CS 451
Software Engineering

Yuanfang Cai
Room 104, University Crossings
215.895.0298
yfcai@cs.drexel.edu
This Course

- SE is unlike other CS topics
  - OS, DBMS, Compilers etc talk about specific types of software product
  - SW Engg. focuses on general software
- Software Engineering is the systematic approach to development, operation, maintenance, and retirement of sw.
- Basic Q. of SW Engg.: How to develop industrial-strength software?
Student Software vs. Industrial Software
Q: If you have to write a 10,000 line program in C to solve a problem, how long will it take?

Answers: generally range from 2-4 months

Let us analyze the productivity

- Productivity = output/input resources
- In SW output is considered as LOC
- Input resources is effort - person months; overhead cost modeled in rate for person month
- Though not perfect, some productivity measure is needed, as project has to keep it high

The productivity is 2.5-5 KLOC/PM
Q: **What is the productivity in a typical commercial SW organization?**

A: Between 100 to 1000 LOC/PM

Q: **Why is it low, when your productivity is so high?** *(people like you work in the industry)*

A: What the student is building and what the industry builds are two different things
Software…

- In a university a **student system** is built while the commercial org builds **industrial strength sw**

- What is the difference between a student program and industrial strength sw for the same problem?
Software...

**Student**
- Developer is the user
  - bugs are tolerable
  - UI not important
  - No documentation

**Industrial Strength**
- Others are the users
  - bugs not tolerated
  - UI v. imp. issue
  - Documents needed for the user as well as for the organization and the project
Software...

**Student**
- SW not in critical use
- Reliability, robustness not important
- No investment
- Don’t care about portability

**Industrial Strength**
- Supports important functions / business
- Reliability, robustness are very important
- Heavy investment
- Portability is a key issue here
- Heavy testing: (30%-50% total efforts)
The Real Problems
The HW/SW ratio for a computer system has shown a reversal from the early years.  
- In 50s, HW:SW :: 80:20  
- In 80s, HW:SW :: 20:80  

So, SW is very expensive  
- Importance of optimizing HW is not much  
- More important to optimize SW
Late & Unreliable

- 20-25% of SW projects never complete
  - Because after some time they realize that the final cost will be much higher
- Many companies report runaways
  - budget & cost out of control
  - consulting companies to help control them
- One defence survey found that 70% of the equipment problems are due to SW
- Many examples of software failures
Unreliable…

- SW failures are different from failures of mechanical or electrical systems
- In software, failures are not due to aging related problems
- Failures occur due to bugs or errors that get introduced during development
- I.e. the bug that causes a failure exists from start, only manifests later
Maintenance is More Expansive

- Once sw delivered, it enters maintenance phase
- Why is maintenance needed for sw when it does not wear with age?
  - Residual errors requiring corrective maint
  - Upgrades and environment changes – adaptive maint
- Over sw life, maint can cost more than the development cost of sw
Software Engineering Challenges
Basic Problem

User needs satisfies Software
SE must deal with problem of scale
- methods for solving small problems do not scale up for large problems
- industrial strength SW problems tend to be large

SE methods must be scalable

Two clear dimensions in this
- engineering methods
- project management

For small, both can be informal or ad-hoc, for large both have to be formalized
Scale...
An illustration of issue of scale is counting the number of people in a room vs taking a census

- Both are counting problems
- Methods used in first not useful for census
- For large scale counting problem, must use different techniques and models
- Management will become critical
## Scale: Examples

<table>
<thead>
<tr>
<th></th>
<th>LOC</th>
<th>Language(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gcc</td>
<td>980KLOC</td>
<td>C, C++, yacc</td>
</tr>
<tr>
<td>Perl</td>
<td>320 KLOC</td>
<td>C, perl, sh</td>
</tr>
<tr>
<td>Appache</td>
<td>100 KLOC</td>
<td>C, sh</td>
</tr>
<tr>
<td>Linux</td>
<td>30,000 KLOC</td>
<td>C, c++</td>
</tr>
<tr>
<td>Windows XP</td>
<td>40,000 KLOC</td>
<td>C, C++</td>
</tr>
</tbody>
</table>
Change

- Only constant in business is change!
- Software must change to support the changing business needs
- SE practices must accommodate change
  - Methods that disallow change, even if high Q and P, are of little use
Hardware-Characteristics

![Graph showing failure rate over time with two phases: "Infant mortality" and "Wear out".](image)
Increased failure rate due to side effects

Change

Actual curve

Idealized curve
Software - Characteristics

- Software is developed or engineered, it is not manufactured in the classic sense. Except obviously the need to copy the disk or download the program.

- Software does not wear out. Is this true?

- The book states,

  *Software is not susceptible to the environmental maladies the cause hardware to wear out.*
More Questions to Think About

- The differences between software and hardware
  - Do they change similarly? Why or why not?
  - Do they age similarly? Why or why not?
  - Do they all need to be specified before construction?

- Some open source software are much more successful than similar proprietary software. Does it mean that open source developers are smarter than company employees? Why or why not?

- Will making a software project open source always improve the quality of the product?
Summary

- The concept of Software
- The difference between industrial vs. student software
- The challenges of software development
- Software vs. Hardware
Summary

- The problem domain for SE is industrial strength software
- Software comprises programs, documentation, and data
- SE aims to provide methods for systematically developing SW
- Main goal – achieve high quality and productivity (Q&P)
Which of these statements are true?

- If we get behind schedule, we can add more programmers and catch up. Sometimes called the Mongolian horde concept. We practiced this at IBM.

- If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

- Project requirements continually change, but change can be easily accommodated, because software is flexible.

- Once we write the program and get it to work, our job is done.