Student Startup Sequence

- Verify network connection
- Rotate to Landscape mode
- Start Presenter 2.0
- Maximize Application
- Role->Student
- Connect->Classroom 1
- Test student submissions
- Have you used a tablet PC before?
- Select All
- Send Selection

Lecture 03: The Fate of Software Projects

Valentin Razmov

Outline

- The Fate of Software Projects
- Is Software Different?
- Lessons from the History of Software Development

References

- "Professional Software Development", Steve McConnell
- "Rapid Prototyping", Steve McConnell

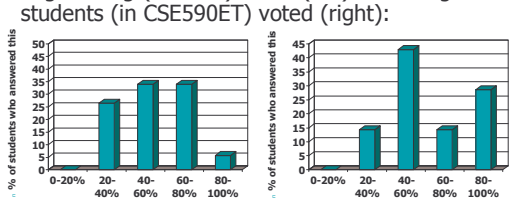
The Fate of Software Projects in Industry: Question

- Under some reasonable definition of a "project" (you make it up), what would you guess is the percentage of software projects that fail (i.e., that don't accomplish their goals)?

  Choose the range in which your estimate falls:
  - 0-20%
  - 20-40%
  - 40-60%
  - 60-80%
  - 80-100%

The Fate of Software Projects in Industry: Answers

- Here is how undergraduate students in software engineering (CSE403) voted (left) vs. how graduate students (in CSE590ET) voted (right):



- Historically, nearly **85%** of software projects fail.

Chief Reasons for Software Project Failures: Question

- What might be the main reasons behind such a large percentage of software project failures?

  State one reason that you think is prevalent.

Chief Reasons for Software Project Failures: Student Answers

- CSE403 students in the past said:
  - Insufficient planning: poor risk analysis, lack of knowledge, lack of motivation, poor decomposition, etc.
  - Too "rosy" assumptions (about future technology, scheduling, etc.)
  - Poor communication
  - Changes to the requirements
  - Changes in the context (funding, priorities)
  - Doing something without a clear customer base
  - Competition
  - Entrepreneurial nature of software (unlike other engineering disciplines)

Chief Reasons for Software Project Failures: Student Answers

- Graduate students (in CSE590ET) stated:
  - Cost overruns
  - Changing of requirements
  - Misunderstanding of requirements
  - Poor understanding of goals
  - Over-ambitious goals
  - Lack of clear specification
  - Original goals were unrealistic
  - Poor planning/research
  - Lack of planning
  - Lack of a reasonable & structured software/feature plan
  - No commercial market for end product
  - Complexity of software

Chief Reasons for Software Project Failures: What Professionals Say

- According to most professionals, the majority of software projects fail…
  - *not* because of technical deficiencies or problems
  - but because of underestimating or sometimes even completely **ignoring the human aspect**, including:
    - the relationship with the customers
    - regular and explicit communication between all stakeholders – managers, developers, testers, marketing, sales, customers
  - Examples:
    - Building a product that no one wants to buy
    - Sabotaging a product (for "political" reasons) that otherwise may have succeeded

Is Software Different? (from Other Engineering Disciplines)

Arguments in favor:	**Arguments against:**

Is Software Different? (from Other Engineering Disciplines)

Arguments in favor:

- Testing the quality of software is harder
  - The Halting Problem presents a fundamental limitation in the extent to which software quality can be evaluated
    - Most properties of software (that we care about) are unverifiable
  - 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

Is Software Different?
(from Other Engineering Disciplines)

Arguments against:

- Popular perception that software is "soft"
  - … that requirements can change, "because change can be *easily* accommodated"
  - In reality, even though change is *possible* in principle, accommodating change is very difficult
    - Often forces a rewriting of the software
- Software developers still need to plan, execute, test, and sell their products

Is Software Different?
(from Other Engineering Disciplines)

More questions to consider:

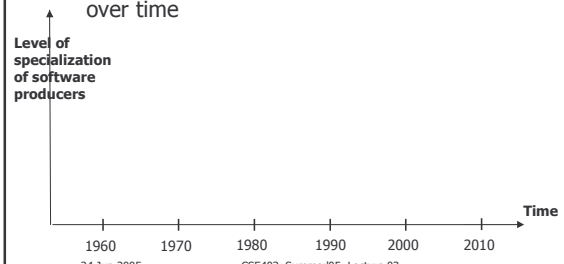
- Is software less reliable?
- Does it break differently?
- Is the environment of use of software different?
- Is the culture of software development different?
- and more…

Lessons from the History of Software Development

- The software 'Gold Rush' fever periods
  - High-risk, potentially high pay-off
  - Typical environment: two guys in a garage
  - Code-and-fix development in hopes of striking it rich by being first-to-market in an unclaimed segment
- The in-between periods
  - Lower-risk, likely lower but stable pay-off
  - Typical environment: larger teams, formal processes
  - Careful, quality-driven development with an emphasis on reliability, interoperability, usability
  - Very different customer base

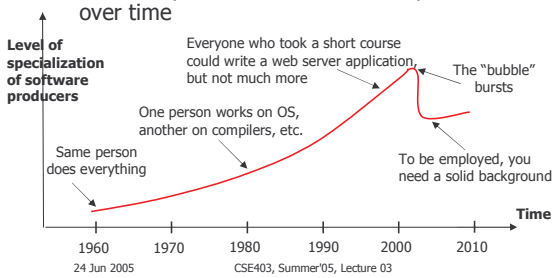
Lessons from the History of Software Development

- Level of specialization of software producers over time



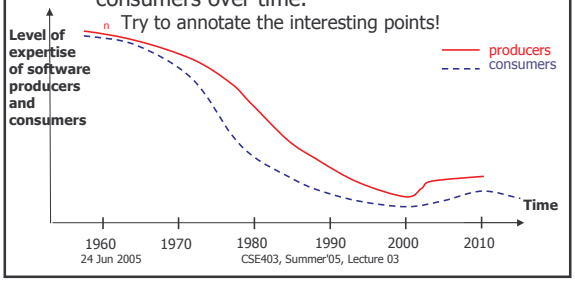
Lessons from the History of Software Development

- Level of specialization of software producers over time



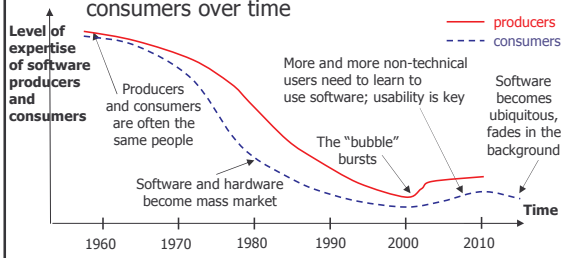
Lessons from the History of Software Development

- Level of expertise of software producers and consumers over time.
  - Try to annotate the interesting points!



Lessons from the History of Software Development

- Level of expertise of software producers and consumers over time



Lessons from the History of Software Development

- Driving forces behind the evolution of software development
  - Software becomes a business and a profession
    - No longer just a hobby
  - Best practices get distilled over time
  - Productivity tools appear that aid developers
  - Economic and societal trends play an increasingly important role

One-minute Feedback

- What one or two ideas discussed today captured your attention and thinking the most?
- List any ideas / concepts that you would like to hear more about. Be specific.