Introduction to Software Engineering

Chapter 1

Introduction to Software Engineering
in the textbook

- 1.1 Professional software development
  - What is meant by software engineering.
- 1.2 Software engineering ethics
  - See CS2315
- 1.3 Case studies
  - An introduction to three examples that are used in later chapters in the book.

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

Who writes software?

- Computer science students
- Business people, scientists and engineers write small programs for their own purposes
- Hobbyists (iphone developers?)
- Most software is developed by professional software developers:
  - usually for some specific business purpose
  - used by people other than the developer(s)
  - usually developed by a team
  - often sold to customers

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).
- See "Software Failures" lecture.

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.	Successfully updating 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)

Software process

- A structured set of activities used to develop a software system/product
- Four fundamental activities in a software process:
  - Specification, where customers (and engineers) define the software that is to be produced and the constraints on its operation.
  - Development, where the software is designed and programmed (implemented).
  - Validation, where the software is checked to ensure that it is what the customer requires.
  - Evolution, where the software is modified to reflect changing customer and market requirements.

Software engineering vs computer science

- Computer Science is the study of computer systems including algorithmic processes and the principles involved in the design of hardware and software.
- Software Engineering is the practice of designing and implementing large, reliable, efficient and economical software by applying the principles and practices of engineering.
  - Some knowledge of computer science is essential for software engineers.

1.2 Software engineering ethics

- The professional societies in the US, ACM and IEEE, have cooperated to produce a code of ethical practice.
- The Code contains eight principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.
- http://www.acm.org/about/se-code

Rationale for the ACM/IEEE code of ethics (from the preamble)

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.

Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession.

Some ethical issues specific to software engineering

- Confidentiality
  - Engineers should respect the confidentiality of employers and clients
- Competence
  - Engineers should not misrepresent their level of competence.
  - They should not knowingly accept work which is outside their competence.
- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc.
  - They should ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers.

1.3 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.