Tuesday, October 25

Announcements

- Crowdsourcing the Midterm
  - http://www.drsusansim.org/teaching/inf111/pligg
- Homework 5
  - Skip lab portion
  - Use anything you want to draw the diagrams for the take home portion

Software Process Models

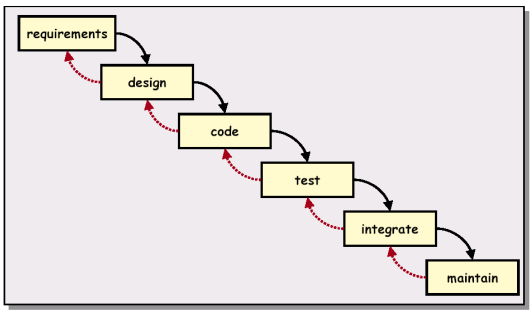
Software Process

- A Software Process Model is a simplified representation of the software process, presented from a specific perspective.
  - General and abstract
- Software Process is a set of activities whose goal is the development or evolution of software
  - Specific and enacted
- Like the difference between class and object/instance

Dimensions of Variation

- Phased or iterative
- Plan-based or incremental
- Continuous testing or late testing
- Feedback
- Risk management

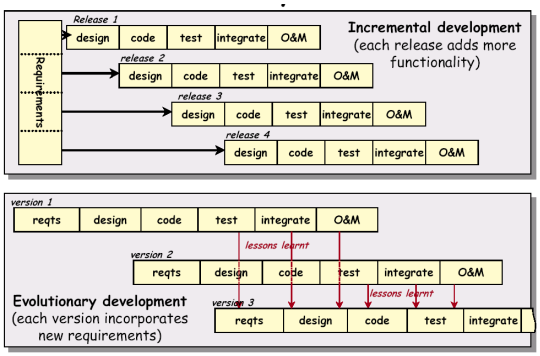
Waterfall Model



Waterfall Model

- Points in Favor
  - Some things must occur before others in the process, it makes sense to have code before test, requirements before design etc.
  - It works as a model because it's essentially true
- Points Against
  - No project or design has ever proceeded this way
  - Very difficult to lock down details before proceeding to next step

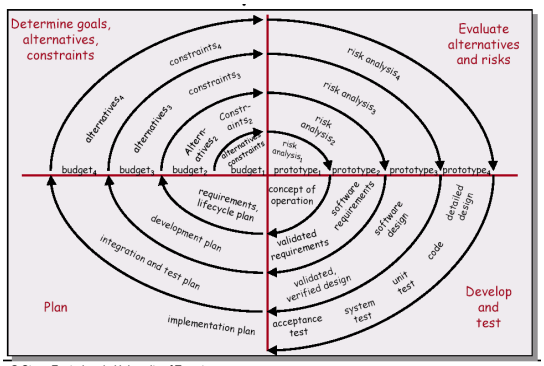
Evolutionary Model



Evolutionary Model

- Points in Favor
  - Accommodates throw-away prototyping
  - Allows for lessons from each version to be incorporated into the next
- Points Against
  - Hard to plan for versions beyond the first
  - Lessons may be learned too late
  - Process is not visible
  - Systems are often poorly structured
  - Special tools and techniques may be required

Spiral Model



Spiral Model

- Points in Favor
  - Incorporates prototyping and risk analysis
- Points Against
  - Cannot cope with unforeseen changes (e.g. new business objectives)
  - Not clear how to analyze risk

Question

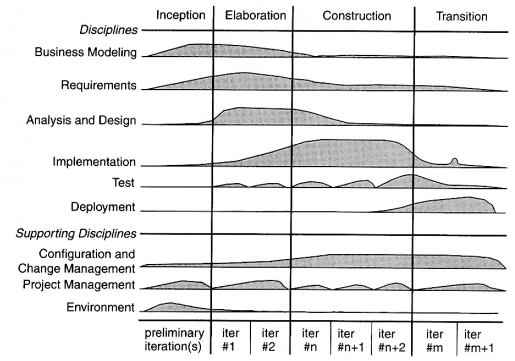
- What is the difference between iterative and incremental?

Thursday, October 28

Rational Unified Process

- RUP is a software (development | design) process that is:
  - Use case-driven
  - Architecture-centric
  - Iterative and incremental

Rational Unified Process



UP Phases

- Inception
  - Approximate vision, business case, scope, vague estimates
- Elaboration
  - Refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates
- Construction
  - Iterative implementation of lower risk and easier elements, preparation for deployment
- Transition
  - Beta tests, deployment

Agile Methods

- Currently, very popular in industry
- Agile means being able to move quickly
  - Mentally quick and resourceful
- Develop software iteratively and incrementally
- Manage risk by managing scope
  - Strong customer focus
- Continuous feedback
  - Between developers, managers, and customers
- Can not plan for all possible changes, instead embrace change

Commonly Used Agile Methods

- SCRUM
  - Emphasis is on managing the project
- Extreme Programming (XP)
  - Guides development and management
- Others
  - Lean
  - Crystal

