INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Summer, 2008
Michele Rousseau
Final Review

Announcements

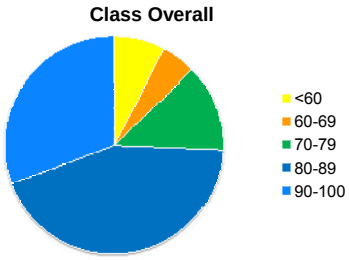
- **Quiz #4 is available if you want it**
- **Please submit re-grades for Quiz #4 by tomorrow.**
- **Thanks for being a great class!**
- **Have a Great Summer!**

Tips for taking your finals

- **Get a good nights rest**
  - I know this is tough, but you don't think as well with a few hours sleep
- **Eat breakfast**
  - Your brain needs protein
  - Try not to eat a high carb breakfast
- **Pace your studying**
  - Try not to put it off until the last minute
- **Study with classmates so you can compare notes**
- **Calm yourself before you come in…**

How are you all doing now?

- **Pretty darn good!**



What will the final exam cover?

Can I tell you exactly what is on the exam? → No
Can I tell you what it will cover? → Yes

Everything… but primarily
- **Lectures**
- **Discussions**
- **Readings**

Format
- **Some problem solving**
- **Some questions similar to the quiz**
- **Likely to have some T/F ---**
- **Maybe multiple choice**

Types of Questions

- **Usually like to know**
  Do you know the topic:
  - What are the tools/methods are available?
  - When do you use them?
  - Why do you need them?
  - What are the Adv/Disadv?
  - Can you apply them?
- **Topics**
  - Tools/Methods
  - Software lifecycle
    - Agile → XP
  - Testing
  - Configuration Management
  - UML
  - Estimation

First Lecture:

- **Why do we need tools?**
  - Automate tasks, Help people to do complex tasks, Improve s/w quality, Increase productivity, Permit verification and conformance checking, Project tracking, Establish procedures
- **Why is there a gap between state of the art tools and practice?**
  - Focus on end product → not the process
  - Learning curve
  - Don't know what is out there
- **What is the difference between a Tool, a Method and a Notation?**
  - Tools → Programs
  - Methods → Processes/Procedures
    - A technique is one activity → a method is a collection
  - Notation → Language used by Methods and Notations

A few questions answered Lecture 1

- **What is a CASE Tool?**
- **What is a workbench?**
- **What is a SEE?**
- **Would you want tight or loose integration of your tools?**
- **What is refactoring?**
- **What is a Model?**
  - An abstract representation → defined by a consistent set of rules

Software Lifecycle Models

- **What is a S/w Lifecycle Model?**
  - An abstract representation of a software process
- **Some Examples:**
  - Waterfall
  - Rapid Prototyping
  - Incremental
  - Spiral Model
  - Agile Method

Traditional

Why do we need them? Why not just build and fix?

Process models

- **Traditional models**
  - Disadv:
    - Reqs and design changes are very costly towards the end the of the process
    - Implementation delayed
- **What is the Agile method good for?**
  - Smaller teams/businesses
  - Adv:
    - Incremental, iterative → adaptive
- **4 central values**
  - Focus on human role of s/w dev
  - Continuously produce tested working s/w
  - Foster the relationship with the client
  - Development group → developers & customer reps
    - EX → XP

XP

A set of key practices that suggest a process

- **Key Concepts**
  - Embrace Change
  - Defer Costs
- **6 Phases of development**
  - Exploration
  - Planning
  - Iterations to Release
  - Productionizing
  - Maintenance
  - Death

XP

- **14 Key Practices**
  - Programmer Practices
    - Simple Design, test-driven development, refactoring, collective code ownership, pair programming, coding standards, just rules
  - Management Practices
    - Planning Game, small releases, 40-hour work week, open workspace
  - Customer Practices
    - On-site, metaphor

What is pair programming?

Problems with XP

- **Corp Culture must support XP**
- **Not well suited for large projects**
- **Some assumptions:**
  - Developers are great (know tools and methods)
  - Customer's are knowledgeable
  - Good communication skills
  - customers can "hang out"
  - It is feasible to collocate

Brooks - NSB

"There is *no single development,* in either technology or management technique, which by itself *promises even one order-of-magnitude improvement within a decade* in productivity, in reliability, in simplicity"

Essence (specs, design, testing)
vs. Accident (programming/Compiling)

- **Software is inherently difficult because of:**
  - Complexity, Conformity, Changeability,Invisibility
- **Brooks suggests**
  - If it already exists → use it
  - Requirements refinement → rapid prototyping & many iterations with the client
  - Grow – Don't build – software → develop incrementally
  - Train great designers

Testing

- **Verification → are we building it right?**
- **Validation → are we building the right product?**
- **Failure → incorrect/unexpected output**
- **Fault → invalid execution state**
- **Error → "bug"**

Can you have a fault without a failure?
Can you have a bug without a fault?
Why do we care about latent faults?

