Software

Software is:
- Executable programs
- Data associated with these programs
- Documents: user requirements, design documents, user/programmer guides, etc.
- Software plays a key role
  - Produces, manages, and presents information
  - Information society
  - Next step after industrial society

Software is everywhere

- Systems software
  - OS, compiler, loader
- Business software
  - payroll, accounting
- Scientific and engineering software
  - Computer-aided design, simulation, weather prediction, ...
- PC software
  - spreadsheets, word processing, games, ...

Software Crisis?

- People that design and build software have to deal with many problems
- Software crisis for the last 30 years?
- In reality, things are not that bad
  - Many more successes than failures
- But problems are persistent
- The field is still immature
  - e.g., compare with civil engineering, etc.

Some 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
Software Cost

- Software projects often go over budget
  - Hurts the company and the customer
- Bad for the project
  - Many projects are cancelled
  - People may get fired or may quit
- Bad for the final product
  - Features are not implemented
  - Bad quality: not enough money to get it right
  - Expensive in the long run

Software Time

- Software projects often take too long
- Loss of revenue and market share
  - Both for the vendor and for the client
  - E.g.: baggage system at Denver airport
    - Cost: $1 million per day
- Projects may become obsolete
  - Technology changes rapidly
  - Competing products already on the market
- Not enough time to implement all features and to ensure quality

Some Questions

- Studies of IT projects by the Standish Group (1995 and 1998)
  - 350+ IT managers, 8000+ applications
- What percentage of projects were
  - cancelled before being completed?
  - over budget and/or late?
  - completed on time and on budget?
- What was the cost/time overrun?

Success Rate

- Category 1: on time and on budget, with all initially specified features
- Category 2: over budget or over time, with fewer features than specified
- Category 3: cancelled

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category 1</td>
<td>31.1%</td>
</tr>
<tr>
<td>Category 2</td>
<td>16.2%</td>
</tr>
<tr>
<td>Category 3</td>
<td>52.7%</td>
</tr>
</tbody>
</table>

Overruns and Deficiencies

- Cost and time overruns
  - Averages for category 2 and category 3
- Cost overruns: 189% of original estimate
- Time overruns: 222% of original estimate
- Feature deficiencies: only 61% of the originally specified features were implemented
  - Average for category 2

Some Reasons for Failure

- Lack of user involvement
- Incomplete requirements and specs
- Changing requirements and specs
- Lack of executive support (politics)
- Lack of planning and management
- Inadequate resources and time
- Death-march projects
- Technology incompetence
Summary

- It is common for projects to fail
  - A third of the projects from this survey
  - Estimated cost for 1995: $81 billion
- It is common for projects to go over budget/time, and to have fewer features
  - Half of the projects in this survey
  - Time/cost is double of initial estimates
  - Estimated cost for 1995: $56 billion
- Similar results from the 1998 survey

Software Quality

- Software defects result in failures
  - Example 1: Windows crashes while you play a game at home
  - Example 2: The software that controls a nuclear reactor crashes
- Direct loss of life and money
  - Millions of dollars
- Indirect loss: missed opportunities
  - e.g. online purchases are down for a day
- Loss of credibility, bad publicity

Example: Ariane 5

- Ariane 5 rocket
- Built by the European Space Agency
- First launch: June 1996
- Crashed 40 seconds after launch
- Cost: $500 million
- No people on board
- Problem: software failure

What Happened?

- Overflow when velocity was converted from 64-bit integer to 16-bit integer
- The exception was not caught
- Inertial Reference System failed
- Backup system failed for the same reason
- Rocket went off course
- Self-destruct module (correctly) activated
- The code was OK for Ariane 4
- Same software, different environment

Software Myths

- "If we get behind schedule, we can just add more people"
  - Fact: Adding people to a late project makes it even later
    - Someone has to teach the new people
- "A general statement of objectives is enough to start programming"
  - Fact: Incomplete requirements are a major cause for project failures

Software Myths

- "Changes in requirements are easy to deal with because software is flexible"
  - Fact: Changes are hard and expensive
    - Especially during coding and after software deployment

<table>
<thead>
<tr>
<th></th>
<th>Definition</th>
<th>Development</th>
<th>After release</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5-6</td>
<td>60-100</td>
<td></td>
</tr>
</tbody>
</table>
Software Myths

• “Once we get the program running, we are done”
• Fact: Most effort comes after the software is delivered for the first time
  • Bug fixes, feature enhancements, etc.
• “The only product is the running program”
• Fact: Need the entire configuration
  • Documentation of system requirements, design, programming, and usage

Software Engineering

• Software is complex, expensive, late, low-quality, hard to maintain
• Goal: approach these problems using software engineering
  • Engineering disciplines: civil engineering, etc.
  • Body of knowledge, established practices, professional education, certification, etc.
• Key problem: the field is very young
  • The term “SE” was introduced in 1968

Definitions of SE

IEEE Standard 610.12

1. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software, and

2. The study of approaches as in (1).

That’s a mouthful, isn’t it?

Definitions of SE

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 software
  • e.g. design, testing, etc. (more later)
• SE process: the organization and management of these tasks
• SE methods: ways to perform the tasks
  • e.g. methods for software testing
• SE tools: assist in performing the tasks
  • e.g. design tools, IDEs, ...
  • UML tools: Rational Rose, Together Center