CSE 470
Software Engineering

Imran Zahid
Lecturer
Computer Science and Engineering, BRAC University

Course Details

Course Goals

- Make you familiar with standard practices and techniques in the **software industry**.
- Reduce the gap between academic and industry practice.
- Teach you to not just write **good code**, but design and create **functional**, **scalable** and **maintainable** software.

Marks Distribution

1. Assignments	5 %
2. Quizzes	15 %
3. Mid-Term Examination	25%
4. Project	20 %
5. Final	35 %

Join the Course Classroom - Section 13

Classroom Code: pc26ao4

https://tinyurl.com/cse470-s13

Join the Course Classroom - Section 14

Classroom Code: ct5mef6

https://tinyurl.com/cse470-s14

Join the Course Classroom - Section 15

Classroom Code: uqfgrq4

https://tinyurl.com/cse470-s15

Reference Books

- Clean Code: A Handbook of Agile Software Craftsmanship
- Design Patterns: Elements of Reusable Object-Oriented Software

General Rules

- Makeup Mid and Final will be conducted as per BRACU policy.
- Quiz makeup will be taken only in case of emergencies.
- Assignments need to be submitted by the given deadline, each day beyond the deadline will result in deduction of marks [1/5th of the marks per day].
- Visit me in my consultation hours for any kind of query or discussion.

Project

- This course is best learnt by working on the project.
- It also holds a reasonable weight in your final result - 20%
- That said, the topics covered in the classes will allow you to improve and refine your project to a point that just working on the project will not reach.

Have you done a long-term (4-6 months) project before?

In that project did you use any version control system (e.g. Git and Github)?

Did you use Git and Github to actively maintain the progress of your project, or just host the files?

Did you write any documentation (e.g. code comments, docstrings, etc.)?

Are you confident that if I asked you to pick up your project again, and add a feature, you could do it in one week?

Are you confident that if I asked someone ELSE to pick up your project, and add a feature, they could do it in one week?

Introduction

Software: What is it?

- Computer programs
  - Source Code
  - Data Structures
- Associated documentation
  - Requirements and specification documentation
  - Design documentation
  - Test suites and testing documentation

Why is this Distinction Necessary?

Computer Program

Delivered as a one-off product and its success is based solely on its functionality.

Software

Often developed for specific clients or general markets, requiring ongoing support and updates.

- Having documentation makes this process easier and more efficient

Real-Life Software

Inevitably most software systems are LARGE systems.

This means:

- Many people involved in design, building and testing,
- Team effort, not individual effort
- Many millions of $ spent on design and implementation
- Millions of lines of source code
- Lifetime measured in years or decades
- Continuing modification and maintenance

Example — Eclipse

Eclipse is a popular software development environment for Java.

Some interesting characteristics of a recent release:

Lines of source code	> 1,350,000
Effort(person-years)	> 400
Classes	17,456
Inheritance relations	15,187
Methods	124,359
Object instantiations	43,923
Fields	48,441
Call relations	1,066,838
Lifetime bugs	> 40,000
Est. Development cost	> $ 54,000,000

Software Engineering: What is it?

Formal Definition

ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB) defines software engineering as:

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

Software Engineering

The science (and art) of building and maintaining high quality software systems -

- On time
- Within budget
- Functioning correctly
- Meeting performance expectations

Goals of Software Engineering

- To produce software
  - That meets all requirements.
  - With minimum effort.
  - At the lowest possible cost.
  - In the least possible time.
  - That is easily maintained and modified.
  - That maximizes the profitability of the software production effort.
- In practice, not all of these ideal goals are achievable at the same time.
- The art of software engineering is balancing these goals for a particular project.

Why is Software Engineering Important?

- Cost of getting software wrong is often terrible.
- Monetary loss of the software producer.
- Injury or loss of human life. Broken software can KILL people.
- Software producer's profitability depends on producing software efficiently and minimizing maintenance effort. Software reuse is an economic necessity.
- Immense body of old software that must be rebuilt or redesigned to be usable on modern computer systems.
- Over $600,000,000,000 spent each year on producing software.

What is Good Software?

Good software should deliver the required functionality to the user and should be efficient, maintainable, dependable and usable.

- Correct
- Maintainable and easy to modify
- Well modularized with well-designed interfaces
- Has a good user interface
- Well documented
  - Internal documentation for maintenance and modification
  - External documentation for end users
- Efficient
  - Not wasteful of system resources, cpu & memory
  - Optimized data structures and algorithms

Challenges Faced in Software Engineering

- All desired attributes cost $ to achieve
- Interaction between attributes
  - High efficiency may degrade maintainability, reliability
  - More complex user interface may degrade efficiency, maintainability, and reliability
  - Better documentation may divert effort from efficiency and reliability
- Software engineering management has to consider trade-offs satisfying desired goals.

