ECS 160
Software Engineering

Instructor: Prem Devanbu
TA: Frank Maker
Grader: Sunny Dhillon
ssdhillon@ucdavis.edu

http://www.cs.ucdavis.edu/~devanbu/teaching/160

January 5, 2010

Introduction

Software problems: Y2K.

- How much did this cost?
- Why did this happen?
- Who were the players? Who was guilty?
- How could it have been prevented? What would we do differently?

More problems: Ariane blow-up; Therac 25; AT&T Switching network collapse; Aiplane crashes; security vulnerabilities

Some surprising facts:

- Productivity/Cost (LOC/day, "Man Month" etc).
- Organizational differences. Which one is right for you?
- Coding is the easy part!
- people differences.
- Quality (defect rates, cost of fixing, best way to find defects)

What this course is about

Not algorithms, languages, operating systems or databases-you already studied that in other classes.

Not business/finance issues—you have or will study that elsewhere

So what is it? *building software faster, better cheaper* using teams of people.

It is concerned with using technologies: *processes, models, and tools* to make teams produce better code, in less time, with less money.

You will learn about techniques for figuring out what to build, how to build it, and evaluating whether you've done what the customer wanted.

The Bad News

You should have taken *ECS 140A*, (and other pre-requisites).

You should really understand data abstractions, static typing, object-oriented programming, etc. C++ (or Java) knowledge is very important. The material presented in this course assume this knowledge.

You will have to complete a *significant* project in this course.

We will help you (a lot) but essentially you'll have to learn a substantial software infrastructure by reading documentation.

the work product is *not just a program*, but a set of documents describing *what* the system does and *how* it was built, and *how* it works.

Your work will be judged by industrial standards:thoroughly tested, bugs documented, etc.

Motto: *"Learn through experience"*

The Good News
About learning through experience

This course will help you get up to speed quickly, and survive on your first job. Specially with: Requirements, Design, Inspection, Testing, etc.

As part of the course, you will produce a system complete with documents and a substantial implementation.

If you excel at this course, you can show your work to your employers, *who will be impressed.*

You will learn about a powerful, current infrastructure for mobile applications.

You will get more done in one quarter (in terms of implementation) than you may have thought possible.

You will become a much more object-oriented, understand software process, architecture, design and coding secrets. Good for interviews.

Logistics

Website will be active shortly. Information on Google group available shortly.

(Posting policy is linked from class home page)

You will get fastest answers from google groups, not email to the TA/grader/me

Course Home Page:
`http://www.cs.ucdavis.edu/~devanbu/teaching/160`

All details (office hours, TA's, assignments, answers, readings, hints, etc etc) will be posted there.

Watch the web page and the google group frequently! We will assume everyone is reading it!

Project Details

Form teams of 4 people—by thursday–email the TA, Frank. Teams finalized friday. Team members will be asked to sign a pledge that they will not drop the class.

By next Monday noon, email project idea to Prem, Frank, and Sunny. (..some ideas...)

Follow a defined process: project assignments are intermediate work products.

Assignments due noon on specified date. 20% credit demerit for each day delay; no exceptions.

You are allowed 1 week after grades are returned for any assignment or quiz request a regrade. No Exceptions.

Tested on our machines!

More about Project

- Requirements will be designed in discussion with Customer (us.) We will ask for n some extensions somewhere around the middle of the quarter. (to be realistic)

- You will find out late about new languages (um, e.g., Lithuanian, Macedonian) that you have to customize your application for. You'll have one day to make the change. *i18n!!*

- "Big Bang" due date, with checking of intermediate milestones. You will:
  - Be awarded credit for steady progress through the course...
  - Penalized less for defects if you are meeting deadlines.

- We will use Google Android API.

- Downside: steep learning curve
- Upside: useful, current technology.
- Real world experience: Application specific platforms are $de\ regeur$.
- Current technology: pub-sub, mobile GUIs, energy issues.

Teams–1

Form Teams of 4.

Report Team membership to TA (flmaker@ucdavis.edu) by thursday, Jan 7th, Noon. If you haven't, we will assign you as we please! Feel free to use the google group to find team mates.

Either way, *you* will have to make it work. Welcome to the real world.

Each team member will be held responsible for demonstrating good working knowledge of all aspects of the product.

I am ready and willing to help with team process. Bring problems to me, and I am willing to facilitate discussions. No negative effect on grade (and perhaps even a positive one) if brought to my attention *in the first 4-5 weeks*.

Project Assignments

Assignment 0 Come up with a brief summary of project (monday, Jan 11, noon) and met with Prem sometime during Tuesday Jan 12th.

Milestone 0 Demonstrate basic familiarity with Android API with a simple Twitter App (About Jan 15th)

Assignment 1 Come up with a set of test scripts for final project.

Milestone 1 Basic, simple functionality demonstrated (TBD)

Assignment 2 Mock up of GUI + class diagram of system

Milestone 2 (roughly) Atleast 50 % of test cases will pass.

Milestone 3 (roughly) Atleast 70 % of test cases will pass.

Final Demos & Interactive grading Give a professional-quality demo to V.C's *Devanbu & Associates*; plus grading with T.A. Plus full documentation.

Teams–2

Team interactions are an important part of this course.

The design of the system should reflect and support team work allocation. Think about:

- shared Data structures
- functional decomposition
- Uniformity of design
- separation of concerns

Teams–3

Every team member should contribute equally to all assignments.

As far as dividing up work—some ideas:

- *Feature-Based Partioning* Members consider different types of users.

- *Switching off on roles* Two members design GUI/Web Pages for one part, the other two progam that part, and then switch around. Likewise testing/ development. *But all team members should be familar with all aspects!*

- *Functional Decomposition* Decide on a set of functional units (subroutines, higher-level code, etc) and divide up the work.

This is not make work, this is real world.

Bad Boys and Girls

(All true)

"He didn't/wouldn't do anything!"

"He didn't return my phone calls/emails"

"She never called me or emailed me!"

"He is completely clueless".

"He changed the code at the last minute and ruined 10 weeks of work!"

"I don't know the system because my team mates would not show me the source code!".

"He copied code from another team and gave it to us!".

Other Details

About UC Davis Software Engineering group:

Zhendong Su, Ron Olsson, Prem Devanbu, and about 12 graduate students– we're among the top 10 most successful groups in the world. Work in defect detection, concurrency, open-source empirical studies, testing.

About me...

My interests: open source, middleware, security, tools.

Industrial Experience: Perkin-Elmer (2 years), AT&T/Lucent 17 years, work consultant for HP Labs, Microsoft Research, Banks, Startups, Lawyers, etc.

I (still) have more experience in Industry software engineering than at teaching: could use good feedback. Anytime. Send email–anonymously, if you wish.

Project description and grading (credit) details will be posted by monday. watch the web page.