Introduction to Software Engineering

Chapter 1

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?

- Computer programs?

- Wikipedia: **Software** is a set of programs, procedures, algorithms and its documentation concerned with the operation of a data processing system
  - Software is executed
  - Includes data, configurations, documentation
  - “data processing system" seems too restrictive

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.
<table>
<thead>
<tr>
<th>8 different types of software applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Batch processing systems</td>
</tr>
<tr>
<td>- Business systems designed to process data in large batches.</td>
</tr>
<tr>
<td>- Process large numbers of individual inputs to create corresponding outputs.</td>
</tr>
<tr>
<td>5. Entertainment systems</td>
</tr>
<tr>
<td>- Primarily for personal use</td>
</tr>
<tr>
<td>- Intended to entertain the user.</td>
</tr>
<tr>
<td>6. Systems for modeling and simulation</td>
</tr>
<tr>
<td>- Developed by scientists and engineers to model physical processes or situations</td>
</tr>
<tr>
<td>- Include many, separate, interacting objects.</td>
</tr>
</tbody>
</table>

| 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   |

<table>
<thead>
<tr>
<th>Who writes software?</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Computer science students</td>
</tr>
<tr>
<td>• Business people, scientists and engineers write small programs for their own purposes</td>
</tr>
<tr>
<td>• Hobbyists (iphone developers?)</td>
</tr>
<tr>
<td>• Most software is developed by professionals:</td>
</tr>
<tr>
<td>- usually for some specific business purpose</td>
</tr>
<tr>
<td>- used by people other than the developer(s)</td>
</tr>
<tr>
<td>- usually developed by a team</td>
</tr>
<tr>
<td>- often sold to customers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Two kinds of software products</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Generic products</td>
</tr>
<tr>
<td>- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.</td>
</tr>
<tr>
<td>- Examples – Microsoft office, iPad apps, Angry birds, software for specific markets such as appointments systems for dentists.</td>
</tr>
<tr>
<td>• Customized products</td>
</tr>
<tr>
<td>- Software that is commissioned by a specific customer to meet their own needs.</td>
</tr>
<tr>
<td>- Examples – embedded control systems, air traffic control software, traffic monitoring systems.</td>
</tr>
</tbody>
</table>
Who determines what the software should do?

- **Generic products**
  - The developer organization decides what the software should do, and how it should be changed and maintained.
  - Often the marketing department takes on the role of “the customer”.

- **Customized products**
  - The customer decides what the software should do and how it should be changed.
  - The developers can have some input on how these decisions are carried out.

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.

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*

<table>
<thead>
<tr>
<th>Product characteristic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>Software should be written in such a way so that it is easy to change, to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.</td>
</tr>
<tr>
<td>Dependability</td>
<td>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.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.</td>
</tr>
<tr>
<td>Acceptability</td>
<td>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.</td>
</tr>
</tbody>
</table>

*Beyond “correctness” (meeting specifications)  

Software process

- The systematic approach to software engineering
- Four fundamental activities in a software process:
  - Software specification, where customers (and engineers) define the software that is to be produced and the constraints on its operation.
  - Software development, where the software is designed and programmed (implemented).
  - Software validation, where the software is checked to ensure that it is what the customer requires.
  - Software evolution, where the software is modified to reflect changing customer and market requirements.

Software engineering vs computer science

- Computer science is concerned with theories and methods that underlie computers and software systems.
- Software engineering is concerned with the practical problems of production software.
  - Some knowledge of computer science is essential for software engineers.

Frequently asked questions about software engineering

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is software?</td>
<td>Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.</td>
</tr>
<tr>
<td>What are the attributes of good software?</td>
<td>Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.</td>
</tr>
<tr>
<td>What is software engineering?</td>
<td>Software engineering is an engineering discipline that is concerned with all aspects of software production.</td>
</tr>
<tr>
<td>What are the fundamental software engineering activities?</td>
<td>Software specification, software development, software validation and software evolution.</td>
</tr>
<tr>
<td>What is the difference between software engineering and computer science?</td>
<td>Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.</td>
</tr>
</tbody>
</table>
Frequently asked questions about software engineering

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What are the key challenges facing software engineering?</td>
<td>Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.</td>
</tr>
<tr>
<td>What are the costs of software engineering?</td>
<td>Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution (maintenance) costs often exceed development costs.</td>
</tr>
<tr>
<td>What are the best software engineering techniques and methods?</td>
<td>While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.</td>
</tr>
</tbody>
</table>

Case Studies

- **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.

**Insulin pump control system**

- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels
- Sends signals to a micro-pump to deliver the correct dose of insulin.
- **Safety-critical system as**
  - low blood sugars can lead to brain malfunctioning, coma and death;
  - high-blood sugar levels have long-term consequences such as eye and kidney damage.

**Insulin pump hardware architecture**

![Insulin pump hardware architecture diagram]
Activity model of the insulin pump

Two Essential high-level requirements

- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.

A patient information system for mental health care

- A medical information system that maintains information about
  - patients suffering from mental health problems and
  - the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- These clinics may be held in hospitals or local medical practices or community centers.

MHC-PMS

- The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.
MHC-PMS goals

- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients.

MHC-PMS key features

- **Individual care management**
  - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.

- **Patient monitoring**
  - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.

- **Administrative reporting:** The system generates monthly management reports showing
  - the number of patients treated at each clinic,
  - the number of patients who have entered and left the care system,
  - number of patients sectioned,
  - the drugs prescribed and their costs, etc.

MHC-PMS concerns

- **Privacy**
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorized medical staff and the patient themselves.

- **Safety**
  - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

The organization of the MHC-PMS

[Diagram of MHC-PMS system]
Wilderness weather station

- The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.
- Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
- The larger weather information system collects data from the weather stations and makes it available to other systems for processing.

The weather station’s environment

Each subsystem is explained on the next slide

Weather information system

- The weather station system (multiple)
  - This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.
- The data management and archiving system
  - This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.
- The station maintenance system
  - This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.

Additional weather station software functionality

- Monitor the instruments, power and communication hardware and report faults to the management system.
- Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
- Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.