- **Black Box Testing →Functional Testing**
- **Whitebox Testing → Structural Testing**
- **Different levels of testing**
  - System, Integration, Unit, Regression

Testing

- **Static Analysis → examine and analyze source code**
  - Code Reviews, Inspections

Advantages / Disadvantages?
What are you looking for?

- **Fundamental Testing questions**
  - Test Criteria → What should we test
  - Test Oracle → is the test correct?
  - Test Adequacy → How much is enough?
  - Test Process → is our testing effective?

Why not test exhaustively?
What are the goals of testing?
Adv/Disadv of formal techniques.
What is correctness?

Testing – Test adequacy

- **Coverage-based→ a sufficient % of the program structure has been exercised**
  - Eg. Control flow graphs (all-statements, all-branches, all-edges, all paths)
- **Fault-based →Failure/test curve flattens out**
  - Eg. Error seeding
- **Error-based → faults found in common are representative of the total population of faults**
  - Equivalence partitioning

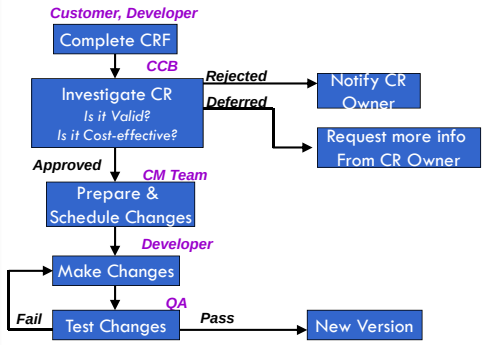
UML

- **Know how the diagrams should be used and what they describe**
- **Do they show Dynamic or static behavior?**
- **How do you use them?**
- **Know how to create them**
- **What are the differences between them?**
- **-Lots of this was covered on the quiz**

Configuration Management

- **What is it? → manages change (software and artifacts)**
- **What types of artifacts should you track?**
- **What is a CCB?**
- **The CM Team controls →**
  - Costs
  - Effort
  - Maintains all changes & documents
- **What is the basic process?**

CM: The Change Process



CM

- **What types of tools does CM use?**
  - Change tracking tools
    - Track the status of a CR. Lock/unlock modules
  - Form Editor
  - Workflow system
  - Change Database
  - Change reporting system
- **What types of workbenches?**
  - Open → tools in each stage are integrated
  - Integrated → provide whole process integrated support
- **How do we identify versions?**
  - Version numbering
  - Attribute-based
  - Change-oriented

What are the Adv/Disadv?

Effort Estimation

Predicting the resources required for a software development process

- Effort, calendar time, total cost, scheduling

- **MMM**
  - Why do we have estimation problems?
    - Poor estimation → overly optimistic
    - Confuse effort with progress
    - Don't back our estimates
    - Schedule progress is poorly monitored
    - Adding people to a late project makes it later

Why aren't men and months interchangeable?

Poor Estimation Techniques

- **Guessing**
- **Parkinson's Law → work fills the time available**
- **Pricing to win → project costs what you need it to in order to get the contract**
- **Budget Method → Costs what the customer has to spend**
- **Brooks → Gutless Estimating**
  - Like P'law → schedule to meet client's date

What are the adv/disadv

Better Estimation Techniques

Based on experience and hard data.

- **Expert Judgement**
- **Estimation by analogy**
- **Delphi Method → team of experts**
- **Algorithmic cost modeling**
  - Usually based on SLOC (or DSI)
  - COCOMO
- **Personal software process**

When do you use them
Adv/Disadv.

COCOMO: Three Models

- **3 Models reflect the** complexity**:**
  - the Basic Model
  - the Intermediate Model
  - and the Detailed Model
- **What are the differences in these models?**
- **When would you use them?**
- **What are cost drivers?**

The Development Modes: Project Characteristics

- Organic Mode
  - developed in a **familiar**, stable environment,
  - **similar** to the previously developed projects
  - relatively **small** and requires little innovation
  - Eg. Payroll system
- Semidetached Mode
  - **intermediate** between Organic and Embedded
  - Eg. Banking System
- Embedded Mode
  - tight, **inflexible** constraints and interface requirements
  - The product requires **great innovation**
  - Eg. Nuclear power plant system