Software Engineering

Lecture 1

Introduction

Adapted from:

Chap 1. Sommerville 9th ed.

Chap 1. Pressman 6th ed.

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.

The Software Product

- **What is Software?**
  - Software is a set of items or objects that form a "configuration" that includes: Programs, documents, data ...
- **Who Creates Software?** Software Engineers
- **Why is Software Important?** Affects nearly every aspect of life.
- **How is Software Built?** By applying a structured process.
- **What are the Work Products?** Programs, documents, data
- **Dual Role of Software**
  - Product
  - Process that delivers products

Software Characteristics

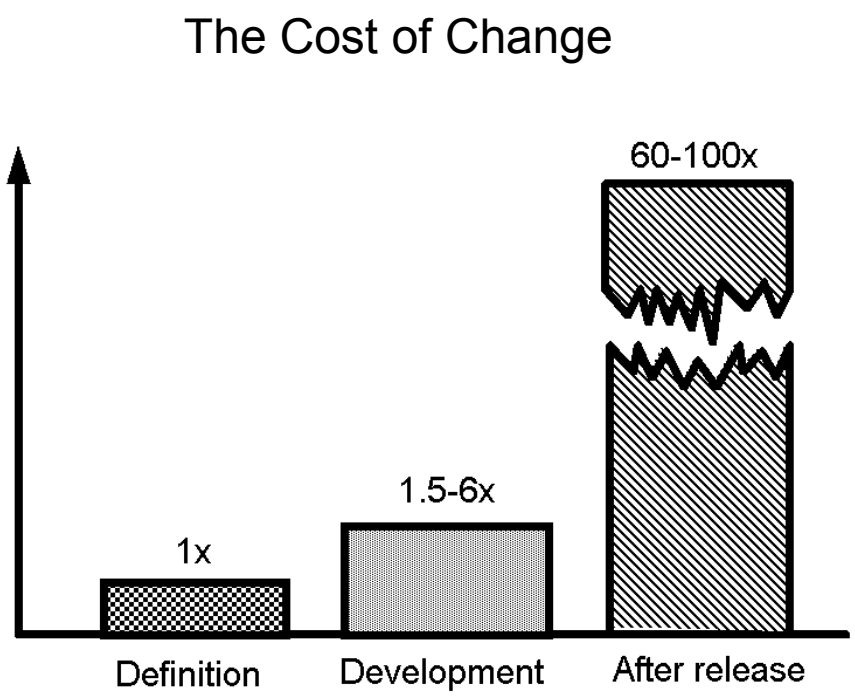
- ✧ software is engineered
- ✧ software doesn’t wear out
- ✧ software is complex
- ✧ software is like an ‘aging factory’

Software economy

- ✧ More and more systems are software controlled
- ✧ The economies of ALL developed nations are dependent on software.
- ✧ Expenditure on software represents a significant fraction of the GNP in all developed countries.
  - Gartner Group Reports (2007): $3 Trillion on IT Worldwide (6% of GDP).

Software costs

- Software errors cost the U.S. economy $60 billion annually in rework, lost productivity, and actual damages.
- Software costs often dominate computer system costs.
- Software costs more to maintain than to develop.
  - For systems with a long life, maintenance costs may be several times development costs.



Software Size

Program/Application	Lines of Code
Solaris	7,400,000
Netscape	17 million
Space Station	40 million
Space Shuttle	10 million
Boeing 777	7 million
Windows 95	Under 5 million
Linux	1.5 million

Software Size Categories

Category	Programmers	Duration	Size (Lines of Code)
Extremely Large	> 200	> 6 years	> 1,000,000
Very Large	20 - 200	3 - 6 years	100,000 - 1,000,000
Large	5 - 20	2 - 3 years	20,000 - 100,000
Medium	2 - 5	6 months – 2 years	3,000 – 20,000
Small	1 - 2	1 - 6 months	500 - 3,000
Trivial	1	1 - 4 weeks	< 500

Software Disasters (1)

- **Medical Machine Kills** (1985), Cost: Three people dead, three people critically injured
  - *Disaster:* Canada's Therac-25 radiation therapy machine malfunctioned and delivered lethal radiation doses to patients.
  - *Cause:* Because of a subtle bug called a race condition, a technician could accidentally configure Therac-25 so the electron beam would fire in high-power mode without the proper patient shielding.

- **Patriot Fails Soldiers** (1991), Cost: 28 soldiers dead, 100 injured
  - *Disaster*: During the first Gulf War, an American Patriot Missile system in Saudi Arabia failed to intercept an incoming Iraqi Scud missile. The missile destroyed an American Army barracks.
  - *Cause*: A software rounding error incorrectly calculated the time, causing the Patriot system to ignore the incoming Scud missile.

