Software Projects – the challenges we face
Your pitch and LCO does not need to include a business case analysis. Focus on the technical end, and feasibility wrt technical challenges/time/resources.

Another grading comment …

Turn in assignments using “turnin” (turnin –c cse403 –p lco file1.doc file2.ppt)

About Wed/Thurs/Fri:
- All presentations, submitted Tue night, will be loaded onto a laptop by us and you will be able to run them from that laptop. LCO’s will also be posed on the web for you to peruse.
- Everyone needs to take part in the presentations -- not just one member per team.
- Teams should plan on a 5 minute presentation – rehearse!
- The order of presentations will be announced on Wed in class.
- As you listen to the presentations, be thinking about whether you would like to work on the project. Why, why not? Project preferences will be due Sat 11:59pm (survey link on web).
Readings

- "Rapid Development", Steve McConnell
  - Chapter 3 – 3.3 Classic Mistakes
Outline

- What is a software project?
- What makes a project successful?
- What’s our track record?
- Is software different?
- Do we now have all (any) of the answers?
What is a software project?

- Projects are a balance of three dimensions, with the goal of producing a successful deliverable.
What is essential to a project?

Look at the prior question from another angle

- Risk
- Investment
- Positive payoff

Take a risk and make an investment in order to get a positive payoff
What makes a project successful?

- On time delivery
- Feature set complete
- Reliable
- Performant
- Meets expectations
- On budget
- Team survives without burnout (!)
- **Ability to evolve** *(maintainable, enhanceable)*

What are your ideas? What would make your 403 project successful?
So...what’s our track record?

What would you guess is the percentage of software projects that fail (i.e., that don’t accomplish their goals)?

- 0 – 20%
- 20-40%
- 40-60%
- 60-80%
- 80-100%
And the answer is …

- Historically, nearly **85%** of software projects fail
Details from the Standish report

Why do you think this is the case??

CSE 403, Spring 2006
Chief reasons for software project failures: UGrad answers

- Insufficient planning
- Too “rosy” assumptions
- Poor communication
- Changes to the requirements
- Changes in the context (funding, priorities)
- Doing something without a clear customer base
- Competition
- Entrepreneurial nature of software
Chief reasons for software project failures: Grad answers

- Cost overruns
- Changing of requirements
- Misunderstanding of requirements
- Poor understanding of goals
- Over-ambitious goals
- Lack of clear specification
- Poor planning/research
- Lack of a reasonable & structured software/feature plan
- No commercial market for end product
- Complexity of software

There are a lot of ways that things can go wrong!
Chief reasons/Classic Mistakes from RD

Table 3-1. Summary of Classic Mistakes

<table>
<thead>
<tr>
<th>People-Related Mistakes</th>
<th>Process-Related Mistakes</th>
<th>Product-Related Mistakes</th>
<th>Technology-Related Mistakes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Weak personnel</td>
<td>15. Insufficient risk management</td>
<td>29. Feature creep</td>
<td>34. Overestimated savings from new tools or methods</td>
</tr>
<tr>
<td>5. Adding people to a late project</td>
<td>18. Abandonment of planning under pressure</td>
<td>32. Research-oriented development</td>
<td></td>
</tr>
<tr>
<td>7. Friction between developers and customers' unrealistic expectations</td>
<td>20. Shortchanged upstream activities</td>
<td>34.</td>
<td></td>
</tr>
<tr>
<td>10. Lack of stakeholder buy-in</td>
<td>23. Insufficient management controls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Lack of user input</td>
<td>24. Premature or overly frequent convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Politics placed over substance</td>
<td>25. Omitting necessary tasks from estimates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. Wishful thinking</td>
<td>26. Planning to catch up later</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>27. Code-like-hell programming</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chief reasons for software project failures: professionals’ answers

The majority of software projects fail...

- *not* because of technical deficiencies or problems
- but because of *underestimating the human aspects of development*, including:
  - the relationship with the customers
  - regular and explicit communication between all stakeholders – managers, developers, testers, marketing, sales, customers
54% of features were rarely or never used! That effort could have been productively spent elsewhere!
Is Software Different?
(from Other Engineering Disciplines)

Breakout session!

Rows 1&2: Arguments in favor

Rows 3&4: Arguments against

Rows 5&6: Arguments in favor

Hand in your notes so we can capture in the slides later!
Sp06 403 – Arguments in favor

- Low overhead of actual implementation
- New profession
- Highly technical
- Highly market driven
- Shorter release cycles
- Lower material investment
- Software is customer centric
- Life cycles are longer due to maintenance
- Way cooler!
- Don’t require large groups
- Technology changes faster
Past arguments in favor

- Testing the quality of software is harder
  - So many cases, so many paths, ...
  - Unlike bridges and buildings where everything can be tested using known procedures
- Much higher rate of failure
  - May also have to do with the immaturity of the discipline
- Customers have a greater role
- Frantic rate of technological change
- Software is easier to copy
Sp06 403 – Arguments against

- Same type of development cycles (spec, dev, release, etc)
- Still trying to generate a product
- Still process of “creating” from ground up
- Start with customer demand, require feedback
- Always breaking new ground
- Still need to meet deadlines and requirements (performance, reliability)
- Work in groups
Past arguments against

- Software developers still need to plan, execute, test, and sell their products
Is Software Different?
(from Other Engineering Disciplines)

More questions to consider:

- Is software less reliable?
- Does it break differently?
- Is the environment of use of software different?
- Is the culture of software development different?
- and more…

Do these differences justify/explain our dismal success rate?
Have we learned from our mistakes?

- We’re starting to!
- Driving forces behind the evolution of software development
  - Software becomes a business and a profession
    - No longer just a hobby
    - Standards
  - Best practices get distilled over time
    - Lifecycle processes
    - Designing for change, for test, for leanness, …
  - Productivity tools appear that aid developers
  - Economic and societal trends play an increasingly important role