CIS 422/522, Winter 2011

More than the sum of its parts

Not just programming in the large
  But scaling up presents important technical and non-technical challenges, and is essential.

Not just generic management
  But management principles and techniques are also essential

What is software engineering, and why should you study it?

Engineering = design in context and under constraint
  Context includes non-technical as well as technical aspects, e.g., teams of developers
  Constraints include budget, schedule, and externalities (e.g., compatibility with foobar)

Design of everything, from product to process to business model

Design, Design, Design

Design of the product as seen by the user
  • Requirements analysis, feasibility, user interface design, ...

Design of a structure for growing the product (or products)
  • Architectural design

Design of a project plan, a process, an organization, documentation, ...

And code – that’s design too
A small taste of a big field

You can’t really learn much of software engineering in a term
Goal: Learn a tiny bit, and be prepared to learn more on your own
Strategy: Focus on what you can make use of now; discuss how it may differ from what you’ll need in other contexts

My goals

You should know a few things
• How to make a schedule and meet it. How to break a big project into pieces, and which pieces to build first. How to survive a teammate who flakes or gets hit by a bus. How to argue about design alternatives. How to devise abstractions, and why.

And you should think like a designer
• Creativity plus discipline, and a habit of learning from everything you build and many things you don’t

Not in my goals

I don’t care if you know UML
— (but you might want to, for communication)

I don’t care what design patterns you know
— (but you should be learning many patterns, all the time, at many levels)

I don’t care if you know XP or RUP or Scrum
— (but you should be able to judge their suitability to a given project and organization)

Components of Course W11

2 project phases (~4 weeks each)
1st project same for all teams
Select today
2nd project can be second iteration, or team choice

Teamwork
I choose teams, but you can indicate preferences
Two chances to execute “trades”

Lectures (some), presentations (enough), discussion (lots), 2 exams
Grading (approximate)

15% phase 1 deliverables
35% final deliverables
  adjusted by group member evaluations & observed contribution
5% presentations
10% contribution to discussion
  includes constructive critique of presentations
15% midterm
20% final exam

Team Project Grading

One base score for whole team
  No excuses: You can’t succeed if your team fails
Group member (peer) evaluations
  Effective score may be adjusted up or down from base score
Trouble? Don’t wait to tell me!

Project Characteristics

Address an actual need
STRICT project schedules; it is your job to scope your project appropriately
Modest technical challenges
  partly set by you
High requirements for completeness and quality

Project Deliverables

Installable package (reasonably portable)
  For users and for developers / maintainers
  Which implies: test harnesses & test suites, internal & external documentation, ...
All documentation
  User manual, technical documentation
  Web site
Read the grading guide!
Project Deliverables (prototype)

Assessment of questions related to risk
  specific information goals
  includes: running program, test harnesses & test suites

Documentation for developers
  Analysis: Not just what, but why
  Reusable code, design, etc

Meetings and Presentations

Brief classroom presentation most weeks
  (starting Tuesday of next week)
Final presentations in dead week
  • Public presentation during class hours. Invite your friends and loved ones.

Reading

No textbook
  Instead: papers
  • Some old research papers, some fairly new
  • Other materials to fill in the gaps and give some flavor
    of the “scholarly” side of SE

Entry Questionnaires

Fill out, turn in.
Pictures

Start shooting while we discuss project 1

Some goals and constraints

Goal: Have fun
   – A reasonable goal, even for commercial software, but seldom the main goal. Fun makes people creative and focused.

Goal: An accomplishment to show off
   – Something to show a prospective employer, or to write about in a grad school application. Implies: Impressive to your target audience.

Goal: Learn something valuable
   – Technical and non-technical (“soft”) skills you will use later

Constraint: 4 + 4 weeks
   – Hard deadlines, and you must balance work load with other classes and life.

Projects

Choosing a project is a design problem
   Identify goals and constraints
   Generate alternatives (mostly done, not entirely)
   Evaluate and decide

What are your goals and constraints?

The Candidates

Room 100 name board:
   Work with Room 100 Users Group to design and evaluate

Cue sheet generator:
   Reverse geocode routes (esp. for cycling)

Lecture record prep:
   Easy merge of slides with lecture video