Software Projects – the challenges we face
What is a software project?

- Projects are a balance of three dimensions, with the goal of producing a successful deliverable.
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)
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 …

2003 data shows nearly **66%** of software projects fail

Why do you think this is the case?

---

CSE 403, Spring 2007, Alverson

2003 Standish Report
## Classic Mistakes from McConnell

<table>
<thead>
<tr>
<th>People</th>
<th>Process</th>
<th>Product</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Undermined motivation</td>
<td>• Overly optimistic schedules</td>
<td>• Requirements gold-plating</td>
<td>• Silver-bullet syndrome</td>
</tr>
<tr>
<td>• Weak personnel</td>
<td>• Insufficient risk management</td>
<td>• Feature creep</td>
<td>• Overestimated savings from new tools or</td>
</tr>
<tr>
<td>• Uncontrolled problem employees</td>
<td>• Contractor failure</td>
<td>• Developer gold-plating</td>
<td>methods</td>
</tr>
<tr>
<td>• Heroics</td>
<td>• Insufficient planning</td>
<td>• Push-me, pull-me negotiation</td>
<td>• Switching tools in the middle of a project</td>
</tr>
<tr>
<td>• Adding people to a late software project</td>
<td>• Abandonment of planning under pressure</td>
<td>• Research-oriented development</td>
<td>• Lack of automated source-code control</td>
</tr>
<tr>
<td>• Noisy, crowded offices</td>
<td>• Wasted time during the &quot;fuzzy front end&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Friction between developers and customers</td>
<td>• Shortchanged upstream activities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Unrealistic expectations</td>
<td>• Inadequate design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Lack of effective project sponsorship</td>
<td>• Shortchanged quality assurance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Lack of stakeholder buy-in</td>
<td>• Insufficient management controls</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Lack of user input</td>
<td>• Premature or overly frequent convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Politics placed over substance</td>
<td>• Omitting necessary tasks from estimates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Wishful thinking</td>
<td>• Planning to catch up later</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Code-like-hell</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSE 403, Spring 2007, Alverson
Classic Mistakes from NASA

- Letting team members work in an unsystematic way (#20,..)
- Setting unreasonable goals (#8, #14)
- Implementing change without assessing impact
- Gold plating (#28, #30)
- Overstaffing, especially early in the project
- Assuming a schedule slip will be made up later (#26)
- Relaxing standards to cut costs/time (#22)
- Assuming that a large amount of docs ensures success
Have we learned from our mistakes?

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

Classic Mistakes

• Rapid Development, 3.3, Steve McConnell
  http://stevemcconnell.com/rdenum.htm
• Survival Guide: 254-257, Steve McConnell