Classroom Interaction via Tablet PCs and Student Submissions

- Inking enables free-form answers beyond what a keyboard allows.
- Everyone who has a tablet device with Classroom Presenter can be engaged simultaneously in class.
- All students take active part in their learning, rather than passively listen and watch.
- Have you used a Tablet PC before? If so, what for?

Student Submission

Concerns You Expressed About the Course

Q: How can we address / mitigate each of them?

- Time pressures
  - Shorter summer quarter
  - Other commitments
  - Picking a project that is too ambitious
- Working well with others
  - Doing one's part of the work
  - Ability to contribute effectively to team
- Team members' roles
- Grading of projects

Lecture 05: Lessons from the History of Software Development

Valentin Razmov

Outline

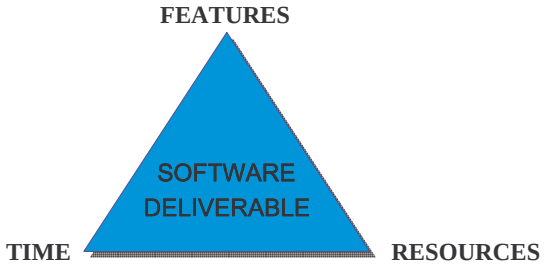
- Software Projects and Constraints on Them
- The Fate of Software Projects
- Is Software Different?

References

- "Rapid Development", Steve McConnell
  - Ch. 3.3
- "Professional Software Development", Steve McConnell

What Is a Software Project?

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



The Goal of Building Software

- A successful deliverable is characterized by being:
  - on time
  - on budget
  - with good quality
- "We do three types of jobs here... Good, Fast and Cheap. You may choose any two!"
- Which two would you pick for a project like yours?
  a) Good and Fast, but not Cheap
  b) Good and Cheap, but not Fast
  c) Fast and Cheap, but not Good

Student Submission

The Fate of Software Projects in Industry: Question

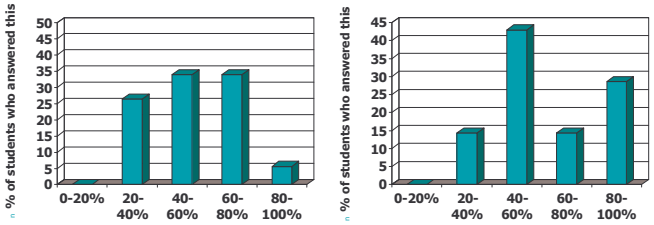
- Under some reasonable definition of a "project" (you decide on it), what would you guess is the percentage of software projects that fail to accomplish their goals?

  Choose the range in which your estimate falls:
  a) 0-20%
  b) 20-40%
  c) 40-60%
  d) 60-80%
  e) 80-100%

Student Submission

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.

Student Submission

Chief Reasons for Software Project Failures: Student Answers

- CSE403 students in the past have 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
  - Not understanding the requirements
  - Changes in the context (funding, priorities, technology)
  - 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) have stated:
  - Changing of requirements
  - Misunderstanding of requirements
  - Poor understanding of goals
  - Overly ambitious goals
  - Lack of a clear specification
  - Lack of a reasonable & structured software/feature plan
  - No commercial market for end product
  - Cost overruns
  - Complexity of software

Chief Reasons for Software Project Failures: What Professionals Say

- The majority of software projects fail…
  - *not* because of technical deficiencies or problems
  - but because of **underestimating the human aspect of development**, 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:

Student Submission

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
- Lower barrier to entry
- Customers have a greater role
- Customer expectations: for quality, delivery timeline, etc.
- Frantic rate of technological change
- Software is easier to copy

Is Software Different? (from Other Engineering Disciplines)

Arguments against:

- Software isn't "soft".
  - Contrary to popular perception, change cannot be "*easily* accommodated"
    - Yet requirements do change.
  - In reality, even though change is *possible* in principle, accommodating change often forces a rewriting of major parts of the software.
- Software developers still need to plan, execute, test, and sell their products. Same lifecycle.
- The discipline is still in its infancy.