



CSE 403 Lecture 7

How to fail at delivering software



Writing assignment

- Due Monday, October 21
- Generate a random integer x in the range $[1, 15]$
- Analyze to what extent chapter x of Mythical Man-Month is relevant in 2002
- Expected length, 3 pages



Lecture goals

- Identify common problems which lead to software projects failing
- Understand risk management techniques



It's not just software projects that fail

- Tacoma Narrows Bridge
- The Kingdome
- UW EECS Building



Software project failures

- Software projects have a reputation for failure
 - Probably well deserved
 - Many examples of massive cost over runs, release delays and cancellations



Project Failure

- Not delivering working program on targeted date
 - Overrun on time/budget
 - Under delivery of functionality or quality



All to common case

- Project starts out fine, with a few minor changes in requirements, delays of supporting activities and changes in personnel
- Coding proceeds at a good rate with most modules almost working at the point when the system is to be integrated



Then everything goes wrong

- Integration reveals incompatibility between components
- Integration reveals severe bugs in components
- Unexpected hardware or software change
- And a few random disasters
 - Source code lost, key people directed to other tasks, sudden changes in requirements or schedule



What happens next

- Devs code like hell
 - Fixing and patching bugs
 - Significant changes in architecture or functionality on-the fly
- Test and documentation held up
 - "The build is broken – I can't do anything"
- Long hours
 - Negative team dynamics
 - Damage control activities



Day of reckoning

- Substandard product shipped
 - "It's just version 1.0 – we can issue an upgrade"
- Schedule shifts
- Project cancelled or downgraded



Classic Mistakes

- McConnell, *Rapid Development*
 - People related mistakes
 - Process related mistakes
 - Product related mistakes
 - Technology related mistakes




People issues (high level)

- Personnel management
 - Functioning team
- Relationship with customer
- Management issues
 - Management support and competence




People related mistakes

- Motivation
- Weak personnel
- Problem employees
- Heroics
- Adding people to a late project
- Crowded offices
- Friction between dev and customers
- Unrealistic expectations
- Lack of sponsorship
- Lack of stakeholder buy-in
- Lack of user input
- Politics over substance
- Wishful thinking



Process issues (high level)

- Accurate planning
 - Realistic scheduling
 - Contingency planning
- Paying attention to all stages of product development




Process related mistakes

- Optimistic schedules
- Insufficient risk management
- Contractor failure
- Insufficient planning
- Abandonment of planning under pressure
- Wasted time in "fuzzy front end"
- Shortchanged upstream activities
- Inadequate design
- Shortchanged QA
- Insufficient management controls
- Premature convergence
- Omitting necessary tasks from estimates
- Planning to catch up later
- Code-like-hell programming




Product related mistakes

- Requirements gold-plating
- Feature creep
- Developer gold-plating
- Push-me, pull-me negotiation
 - Adding new tasks when schedule slips
- Research-oriented development



Technology related mistakes

- Sliver-bullet syndrome
- Overestimating savings from new tools or methods
- Switching tools in the middle of a project
- Lack of automated source-code control



Questions

- To what extent are these problems specific to software projects?
- Are there characteristics of software projects that make them more likely to occur?
- Why do people make the same dumb mistakes over and over again?