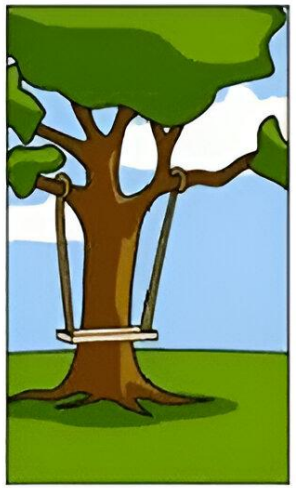
What makes Software Development Hard?

- Imprecise and incomplete requirements and specifications
- Changing requirements and specifications
- Communication and coordination
- Inability to accurately estimate effort or time required
- Programmer variability and unpredictability
- Overwhelming complexity of large systems, more than linear growth in complexity with size of the system
- Poor software development processes

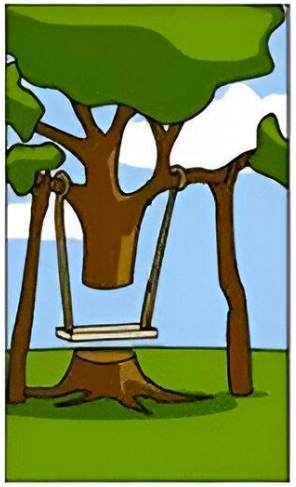
Why do Software Fail?

- Lack of communication
  - Management, clients and developers
- Misunderstanding requirements
- End user expectations
- Lack of expertise
- Requirements froze/change

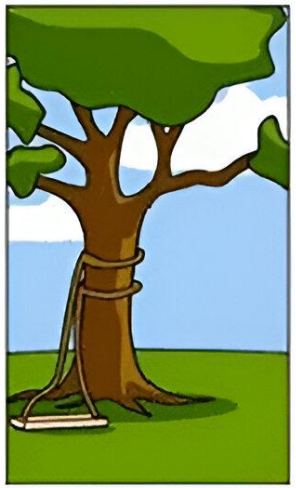
How the customer explained it



How the project leader understood it

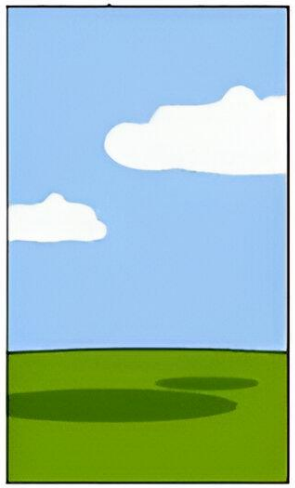


How the engineer designed it

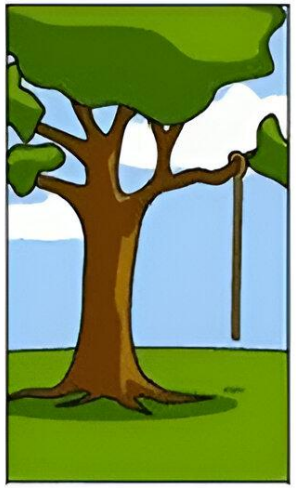


How the programmer wrote it

How the sales executive described it

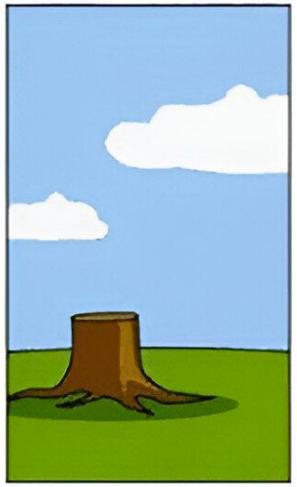


How the project was documented



What operations installed

How the customer was billed



How the helpdesk supported it

What the customer really needed

Software: Why must it change?

- Software must be adapted to meet the needs of new computing environments or technology.
- Software must be enhanced to implement new business requirements.
- Software must be extended to make it interoperable with other more modern systems or databases.
- Software must be re-architected to make it viable within a network environment.

TLDR: Because the client wants it.

Need Different Approach for Large Software

- Need formal management of software production process.
- Formal & detailed requirements, specification and design.
- Much more attention to modularity and interfaces.
- Must be separable into manageable pieces.
- Need version control.
- More emphasis on rigorous and thorough testing.
- Need to plan for long term maintenance and modification.
- Need more documentation, internal and external.

Project

Project Details

Project Purpose

The purpose of the project is to provide hands-on training on how to develop a software from scratch.

Key Definitions

Requirement: An action/capability that is expected of a software/product to fulfil

Feature: A set of logical actions that need to be performed in order to fulfil a part of a requirement

Project Details

Example of a requirement:

1. Customers can add products to their cart

Example of features:

For requirement (1), the features which need to be implemented are:

1. The system should be able to display products to customers
2. The system should be able to book products and mark them as added to the cart of a customer if the product is in stock

Project Constraints

- Students may develop either a web application, android application, iOS application or a desktop application.
- Students may use any tech stack/programming language/scripting to develop their application.
- The project must have at least five (5) requirements with four (4) features each.
- The application developed MUST follow the MVC architecture.

BRAC UNIVERSITY
Inspiring Excellence

Thank you