Introduction to Software Engineering

(Chapter 1)

What is Software?

- Wikipedia: Software is a collection of computer programs (and related data) that provides the instructions for telling a computer what to do and how to do it.
- Does software include:
  - Google chrome? MS Word? Mac OS X?
  - An Excel spreadsheet?
  - the source code program?
  - the machine code version of the program?
- Software is executable.

8 different types of software applications

1. Stand-alone applications
   - Run on a local computer, such as a PC.
   - Do not need to be connected to a network.
2. Interactive transaction-based applications
   - Execute on a remote computer
   - Accessed by users from their own computers.
   - These include web apps such as e-commerce applications.
3. Embedded control systems
   - Software systems that control and manage hardware devices.
   - Numerically, there are probably more embedded systems than any other type of system.

8 different types of software applications

4. Batch processing systems
   - Business systems designed to process data in large batches.
   - Process large numbers of individual inputs to create corresponding outputs.
5. Entertainment systems
   - Primarily for personal use
   - Intended to entertain the user.
6. Systems for modeling and simulation
   - Developed by scientists and engineers to model physical processes or situations
   - Include many, separate, interacting objects.

8 different types of software applications

7. Data collection systems
   - Collect data from their environment using a set of sensors.
   - Send that data to other systems for processing.
8. Systems of systems
   - Composed of a number of other software systems.
   - Department of Defense applications

Two kinds of software products

- Generic products
  - Stand-alone systems that are marketed and sold to anyone.
  - Examples – Microsoft office, iPad apps, Angry birds, software for specific markets: appointments systems for dentists.
  - The developer organization decides what the software should do
- Customized products
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
  - The customer decides what the software should do and how it should be changed.

What is Software Engineering?

- Software engineering is an engineering discipline that is concerned with **all aspects of software production** from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering discipline
  - Using appropriate theories and methods to solve problems
  - Bearing in mind organizational and financial constraints.
  - "getting results of the required quality within the schedule and budget"
- All aspects of software production
  - Not just technical process of development.
  - Project management
  - The development of tools, methods etc. to support software production.

What is Software Engineering?

Another definition from the textbook

- Software engineering is a systematic approach to **the production of software** that takes into account practical cost, schedule, and dependability issues, as well as the needs of software customers and producers

Why do we need Software Engineering?

- Without it, software tends to be unreliable and more difficult to maintain (change).
- As size and complexity of software projects increases, so do the number of failed projects.
- Software **Project** Failure.
- Software **Product** Failure.

Why do we need Software Engineering?

- Software **Project** Failures.
  - 1994 US Federal Aviation Administration
    Advanced Automation System canceled after $2.6B spent
  - 2002 McDonald's Corp.
    Innovative information-purchasing system cancelled after $170M spent
  - 2004 J Sainsbury PLC [UK]
    Supply-chain management system abandoned after deployment costing $527M

Why do we need Software Engineering?

- Software **Product** Failures.
  - Y2K problem
    $300 B to fix code storing only 2 digits for the date
  - 1994 Intel Pentium microprocessor
    $475 M: error in chip causes error in floating point division
  - 1996 Ariane 5 failure
    Rocket crashes: reusing code for Ariane 4 causes overflow
  - 1985-87 Therac-25 medical accelerator
    5 patients die after receiving lethal doses of radiation

Why do Projects fail?

http://spectrum.ieee.org/computing/software/why-software-fails

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined specifications (requirements)
- Poor reporting of the project's status
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Stakeholder politics
- Commercial pressures

Software engineering processes address many of these problems

Software Quality: Essential attributes of good software

Product characteristic	Description	Example
Functional Correctness	The software meets its specifications. It is generally free of defects (bugs).	A calculator app always gives the correct answer, for every operation.
Maintainability	Software is written in such a way so that it is easy to change, to meet the changing needs of customers.	Apple successfully updates the Safari Web browser to work with a new version of Mac OS X.
Dependability	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.	Unauthorized users are not able to access your banking account in an online banking app.

Software Quality: Essential attributes of good software

Product characteristic	Description	Example
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.	When you sort your iTunes library using one of the column headers, you see the results almost immediately.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.	Facebook privacy controls: seem to change often and are difficult for users to understand. (Negative example)

Case Studies

The following case studies are used for examples in the book (see chapter 1 for background).

- A personal insulin pump
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- A mental health case patient management system
  - An information system used to maintain records of people receiving care for mental health problems.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas.