CSE 210 – Principles of Software Engineering

Bill Griswold
Computer Science & Engineering
UC San Diego
wgg@cs.ucsd.edu
Introductions

Research in SE, CS Education

“Summer Fellow” at AppFolio this summer
Building Great Software is Hard

2/3 of projects are late [Tata]

1/4 of all projects are cancelled [Standish]

1/2 run over budget [Tata, SGR CACM]

Allstate insurance planned a 5-year, $8M project. Six years later they replanned for $100M.

Why the disasters? Scale.

Users want more and more features

“…reworking a software requirements problem once the software is in operation typically costs 50 to 200 times what it would take to rework the problem in the requirements stage... A 1-sentence requirement can expand into...500 lines of code...and a few dozen test cases.”
Change/Evolution yields Complexity/Bugs

Figure 4 Serial and average growth trends of a particular attribute

Figure 7 Complexity growth during the interval prior to each release

Belady & Lehman, A Model of Large Program Development, IBM Systems Journal, (15)3, 1976
Wait a minute. You mean we need to be able to quickly steer a large ship?
S.E. Practices Like Agile Make a Difference

Results of Raytheon’s use of best-practices.

SE practices are rooted in process-centric quality control.

SOURCE: Raytheon

<table>
<thead>
<tr>
<th></th>
<th>Rework Cost (% of total)</th>
<th>Productivity Increase (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>1988</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>1989</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>1990</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>1991</td>
<td>15</td>
<td>75</td>
</tr>
<tr>
<td>1992</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>1993</td>
<td>5</td>
<td>125</td>
</tr>
<tr>
<td>1994</td>
<td>2</td>
<td>150</td>
</tr>
</tbody>
</table>

SAVINGS OF $17.2 MILLION
Quality control in early manufacturing was **Product-Centric** ("what")
- Regularly test *product* outputs
- Make adjustments to factory as needed
- But what to fix?

Mid-20th c., shift to **Process-Centric** ("how")
- Still test *product* outputs
- Also measure *process* elements
  - plans, people, tools, product-in-progress
- Use **cause-and-effect model** to adjust factory as needed
- Statistics to precisely track variation
- Buzzword: *Statistical Process Control*

- SE has inherited this legacy
- SE methods are **process-centric**
What’s a Software Process?

It’s the “how” that produces the “what” – quality software
- **What**: what customer wants, on time, under budget, free of flaws

A prescribed sequence of steps

Steps include:
- Planning
- Execution
- Measurement
  - Product, and process itself
  - Examples: bugs, progress, time, feature acceptance by cust.

A software process is a self-aware algorithm
- Observes and adapts according to measurements

Agile processes are adaptive to the “customer”
- Features, schedule, budget, priorities, markets, change
- Must measure these as well as internal elements (correctness)
- Easily extended to adapting to many other “problems”
  - ...as long as they can be observed and measured
The Changing Face of Software

Applications
- Web 2.0, Mobile 2.0, ...
- Ubiquitous computing
- Developing world
- Big data, AI, ....

Methodologies
- Open Source
- Agile (XP, Scrum)

Technologies
- Web services, javascript, AJAX, JQuery, ...
- Programming environments (Eclipse), AOP
- Component-based, Model-driven software development

Do we rewrite the rules, or just reinterpret them?
Technical Themes of the Course

Scale
All of computer science, especially CS research, is about managing scale. So is SE.

Risk
SE is all about managing risk. Doing something important requires taking risks. SE seeks to increase upside risk (great products), while decreasing downside risks (late, buggy, etc.)
Goals of the Course

• Learn *foundational concepts* of SE
• Exposure to the foundational literature
• Improve reading papers critically
• Improve discussing technical ideas
• Take ideas and skills into your own practice
• Ultimately, *software engineering literacy*
  - Conversant in issues – think and talk like a software engineer(ing researcher)
OUR CONTRACT
My Promise

Authentic practice

A minimum of busy work
Your Promise

Come prepared every day

Eager to learn
Rest of Today

- Structure of course
- Grading
- How to read and discuss papers
- Project
- Questions (at any time)
First Week: Intro to Agile Process

You come from many backgrounds

Seen different variants of software process

I’m going to introduce a generic Agile Process

Will be point of contrast for much of course

Also will be used in project

Looks like a lot of reading, but actually not many words

- Don’t skip the side bars and pictures!
- Great examples, great exercises, Q&A
- Don’t have to do the crossword puzzles
Preparation for Class
In-Class Discussion

- “Socratic Circles” round-table discussion
- More dynamic, less controlled, more open-ended
- “Peer learning”
Alternative Formats Throughout

“Workshop”
- working or problem-solving in small groups

Lecture (talk) – me or visitor

No class on Tuesday before Thanksgiving (week 9 – work on your project, read book)
First Couple In-Class Discussions Will be Train-Up

- If enough 218 students, will have you observe example group
- Perhaps a few directed by me. Something like:
  1. I will ask a question (see next slide)
  2. I will select one of you to answer
  3. An assessment of the answer will be made
  4. I may open the question for discussion with neighbors
  5. I will select someone to elaborate the first answer
     [iterate as necessary]
In-Class Discussion – First Couple Discussions

Questions will come from a few places:

- Paper-reading rubric
  - people problem, technical problem…
- One of your questions from the paper
- One of my questions

(The more and better your questions are, the fewer of my questions I’ll ask. :)
Take-Aways

Course is an introduction to:
- SE Concepts, Principles, and Skills
- How to critically read and discuss the literature

Readings and discussion start Tuesday

Start forming a group now (Piazza can help)

Start reading *The Mythical Man Month*
- Read a chapter a week or so to avoid crush in week 10
- Will help you on the project and other readings