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Agile is not entirely new

- Iterative and incremental process models have existed for a long time
  - Spiral model by Barry Boehm (1985)
- Evolutionary software development
  - Mentioned by Fred Brooks in “No Silver Bullet” (1987)
- Frequent deliveries and feedback
  - EVO by Tom Gilb (1985)

Current Differences

- Popularity
  - Initiated by software developers
  - Now taken up by executives
- New Techniques and Tools
  - Test-driven development
  - Refactoring
  - User stories
- Growing Community with Shared Terminology
- Infrastructure
  - Coaches, training, certification, courses, conferences

Current Reasons for Popularity

- Effectiveness
- Results
- Compatibility with web applications
- Cool factor

Extreme Programming (XP)

- Four Values
  - Communication, simplicity, feedback, and courage
- The Principles
  - Concrete applications of the principles
    - Rapid feedback; assume simplicity; incremental change; embracing change; quality work
- Four Basic Activities
  - Coding, testing, listening, and designing
- Twelve Practices

Extreme Practices

- If code reviews are good, we'll review all the time (pair programming).
- If testing is good, everybody will test all the time (unit testing).
- If design is good, we'll make it part of everybody's daily business (refactoring).
- If simplicity is good, we'll always leave the system with the simplest design that supports the functionality (the simplest thing that could possibly work).
- If architecture is important, everybody will work defining and refining the architecture all the time (metaphor).
- If integration testing is important, then we'll integrate and test several times a day (continuous integration).
- If short iterations are good, we'll make the iterations really, really short– seconds and minutes and hours, not weeks and months and years (the Planning Game).

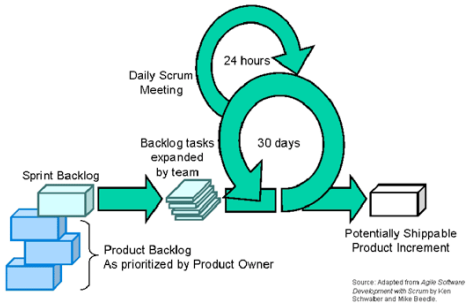
Twelve Key Practices of XP

Programmer Practices	Simple Design Test-driven development Refactoring Pair programming Collective code ownership Continuous integration Coding standards
Management Practices	Planning Game Small releases 40-hour week
Customer Practices	On-site customer Metaphor

Scrum

- Derived from the rugby term “scrum”
  - Despite appearances, is a organized test of strength and skill
- Work is done in sprints (iterations) that form releases
- Key Roles: Scrum Master and Product Owner (On-Site Customer)
- Key Practices: Daily stand-up meeting, time-boxing, and burn-down chart

Scrum Schematic



Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Steps in the Scrum Process

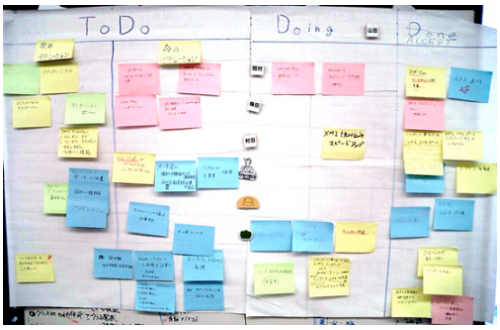
- Product Owner identifies backlog task items (User Stories)
- Releases are planned
  - Scrum team estimates the cost of user stories
  - Product Owner prioritizes user stories
- A sprint begins with a Sprint Planning Meeting
  - User Stories are broken down into tasks with time estimates
  - Anyone doing work is involved, including testers, system administrators, documentation writers
- Every day begins with a daily stand-up meeting
- A sprint ends with a Sprint Review Meeting

Scrum Master

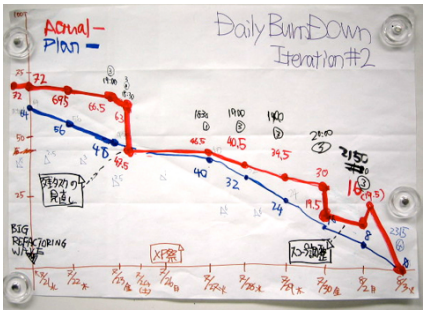
- A project management role
- Chairs the meetings
- Tracks progress
- Maintains burn down chart

- Work allocation is based on historical performance
  - Number of story points per release
  - Number of hours per sprint

Task Board



Burn Down Chart



Misconceptions About Agile

- Myth: Agile is undisciplined
  - Fact: Agile *is* disciplined, but not in a traditional way
- Myth: Agile is not suitable for large teams
  - Fact: Agile can be used on large teams, but requires a more overhead than plan-based approaches
- Myth: Agile is not suitable for geographically-distributed teams
  - Fact: Agile does work in these settings, if you have the discipline, infrastructure, and organizational support.
- Myth: Agile means no documentation
  - Fact: You can use as much or as little documentation as you need on agile.
- Myth: Agile means no architecture
- Myth: Agile means no planning