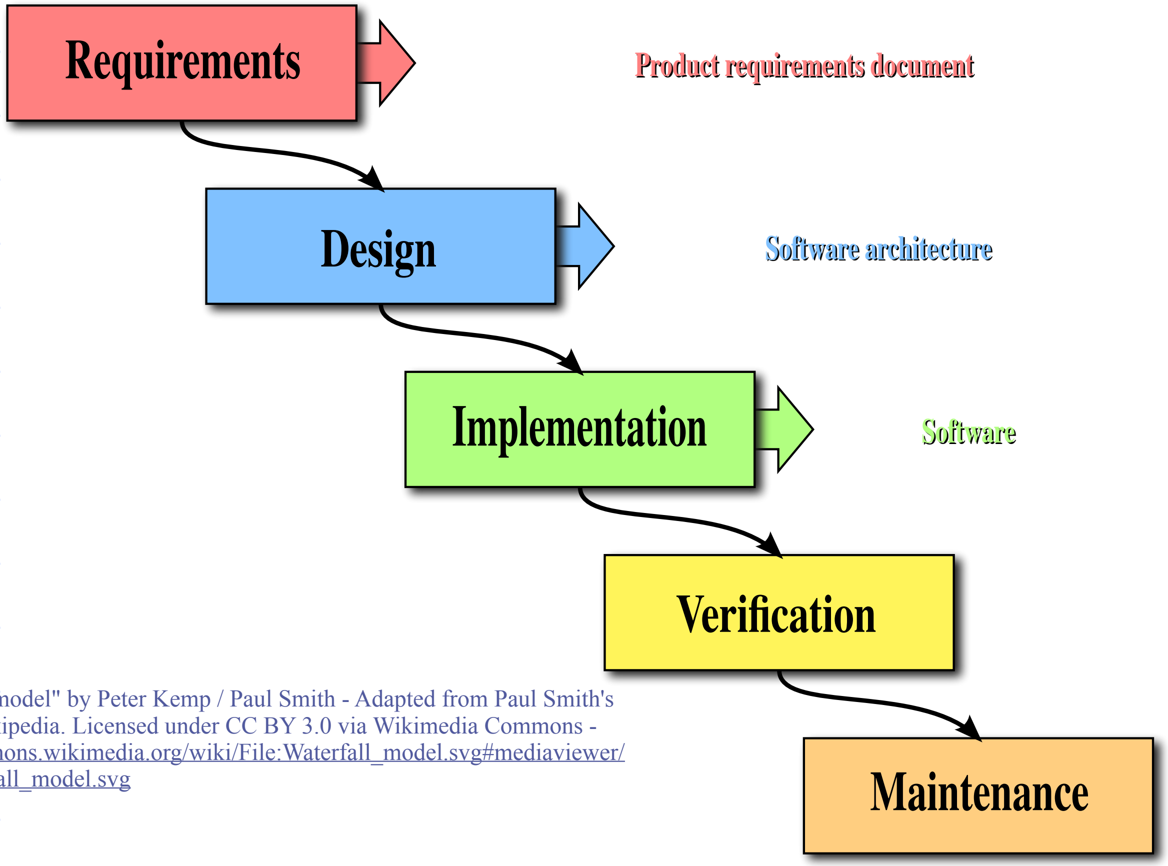
"A software process is a set of related activities that leads to the production of a software product."
"A software process model is a simplified representing of a software process."
— Ian Sommerville, Software Engineering Vol 9.

- Example software process models:
  - Waterfall model
  - Extreme Programming

Three Basic Steps in a SW Project

- Analysis: "figure out what needs to be done"
  - ➡ Product requirements document, functional specification
- Design: "figure out how to do this"
  - ➡ Software architecture, class design
  - ➡ Breaks overall system into a set of interacting components
- Implementation: "just do it (right)"
  - ➡ Implement & test individual components
  - ➡ Integrate & test components into subsystem, overall system
  - ➡ Delivers software product

The Waterfall Model



"Waterfall model" by Peter Kemp / Paul Smith - Adapted from Paul Smith's work at wikipedia. Licensed under CC BY 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Waterfall_model.svg#mediaviewer/File:Waterfall_model.svg

