Software Development Overview

CPSC 315 – Programming Studio
Fall 2011

Variety of Software Development Processes

- Traditionally covered in Software Engineering
  - We'll only give a very brief overview of most
- Many are not “clear cut” ideas
  - Often modified to incorporate ideas from other models; seldom used in “pure” form

Waterfall Model of Development

- It is the “traditional” software engineering approach
- Involves series of stages, each a process that converts one product to another
- The development “flows” from the top (early processes) through to the bottom

Waterfall Model

Customer’s Needs
Requirement Engineering
Requirement Specification
Design Specification
Executable Modules
Integrated Software
Delivered Product
New Requirements

Requirement Engineering
Design
Programming
Integration
Delivery
Maintenanc
e

Delivered Product
New Requirements
Waterfall Model

- It can get more complex
  - Feedback from later stages to earlier ones
  - Verification and Validation testing in each stage
  - Or, a separate testing stage after integration
  - Can extend to incorporate iterative approaches

Good Points
- Provides clear structured process, especially useful on large projects
- Clear requirements, design at beginning can make things much easier and better later on
- Tend to have good documentation throughout

Bad Points
- Can be tough to know requirements ahead of time
- Difficult to evaluate how later parts of system will really work in practice
- Requires more discipline by programmers to implement

Iterative Software Development

- Rather than produce a single product “all at once”, provide incremental improvements
  - Deliver pieces of the product at various times
- Time is planned to iterate on the design and implementation of the system
- Includes user analysis, feedback to improve

Iterative Approach

1. Initial Idea
2. Collect Requirements
3. Plan and Design
4. Test and Evaluate
5. Implement
6. Deliverable
Prototyping

- Fits into iterative approach
- Deliver early prototypes of the software
  - Not fully functional, or with poor functionality
- Prototypes should allow one to get user feedback
  - Allows revision of requirements, design
- Possible problems:
  - Can hide difficulties underlying the prototype
  - Can set expectations too high
  - Provides early design anchoring (less flexible)

Spiral Model

- Combines iterative and prototype approaches
- Starting from center, (basic requirements), a prototype is created in the first iteration
- Each successive iterative cycle produces a newer, better prototype (spiraling out)
- When good prototype is found, fix system

Cleanroom Development

- Couple iterative process with very detailed evaluation
- Every iteration gets tested on a very large test data set
  - Provides “hard” statistical data on how reliable the method is
- Measure whether iteration has introduced or reduced defects
  - Introducing defects indicates problem – go back to previous stage and start over

Formal Processes

- Some of these techniques have been collected into more formal descriptions
  - The Rational Unified Process – incorporates much of this, plus more; suite of software products to support the process
- Standards developed for specifying many stages, such as requirements, processes, assessments
Agile Software Methods

- Newer trend in software development
- Meant to contrast vs. “heavyweight” methods of software development
  - Heavyweight – Highly regimented methods, typified by the waterfall model
  - Designed to respond/change quickly, but involves much less long-term planning
- Many methods fall under the “Agile” heading
  - Extreme programming
  - Scrum
  - Plus, it overlaps with some ideas of iterative development

Agile Methods

- Tend to involve lots of collaboration
- Seem to work best with smaller, co-located teams
- Tend to be good for projects where requirements will shift during development
- Will be the focus of the next lecture