Chapter 1- Introduction

Lecture 1

Topics covered

- ✧ Professional software development
  - What is meant by software engineering.
- ✧ Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.
- ✧ Case studies
  - An introduction to three examples that are used in later chapters in the book.

Software engineering

- ✧ Software is everywhere: the economies of ALL developed nations are dependent on software.
- ✧ Software systems are abstract and intangible, not constrained by properties of materials or governed by physical laws.
- ✧ Software systems can quickly become extremely complex, difficult to understand, and expensive to change.
- ✧ Software engineering seeks to help manage these issues, and has allowed us to explore space and create the internet.

Software products

Software engineering is concerned with developing software products (software which can be sold to a customer):

- ✧ Generic products
  - Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
  - Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.
- ✧ 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.

Product specification

- Generic products
  - The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- Customized products
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

Software quality

- Software quality is more than just meeting the functional specifications.
- The software must behave in appropriate ways:
  - a banking system must be secure,
  - an interactive game must be responsive, etc.
- The software usually needs to be updated so the code itself must be well-organized and understandable.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
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.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
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.

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.
- All aspects of software production
  - Not just technical process of development.
  - Project management
  - The development of tools, methods etc. to support software production.

Software process activities

- 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.
- 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.

Computer science vs software engineering

- 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.

General issues that affect most types of software

- Heterogeneity
  - Various computers and mobile devices must run same software.
  - Integrate new software with legacy system.
- Business and social change
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available.
  - Existing software must be adapted and new software must be rapidly developed.
- Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

Need new tools, techniques, software engineering methods to address these.

Software engineering diversity

- There are many different types of software systems:
  - No universal set of software techniques is applicable to all of them.
- The software engineering methods and tools used depend on
  - the type of application being developed,
  - the requirements of the customer and
  - the background of the development team.

Application types

- Stand-alone applications
  - Run on a local computer, such as a PC.
  - Do not need to be connected to a network.
- 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.
- Embedded control systems
  - Software systems that control and manage hardware devices.
  - Numerically, there are probably more embedded systems than any other type of system.

Application types

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

Application types

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

Software engineering fundamentals

- Fundamental software engineering principles that apply to all types of software systems:
  - Systems should be developed using a managed and understood development process.
  - Dependability and performance are important for all types of system (see essential attributes).
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Reuse software that has already been developed rather than write new software whenever appropriate.

Frequently asked questions about software engineering

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Frequently asked questions about software engineering

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	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.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Key points

- ✧ Software engineering is an engineering discipline that is concerned with all aspects of software production.
- ✧ Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- ✧ The high-level activities of specification, development, validation and evolution are part of all software processes.
- ✧ There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- ✧ The fundamental notions of software engineering are universally applicable to all types of system development.

Chapter 1- Introduction

Lecture 2

Software engineering ethics: Issues of professional responsibility

- Confidentiality
  - Engineers should respect the confidentiality of employers and clients
  - regardless of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence.
  - They should not knowingly accept work which is outside their competence.

Issues of professional responsibility

- 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.
  - Computer misuse ranges from
    - relatively trivial (game playing on an employer's machine) to
    - extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour 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.

Rationale for the ACM/IEEE code of ethics

- *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.*

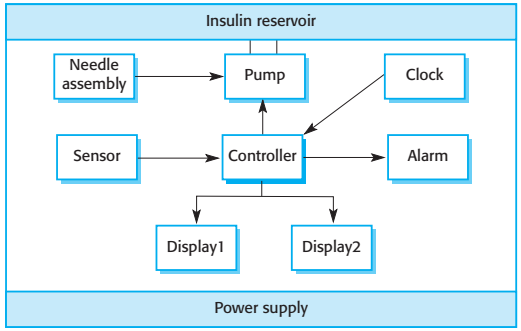
Case studies

- 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.

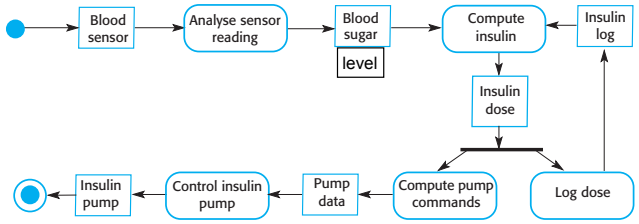
Insulin pump control system

- 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



Activity model of the insulin pump



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.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

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.

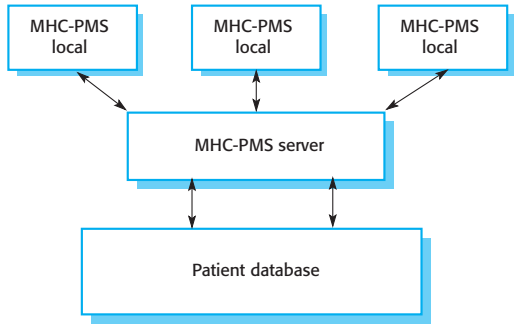
MHC-PMS

- 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.

The organization of the MHC-PMS



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.

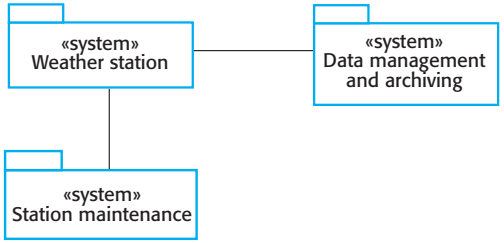
MHC-PMS concerns

- Privacy
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised 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.

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 weather information system collects data from the weather stations and makes it available to other systems for processing.

The weather station's environment



Weather information system

- The weather station system
  - 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 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.

Key points

- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.
- Three case studies are used in the book:
  - An embedded insulin pump control system
  - A system for mental health care patient management
  - A wilderness weather station