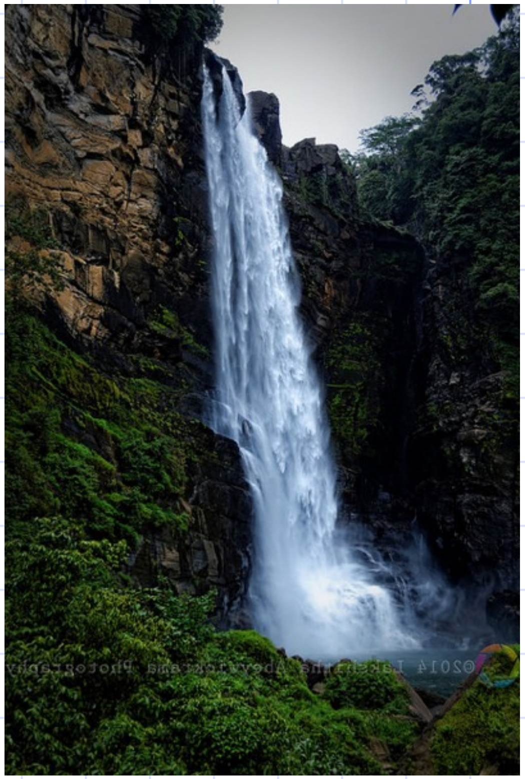


Photo credit: Lakshitha88, cc, flickr.com

- Analysis: gather requirements, write functional spec
- Design: define software architecture, class design
- Implementation: implement components, integrate, test

1. Analysis Phase, Requirements

Gives a Functional Specification

- In the form of a document with:
  - Use cases (textual description, use case diagrams)
  - List of features

Concerned with what needs to be done.

- This should:
  - Completely define tasks to be solved
  - Be free from internal contradictions
  - Be readable by both domain experts and software developers
  - Be reviewable by diverse interested parties
  - Be testable against reality

Feasible?
Preconditions

Example: Word Processing Program

Describes system in terms of:

- fonts, footnotes, sections, toc, etc

Defines user interface

2. Design Phase

Task

- Identify classes
- Identify behavior of classes
- Identify relationships among classes

Delivers artifacts

- Textual description of classes and key methods
- Diagrams of class relationships
- Diagrams of important usage scenarios
- State diagrams for objects with rich state

Iterative process

Why?

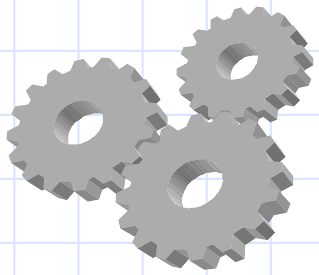
3. Implementation and Testing Phase

What to do

- Implement and test classes
- Integrate classes into packages, test packages
- Integrate packages into program, test program

Experiences

- Testing can be interleaved or separated into subsequent step (verification)
- Last step of verification is user acceptance test
- Avoid "big bang" integration



The Waterfall Model

- You see some variation of the model in literature to emphasize different aspects
  - Requirements vs gather requirements + specification
  - Implementation vs implementation of components + integration
- Observations
  - Top-down design: requirements, spec, design
  - Bottom-up implementation: implement, integrate, integrate, …
- Quality assessment (testing) after each phase
  - Verify requirements with customer
  - Check consistency of functional specification with requirements
  - Check class design with functional specification
  - Test implementation against design, functional specification

The Waterfall Model

- Pros
  - Emphasis on spec, design, testing
  - Emphasis on communication through documents
- Cons
  - Relies heavily on accurate assessment of requirements at start
  - Unprepared for change over time: functionality, environment
  - Little feedback from customers until end
  - Process can take a long time before first working version
    - Problems with integration show up late
    - Mismatch with customer expectations (possibly changed) show up late
  - Sequential
    - Developers involved late, little room for parallelism

Is that it?

Problems seen in real world projects

- project solves the wrong problem, code is unusable
- project not finished on time, critical components not working
- project’s internal components do not fit together
- project relies on key personnel to be developed/maintained

Many ways to proceed developed, proposed, refined, established

=> Models of Software Engineering

- Waterfall
- V-Model
- Spiral model
- Transformation model
- Incremental Model
- Unified Process Model
- …
- Agile Software Development, Test Driven Design