Software Methodologies & Life Cycles

Software “Methodology”

HOW TO design and develop software

Historical Perspective

1967

Historical Perspective

1972

Historical Perspective

1976

Last time

Software design/development is a “wicked problem”
Outline

- Methodologies
- Historical Perspective
- Essential processes of software development
- Life-cycle models
- Methodologies (NEXT TIME)

Essential Processes of Software Development

- Build the software

Essential Processes of Software Development

- What is the software supposed to do?
- How should it do it?
- Build the software
- Does it work?

Processes of Software Development

- Feasibility
- Requirements
- Design
- Implementation
- Test
- Deployment
- Maintenance

Life Cycle Models

A “Software Life-Cycle Model” specifies when the processes are conducted and how they feed into each other.
Waterfall Model

Requirements

Design

Implementation

Test

what is wrong with this picture?

feedback here is essential

Waterfall Model with Backflow

Requirements

Design

Implementation

Test

here too

"Life-Cycle" Models

• Single-Version Models
• Multiple-Version Models

Single version models

• Big Bang
• Waterfall
• Waterfall with backflow
• V model
Big Bang Model

Waterfall Model

Waterfall Model with Backflow

"V" Model

“Life-Cycle” Models
- Single-Version Models
- Incremental/Iterative Models

Iterative vs. Incremental
- Iterative: re-do project in each stage
  more general
- Incremental: add to project in each stage
Iterative Development

- In each iteration:
  - Identify the largest risks of the project
  - Brainstorm on ways to reduce or eliminate these risks
  - Form a concrete plan with specific artifacts
  - Carry out the plan

Risk Driven Design/Development

- Better to find out about infeasible, intractable, or very hard problems early.
- Better to discover flaws early.
- The easy parts will be worthless if the hard parts are impossible.

Iterative/incremental "Life-Cycle" Models

- Sawtooth Model
- Spiral Model & Variants
  - Controlled Iteration Model: Unified Process
  - Time Box Model
- Scrum

Sawtooth Model

(extension of V model)

- Requirements Analysis
- System Design
- Program Design
- Implementation
- Unit Test
- Integration Test
- Acceptance Test

Boehm Spiral Model

Iterates cycles of these project phases:
1. Requirements definition
2. Risk analysis
3. Prototyping
4. Simulate and benchmark
5. Design, implement, and test
6. Plan next cycle (if any)
Controlled-Iteration Model

- Four phases per iteration
  - **Inception**: Negotiate and define product for this iteration
  - **Elaboration**: Design
  - **Construction**: Create fully functional product
  - **Transition**: Deliver product of phase as specified
- The next phase is started before the end of the previous phase (say at 80% point).

Rational Unified Process
(a form of controlled iteration)

- **Business Modeling**
- **Requirements Analysis & Design**
- **Implementation**
- **Test**
- **Deployment**
- **Configuration Management**
- **Environment**

Iterations within phases

Time-Box Requirement

- Requirements analysis
- Initial design
- while( not done )
  
  
  
  Develop a version within a bounded time
  Deliver to customer
  Get feedback
  Plan next version

  
Scrum Model
(incremental model, includes some aspects of team structure, as well as process)

A small group is responsible for picking up the ball and moving it toward the goal.

Scrum, A cure for Wicked?


Argument for the Scrum Model over other iterative models

- A software development project might not be compartmentalizable into nice clean phases as the Spiral models suggest.
- Scrum may be "just the thing" for wicked problems, because the team can quickly react to new information.
Some Principles of Scrum Model

• Always have a product that you can theoretically ship; “done” can be declared at any time.
• Build early, build often.
• Continuously test the product as you build it.
• Assume requirements may change; Have ability to adapt to marketplace changes during development.
• Small teams work in parallel to maximize communication and minimize overhead.

Use of Iteration in Scrum

http://www.controlchaos.com/scrumwp.htm

• Each iteration consists of all of the standard Waterfall phases.
• But each iteration only addresses one set of functionality.
• Overall project deliverable has been partitioned into prioritized subsystems, each with clear interfaces.
• Test the feasibility of subsystems and technology in the initial iterations.
• Further iterations can add resources to the project while ramping up the speed of delivery.
• Underlying development processes are still defined and linear.

When to use single version?

• IMHO: For simple projects!

When not to use single version?

• IMHO: For everything else!

Note: some might disagree, e.g. Budgeon

Why not single-version?

• Initial requirements are speculative

Growth in requirements

Why not single-version?

- Initial requirements are speculative
- Initial designs are speculative
- Speculative decisions compound in absence of feedback

Software is “hard”

- Software is very “hard”
  Discover Magazine, 1999: Software characterized as the most complex “machine” humankind builds.

Which may explain this:
Decrease in Productivity

![Graph showing decrease in productivity with respect to project size in KSLOC.](chart.png)

Based on 1,600 systems.

Why not single-version?

- Initial requirements are speculative
- Initial designs are speculative
- Speculative decisions compound
- High complexity/low adaptability
- High risk issues identified/addressed late in the life cycle