Course Information

Course Topics

• Software process
• Requirement analysis
• Software design
  – Architecture styles
  – Design patterns
• Unified Modeling Language
• Software testing
• Software maintenance
• SE research topics
Grading Scale

<table>
<thead>
<tr>
<th>Score</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>97-100</td>
<td>A+</td>
</tr>
<tr>
<td>93-96.9</td>
<td>A</td>
</tr>
<tr>
<td>90-92.9</td>
<td>A-</td>
</tr>
<tr>
<td>87-89.9</td>
<td>B+</td>
</tr>
<tr>
<td>83-86.9</td>
<td>B</td>
</tr>
<tr>
<td>80-82.9</td>
<td>B-</td>
</tr>
<tr>
<td>77-79.9</td>
<td>C+</td>
</tr>
<tr>
<td>73-76.9</td>
<td>C</td>
</tr>
<tr>
<td>70-72.9</td>
<td>C-</td>
</tr>
<tr>
<td>65-69.9</td>
<td>D</td>
</tr>
<tr>
<td>&lt;65</td>
<td>F</td>
</tr>
</tbody>
</table>

• I may choose to curve the grades at the end of the term

Group Project

• Work in teams (3-4 people)
• One project
  – Choose from a set of given topics
  – Come up with your own and get the instructor’s approval
• Go through analysis and design
• Turn in required documents and diagrams
• Give a presentation
• Peer review inside/between groups
Introduction to Software Engineering

Overview

- Software in our lives
- Hardware vs. Software
- What is software engineering?
- Software engineering - precis of a short history by [Barry Boehm, ICSE’06 Keynote]
- Software myths
- Learning objectives
Software is ubiquitous

• System software
  – OS, compilers, device drivers
• Business software
  – Payroll, accounting
• Engineering/scientific software
  – Computer-aided design, simulation
• Embedded software
  – GPS navigation, Flight control, Toaster

Software is ubiquitous

• Product-line software (PC-like based)
  – Spreadsheets, word processing, games
• Web-based software
  – Gmail, Facebook, Youtube
• Artificial intelligence software
  – Robotics, artificial neural networks, theorem proving
What is Software?

• Definition [Pressman]
  – The product that software professionals build and then support over the long term

• Software encompasses:
  – Executable programs
  – Data associated with these programs
  – Documents: user requirements, design documents, user/programmer guides

Hardware vs. Software

☐ Manufactured
☐ Wear out
☐ Built using components
☐ Relatively simple

☐ Developed/ engineered
☐ Deteriorate
☐ Custom built
☐ Complex
Manufacturing vs. Development

- Hardware is difficult or impossible to modify
- Software is routinely modified and upgraded
- Hiring more people causes more work done
- This is not always true
- Costs are more concentrated on products
- Costs are more concentrated on design

Hardware does “wear out”

Failure curve of hardware—“bathtub curve”
Software does “deteriorate”

Component based vs. Custom built

- Hardware products employ many standardized design components.
- Most software is always custom built.
- The software industry does seem to be moving (slowly) toward component-based construction.
Software Crisis?

- Projects running over-budget
- Projects running over-time
- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code difficult to maintain
- Software was never delivered

What is software engineering?

Pressman’s book

A discipline that encompasses
- process of software development
- methods for software analysis, design, construction, testing, and maintenance
- tools that support the process and the methods
Process, Methods, Tools

• Various tasks required to build and maintain software
  – e.g. design, testing, etc.
• SE process: the organization and management of these tasks
  – various process models
• SE methods: ways to perform the tasks
• SE tools: assist in perform the tasks
  – UML tools, IDEs, issue tracking tools

Importance of Historical Perspective

• Santayana half-truth:
  – “Those who cannot remember the past are condemned to repeat it”
• Don’t remember failures?
  – Likely to repeat them
• Don’t remember successes?
  – Unlikely to repeat them

Cf Barry Boehm/ICSE06 Keynote
History of SW Development

1950’s: engineer software like hardware

- Hardware-oriented software applications
  - Airplanes, circuits
- Economics: computer time more valuable than people time
  - Boehm supervisor, 1955: “We’re paying $600/hour for that computer, and $2/hour for you, and I want you to act accordingly.”