Software Disasters (2)

- **Ariane Rocket Goes Boom** (1996), Cost: $500 million
  - *Disaster*: Ariane 5, Europe's newest unmanned rocket, was intentionally destroyed seconds after launch on its maiden flight. Also destroyed was its cargo of four scientific satellites to study how the Earth's magnetic field interacts with solar winds.
  - *Cause*: Shutdown occurred when the guidance computer tried to convert the sideways rocket velocity from 64-bits to a 16-bit format. The number was too big, and an overflow error resulted. When the guidance system shut down, control passed to an identical redundant unit, which also failed because it was running the same algorithm.

- **Mars Climate Crasher** (1998), Cost: $125 million
  - *Disaster*: After a 286-day journey from Earth, the Mars Climate Orbiter fired its engines to push into orbit around Mars. The engines fired, but the spacecraft fell too far into the planet's atmosphere, likely causing it to crash on Mars.
  - *Cause*: The software that controlled the Orbiter thrusters used imperial units (pounds of force), rather than metric units (Newtons) as specified by NASA.

So What Does All This Mean?

- ✧ Seat of your pants programming will not work!
- ✧ Urgent need for techniques that can be used to manage the development of large software systems
- ✧ This is part of what software engineering is all about

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

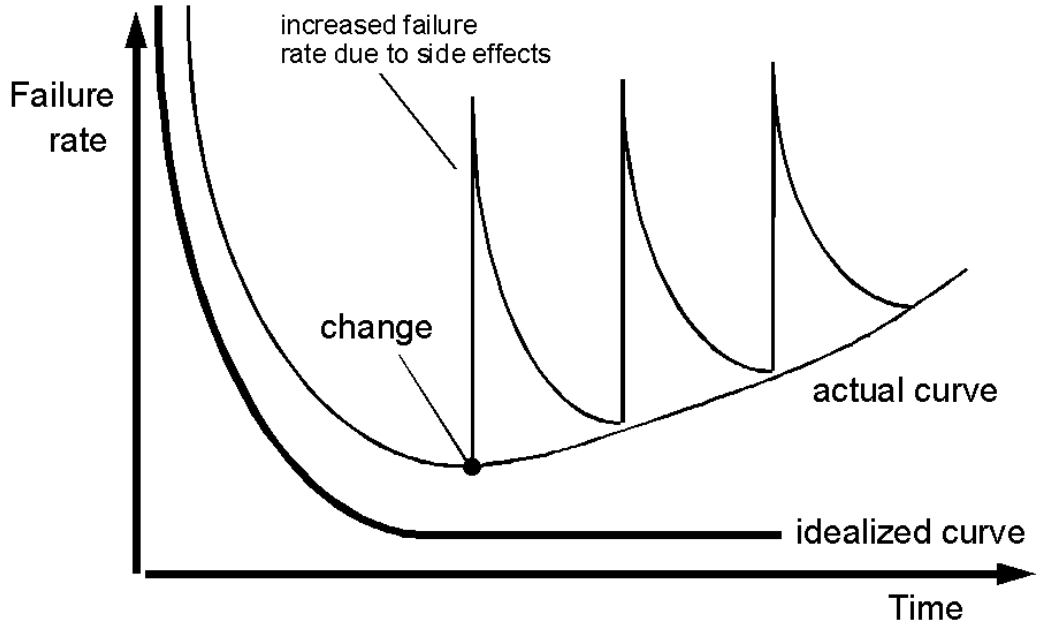
Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For 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.

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 and security	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.

Questions Asked of Software Engineers

- Why does it take so long to finish software?
- Why are development costs so high?
- Why can't we find all the bugs before delivery?
- Why do we continue to have difficulties measuring progress in the software development process?



Software engineering

Software Engineering Definition:

- The establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.
- (1)The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. The study of approaches, as in (1).

Importance of software engineering

- Individuals and society rely on advanced software systems.
  - We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
  - For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

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.

General issues that affect most software

- ✧ Heterogeneity
  - Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- ✧ Business and social change
  - Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
- ✧ Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

Software engineering diversity

- There are many different types of software systems and there is no universal set of software techniques that is applicable to all of these.
- 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.

Software Applications

- ✧ system software
- ✧ real-time software
- ✧ business software
- ✧ engineering/scientific software
- ✧ embedded software
- ✧ PC software
- ✧ mobile software
- ✧ WebApps (Web applications)

Software engineering fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Software engineering and the web

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web services allow application functionality to be accessed over the web.
- Web-based systems are complex distributed systems, but the fundamental principles of software engineering are as applicable to them as they are to any other types of systems.

