Software Engineering

Özgür YILMAZEL
Introduction

• Why do we need a course on Software Engineering?
Problems

• FAA/Advanced Automation System
  – FAA declared that 1.5 Billion spent out of 2.6 Billion was useless
  – "We royally screwed up AAS, no doubt about it, in any way that a project could be screwed up," says FAA Associate Administrator for Acquisitions Steve Zaidman.

Problems

• Ariane 5 Flight 501
  – Ariane 5 reused code from Ariane 4
  – Exceptions were handled by shutting down the software
  – Cost $370 million USD
Problems

• Therac-25
  – Radiation therapy machine
    • Two beams (5 MeV vs 25 MeV)
    • Accidentally activated high-power beam without the spreader plate in place
    • Software didn’t recognize plate was in place
    • 3 out of 6 patients died
    • Software flaw was a race condition
Read more...

- http://spectrum.ieee.org/sep05/1685/
What is Software

Wikipedia:

a collection of computer programs, procedures and documentation that perform some task on a computer system.
American Engineers’ Council for Professional Development defines Engineering as follows:

The **creative application** of **scientific principles** to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an **intended function, economics of operation** and **safety to life and property**.
Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.

- The term software engineer is used very liberally in the corporate world.
  Software Engineering ≠ Programming

You can read more @ http://en.wikipedia.org/wiki/Software_Engineering
Hardware – Software Cost Ratio
Class Survey

• How many of you had any commercial experience?
• Was your project a success?
Class Survey (ii)

– For failed projects, what went wrong?
  • Ill-defined Requirements?
  • Ill-managed Requirements Changes?
  • Poor software development methodology?
  • Unrealistic schedule?
  • Poor Project Management?
  • Lack of user involvement?
  • Lack of stakeholder involvement?
  • Lack of qualified people?
  • Lack of Tools?
  • Corporate politics get in the way?
  • Poor or no architecture?
  • Lack of measurements and controls?
Current State of Software Projects

• Most Software Projects Fail
• One definition for Failure:
  – Missed schedule
  – Missed functionality
  – Missed budget
Current State of Software Projects II

• In 1995, only 16% of software projects were expected to finish on time and on budget.(1)
• Projects completed by the largest US organizations have only 42% of originally proposed functions.(1)
• An estimated 53% of projects will cost nearly 190% of their original estimates.(1)
• In large companies, only 9% of projects will be completed on time and on budget.(1)

(1) Standish Group International Report, “Chaos”, as reported in March ‘95 Open Computing. Copyright 1995 SPC.
Current State of Software Projects III

• Canceled projects—$81 billion loss to US in 1995(1)
• Average MIS—1 year late, 100% over budget(2)

(1) Standish Group International Report, “Chaos”, as reported in March ‘95 Open Computing. Copyright 1995 SPC.
So how can this be called Engineering?

• Why do we allow this to happen?
• Would you like your next <item> built like this?
  – Item ➔ car, home, airplane, bridge, ICU monitoring s/w
• Engineering and construction firms don’t build things in the way most software gets built…
• Why is software allowed to so often be done without any engineering?
• Can we do anything about this?
• Are there some better ways?
The Core Triad for Success

- People
- Processes
- Technology
Engineering Evolution

• Civil Engineering, for example, has been around for centuries

• Task: Build a House
  – Easy to describe features and to know when “Done”
  – Design discipline is well-understood by the architect and Civil engineers
  – Other stakeholders have “bought-in”
  – All are able to read engineering & mgt. artifacts (e.g., blueprints, stress diagrams, Gantt charts)
  – Progress easy to measure, obvious milestones/goal
  – Project can be tackled by forming teams of skilled subs
Engineering Evolution (ii)

• People:
  – Architects, Mechanical/Structural Engineers, Civil Engineers, Project Managers, etc.
  – Specialists: welders, concrete, site layout, etc.

• Process
  – Well-worn steps to achieve writing a proposal
  – Tried-and-true (profitable) management techniques

• Technology
  – Improved design and analysis tools
  – Improved materials
  – Automated management tool support
S/W Solution Evolution

– People
  • Position Descriptions of the 70s:
    – Programmers
  • 80s
    – Programmer/analysts
  • 90s
    – Developer
    – Architect
    – Business Analyst
  • 00s?
    – UI Designer
    – Deployer
    – Assembler
    – Bean Developer
Software Evolution (ii)

- Processes
  - Ad Hoc
  - Waterfall
  - Spiral
  - Incremental
  - Iterative
  - RAD/JAD
  - Unified Process
  - XP
  - Agile
Software Evolution (iii)

• Technology
  – Languages: Assembler, Procedural, Structured, Object-Oriented
  – Life Cycle Tools
    • Requirements, Architecting, Building, Testing
    • Configuration Management/Version Control
    • Two-way, always-in-sync tools
    • Roundtrip Engineering
  – UML
Some Fundamental Issues

• Software is very complex today
  – Hard for one to understand it all
  – Difficult to express in terms all stakeholders understand

• Business drivers add pressure
  – Shrinking business cycle
  – Competition increasing
  – Ever rising user expectations

• Flexibility and resilience to change is key
  – Ability to adapt to sudden market changes
  – Design is solid enough that change does not impact the core design in a destabilizing way
Fundamental Issues (ii)

• Most projects are unpredictable
  – Lack of knowing where and what to measure
  – Lack of yardsticks to gauge progress
  – Requirements creep is common
  – Scrap and rework is common

• Lack of quality people results in failure
  – …in spite of best processes and tools!
  – Managing people is difficult

• Major change is organizationally difficult
Putting Engineering into Software Development

• Applications need to be architected and designed, not just “built”

• Hardware/construction engineering uses
  – Processes
  – Tools
  – Standards
  – Highly Skilled People
  – Employs System Integration concepts
  – Things aren’t just “built” with no planning
Putting Engineering into S/W (ii)

• Software engineering should be no different than hardware engineering
• Projects that fail are numerous and the reasons too varied to delve into
• Projects that succeed follow typical patterns
  – Employ a process
  – Use standards
  – Do up-front architecting/design
  – Perform continuous integration, use metrics & QA
  – Work with real, tangible artifacts (running code)
  – Have solid team support and communication
Putting Engineering into S/W(iii)

- Successful Project Patterns (cont’d)
  - Use tools to automate tasks
  - Are able to reuse items
  - Have quality people
    • Management
    • Development
    • Specialists
  - Have good scheduling (realistic, accurate)
  - Frequent, positive involvement with stakeholders
  - Progress reporting
  - Requirements well understood
My belief

• Software Engineering is different than other engineering disciplines.
• Only thing that won’t change in a software project is that “Requirements will always change” (unlike a construction project)
• We need to:

  Keep software soft!
This Semester

• Learn processes, tools and techniques to build software in a controllable and repeatable environment.
  – Software Engineering Best Practices

• Apply these tools and techniques in the class project and gain life long habits.