Why Do We Study Software Engineering?

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

February 1, 2022

Outline

Table of Contents

What do we learn?

What should you expect to learn from this course?

To build high-quality software within budget using software engineering

- methodologies,
- techniques, and
- tools

Table of Contents

Why Software Engineering?

It is about writing software.

- Complexity
- Size

Software Development Effort

Size (LOC)	Example	
10^2	Class Exercise	Programming
10^3	Small Project	
10^4	Term Project	
10^5	Business Application	Software Engineering
10^6	Word Processor	
10^7	Operating System	

What happens when the complexity and size of a software system increases?

- Can the software be written in a single class?
- Can the software be written by a single programmer?
- Is the software used by a single user?
- ...

Software Failure

As size and complexity of software projects increased, so did the number of failed projects.

Software Failure Hall of Fame

Let's search software failures on the Web ...

Engineering Software!

"Engineering" software was thought to be the cure:

- Put some discipline into "programming."
- Do more than just coding/programming.
- "Study" (model/measure), "Understand" (analyze), and "Improve" (change) software development

But why another "engineering" discipline?

Software vs. Hardware Engineering

Arguments

- easy to modify vs. difficult to modify
  - Fix a bug in software vs. in hardware
- software engineering is still a young discipline
- software complexity is unprecedented

Some Statistics of Software Failures

- Chaos Report (1995) sampled some 300 software projects and reported that only about 16% of those projects "completed," "on time," and "within budget"! $\rightarrow$ That is 84% of projects failed!
- Chaos Report (2009) stated that software projects have improved with 32% "completed," "on time," and "within budget." $\rightarrow$ That is still 68% of projects—failure!
- Chaos Report (2015) stated that 39% successful software projects, 43% challenged software projects (late, over budget, or less functionality), and 18% failed software projects (cancelled or never used) $\rightarrow$ This means we still have 61% project failures.

Table of Contents

Software Development Process

Software engineering tasks are split into multiple steps or areas according to a software development process

Software Development Process Models

- Waterfall
- Spiral
- Agile
- ...

Example Engineering Areas

- Requirement (Chapter 6)
- Design (Chapters 7, 8)
- Implementation and Coding (Chapter 9)
- Verification (Chapter 10)
- Validation

Software Product Failures and Engineering Areas

Source: Casper Jones, 1992

Code errors	38.33%
Design errors	24.17%
Documentation errors	13.33%
Requirements errors	12.50%
Bad-fix errors	11.67%

All errors can be serious and very costly but

Should we worry about coding more or requirements more – Why?

Requirements errors are very "costly" if not detected & left in the product – Why?

"Key Strategies" to Success

3 "key" strategies to ensuring delivery of (Source: US General Accounting Office Report to US Senate, 2004):

1. high-quality software,
2. on time, and
3. within budget

- Focused attention on software development environment (people/tools/management/etc.)
- "Disciplined" development process
- Methodical use of metrics to gauge cost, schedule, and functional performance targets

Summary

Software Engineering

- helps us manage the complexity of designing software
- is a set of scientific principles and best practices

which we learn through lectures, in-class discussions, and term project.