Web software engineering

- Software reuse is the dominant approach for constructing web-based systems.
  - When building these systems, you think about how you can assemble them from pre-existing software components and systems.
- Web-based systems should be developed and delivered incrementally.
  - It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.
- User interfaces are constrained by the capabilities of web browsers.
  - Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with local scripting are more commonly used.

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.
- ✧ The fundamental notions of software engineering are universally applicable to all types of system development.

Key points

- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- The fundamental ideas of software engineering are applicable to all types of software system.

Software engineering ethics

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

Issues of professional responsibility

- ✧ Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective 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 outwith 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 be careful to 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, say) 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 organizations 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 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.*

The ACM/IEEE Code of Ethics

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

Ethical principles

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Ethical dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

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
  - A 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.

Figure 1.4 Insulin pump hardware architecture

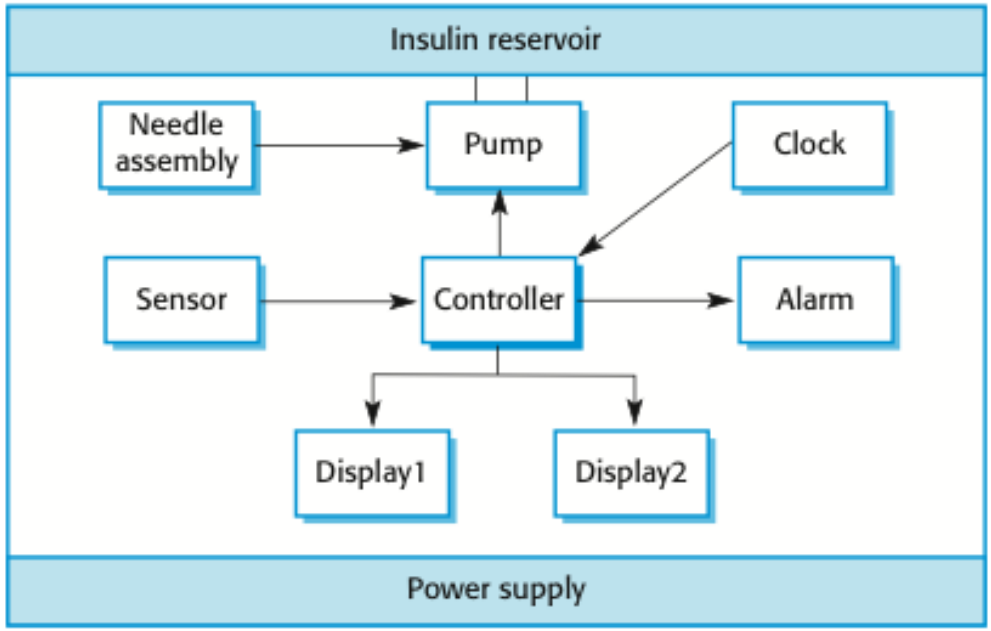
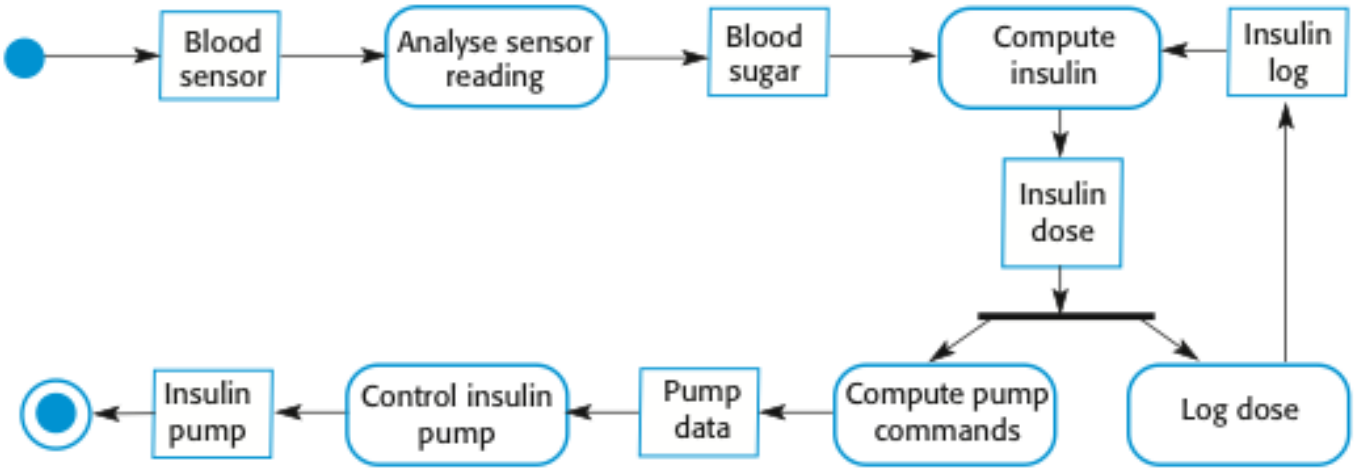