Cf Barry Boehm/ICSE06 Keynote
1960’s: software is NOT LIKE hardware

• Properties
  – Invisible, complex, had to be executed by computers, hard to change, doesn’t wear out, unconstrained by physical laws of nature

• Demand for programmers exceeded supply
  – Cowboy programmers as heroes
  – Computer Science Department formed
1960's: software is NOT LIKE hardware

- Code-and-fix process
- Better infrastructures
  - OS, compilers, utilities
- Some large successes
  - Apollo, BofA checking processing, ESS
- Failure of most large systems
  - Unmaintainable spaghetti code
  - Unreliable, undiagnosable systems
  - Code-and-fix process is too expensive

1970's Formal and Waterfall Approaches

- Structured programming, eliminate goto
- Formal methods
  - Specification, development, verification
  - Problems
    - Successful for small, critical programs
    - Proofs show presence of defects, not absence
    - Scalability of programmer community

Cf Barry Boehm/ICSE06 Keynote
• **Waterfall process model**
  - Requirements, design, coding, testing, operations (maintenance)
  - **Problems**
    - Customers' changing requirements destroy distinctions between phases
    - Heavyweight documentation hard to review and maintain
    - Assumption for smooth transition from design and implementation

---

**Large-Organization HW/SW Cost Trends (1973)**

<table>
<thead>
<tr>
<th>Year</th>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1955</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>1970</td>
<td>20%</td>
<td>80%</td>
</tr>
<tr>
<td>1985</td>
<td>40%</td>
<td>60%</td>
</tr>
</tbody>
</table>

% of total cost

*Cf Barry Boehm/ICSE06 Keynote*
1980’s Synthesis: Productivity, Reuse, Objects

- Major SW productivity enhancers
  - Working faster: tools and environments
  - Working smarter: processes and methods
  - Work avoidance: reuse, simplicity, objects
  - Technology silver bullets: AI, Do what I mean, programming by example
- Reuse libraries
- Object orientation
  - Smalltalk, Eiffel, C++

Cf Barry Boehm/ICSE06 Keynote

“No Silver Bullet”, Fred Brooks

IEEE Computer, 1987

“There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.”
• “Essential” difficulties to build software
  – Complexity: no two parts are alike
  – Conformity: conform to existing interfaces
  – Changeability: subject to change
  – Invisibility: has no ready geometric representation

• Closest thing to silver bullet: great designers and communicators

1990’s maturity models and agile methods

• Capacity Maturity Models (CMM)
  – Reliance on explicit documented knowledge
  – Heavyweight but verifiable, scalable

• Agile Methods
  – Reliance on interpersonal tacit knowledge
  – Lightweight, adaptable, not very scalable

• Other trends
  – reverse engineering, Open Source SW, Spiral process model

Cf Barry Boehm/ICSE06 Keynote
2000’s Synthesis

- Model-driven development
- Risk-driven model
- Service-oriented architecture
- Hybrid agile/plan-driven product and process architectures

Existing SW Problems

- Software is too expensive
- Software takes too long to build
- Software quality is low
- Software is too complex to support and maintain
- Software does not age gracefully
- Not enough highly-qualified people to design and build software
Data by the Standish Group (1995)

- $81B on canceled software projects
- $59B for budget overruns
- Only 1/6 projects were completed on time and within budget
- Nearly 1/3 projects were canceled
- Over half projects were considered “challenged”
- Among canceled and challenged projects
  - Budget overrun: 189% of original estimate
  - Time overrun: 222% of original estimate
  - Only 61% of the originally specified features

Software Myths
Management Myths

• “If we get behind schedule, we can just add more people and catch up”
• Fact: Adding people to a late project makes it even later
  – The people working now must spend time educating the newcomers

Customer Myths

• “A general statement of objectives is enough to start programming”
• Fact: An ambiguous statement of objectives leads to project failures
  – Unambiguous requirements need effective and continuous communication between customer and developer
Customer Myths

• “Changes in requirements are easy to deal with because software is flexible”
• Fact: Changes are hard and expensive

Practitioner’s Myths

• “Once we get the program running, we are done”
• Fact: 60-80% effort comes after the software is delivered for the first time
  – Bug fixes, feature enhancements, software reengineering, migration
Practitioner’s Myths

• “Until I get the program running, I cannot assess quality”
• Fact: Software reviews can be applied once code is written and are very effective; pair programming techniques as well

Practitioner’s Myths

• “The only deliverable work product is the running program”
• Fact: Need the entire configuration
  – Documentation of system requirements, design, programming, and usage
Practitioner’s Myths

• “SE will slow us down by requiring unnecessary documentation”
• Fact: SE is about creating quality
  • Better quality -> reduced rework
    -> faster delivery time
  • Brooks recommends time division of:
    1/3 planning; 1/6 coding; 1/4 component test and early system test; 1/4 system test

Learning Objectives

• Knowledge of basic concepts in software engineering
• Ability to do Object-oriented requirement analysis
• Ability to do Object-oriented design
• Good command of UML and Patterns
• Understanding importance of teamwork
Software Engineering

- Software is complex, expensive, late, low-quality, hard to maintain
- Goal: approach these problems using software engineering
- Key message: the field is very young - The term “SE” was introduced in 1968