Lecture 5: Peer Evals
Software Process Models (cont)
Product Backlogs

Notes
- break project into smaller *mini-projects*
- each mini-project addresses one or more “*risks*”
- risks are interpreted *broadly*
  - poorly understood requirements, architecture, …
  - performance problems
  - missing features
Example from Rapid Application Development

Just an example:

- projects may have more or less iterations
- projects may do different steps in each iteration

Spiral Model

What are the advantages?

- as costs increase (later iterations), risks decrease
- checkpoints/planning each iteration (management control)
- good for high-risk, not well understood projects
- ultimate flexibility … waterfall is like a cruise, spiral like a road trip
What are the disadvantages?

- complicated (especially for managers)
  - need focused, experienced management
  - complicated because of ultimate flexibility
- most projects are not so high-risk …

Staged Delivery (Incremental Development)

Notes

- waterfall-like requirements and design steps
- show (deliver) software in each stage
  - partition system in subsystems in beginning
  - implement one subsystem at a time
  - each stage (increment) is a traditional waterfall approach
Staged Delivery (Incremental Development)

What are the **advantages**?

- feedback on product (visibility) earlier, instead of at end
- similarly, tangible signs of progress (after each increment)
- if important subsystems developed early, can be used

What are the **disadvantages**?

- requires careful planning (# of stages, details of each)
- can be hard to build subsystems independently
- increments must be meaningful to customer (for visibility)
Evolutionary Prototyping (Iterative Development)

- Design and Implement Initial Prototype → Refine Prototype Until Acceptable to User → Complete and Release Prototype → Release

- focus on visible aspects in each iteration
- demonstrate prototype to customer/user after each iteration
- continue to extend prototype based on user feedback
- stop when customer/users agree “good enough”
- complete remaining work and release

Example Scrum-like Iteration

- Requirements Analysis → Design → Code → Test

- One iteration
- After each iteration, you (ideally) have quality software

After initial requirements and rough design …
- each (weekly) iteration becomes a mini-project
- each iteration produces a working system
- get feedback at end of iteration from customers/users
- after each iteration, use feedback to plan tasks for next iteration
Incremental vs Iterative

Assume we want to build a word processor with features for:
- creating text
- organizing text (e.g., cut and paste)
- formatting text (e.g., font size, style, margins)
- etc.

**Incremental Development:**
- define subsystems (e.g., viewing, editing, managing files, …)
- order subsystems: managing files, viewing, editing, …
- 1st increment: files; 2nd: viewing, …

**Iterative Development:**
- come up with a basic design
- 1st iteration: basic features from each “subsystem”
- etc.

What are some *advantages*?
- visibility (after each iteration)
- identify and adapt to changes early
- time and budget constraints (each iteration extends working system)
- good for customers unsure of (or reluctant to commit to) features
Iterative Development

What are some disadvantages?

- can be hard to estimate time/cost at beginning
  - try to gather bulk of requirements first
  - then in iterations, deal with changes
- can lead to “do-stuff-until-we-run-out-of-time-or-money”
- can degrade to code-and-fix or quality issues if not careful

Software Project Success, Agile vs Waterfall ...

The resolution of all software projects from FY2011-2015 within the new CSAM database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

* Standish Group, 2015
Introduction to Sprints

Work on your milestones as a series of iterations

- called “sprints” in scrum
- usually 1-2 weeks (for your project 2 weeks)

Each sprint has

- a goal (what functionality are you tackling?)
- something new to show customer at end of sprint (visibility)
- review with customer/users & plan next sprint

“Product Backlog”: unfinished features / tasks / etc

“Sprint Backlog”: unfinished sprint features / tasks / etc

Requirements

The plan

- “user stories” (agile/scrum technique)
- requirements in general

Requirement Statement (or just requirement)

- a single feature or characteristic of the system
- needed by the customer to make the system useful

Requirement Specification

- a set of requirement statements
- together make up the requirements of the product

… captured through the product backlog in scrum