Figure 1.5 Activity model of the insulin pump



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 patient information system to support mental health care is 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.
- To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

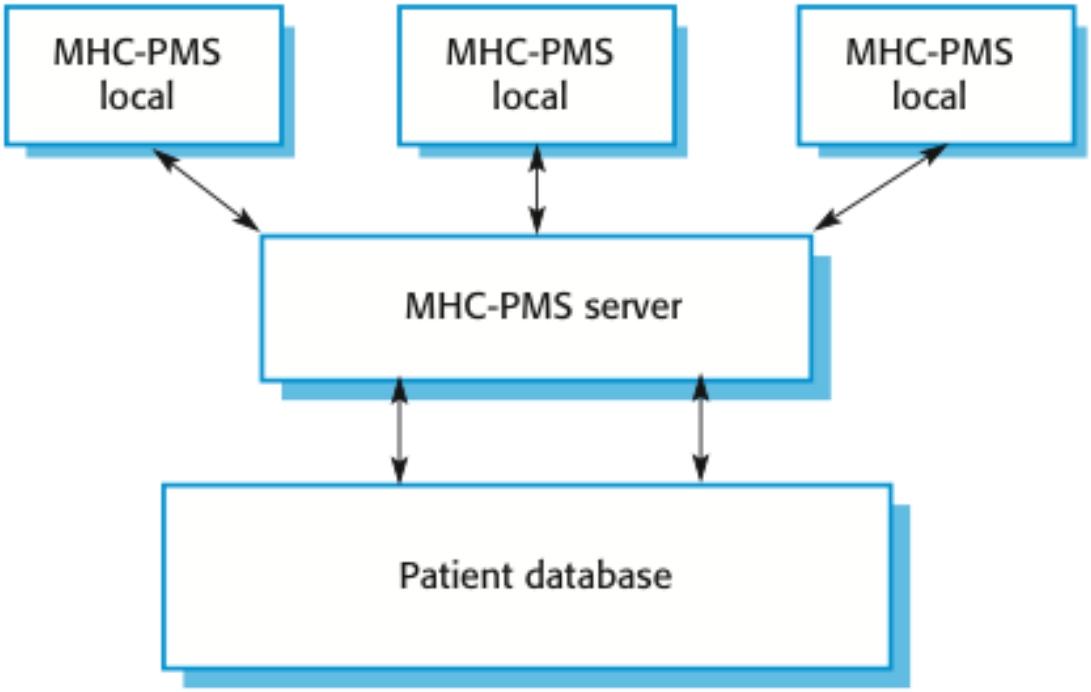
MHC-PMS

- 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.

Figure 1.6 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.

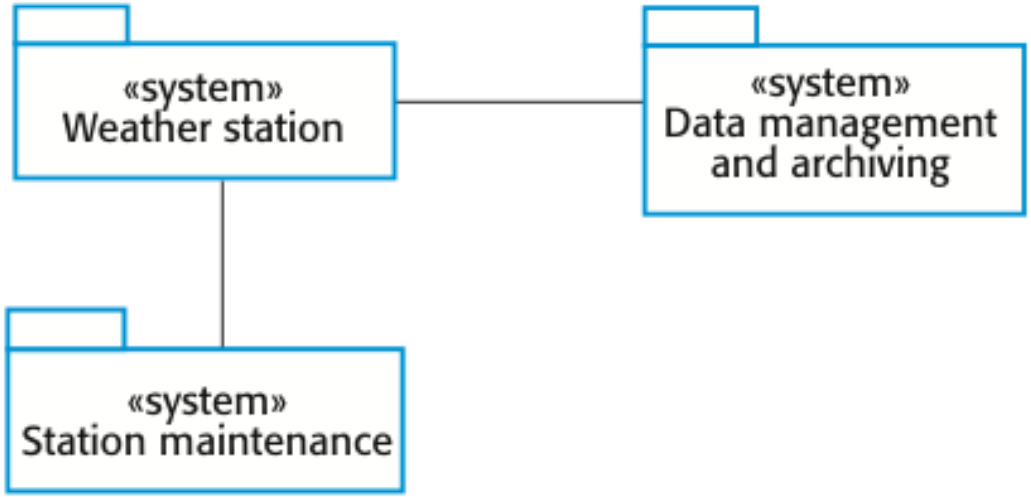
MHC-PMS concerns

- ✧ 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 station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period.
  - Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

Figure 1.7 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