Chapter 1- Introduction

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

- ✧ The economies of ALL developed nations are dependent on software.
- ✧ More and more systems are software controlled
- ✧ Software engineering is concerned with theories, methods and tools for professional software development.
- ✧ Expenditure on software represents a significant fraction of GNP in all developed countries.
- ✧ History of software engineering
  - The notion of SE was first proposed in 1968 at a conference held to discuss what was then called the **'software crisis'**

Software costs

✧ Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.

✧ Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.

✧ Software engineering is concerned with cost-effective software development.

Software project failure

✧ *Increasing system complexity*

- As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.

✧ *Failure to use software engineering methods*

- It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

Professional software development

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.

Software products

- ✧ 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.

✧ The distinction between these system product types is becoming increasingly blurred

- Systems are built with a generic product as a base, which is then adapted to suit the requirements of a customer, e.g., ERP of SAP

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.

* The specific set of attributes for a software system obviously **depends on its application**.

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, methods, and tools to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

* Software engineers adopt a systematic and organized approach to their work. The systematic approach is sometimes called a software process.

Importance of software engineering

- ✧ More and more, 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

- A software process is a sequence of activities leading to the production of software. Four fundamental 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.
- Different types of systems need different development processes
  - Generic activities may organized in different ways and described at different levels of detail

General issues that affect software

✧ Heterogeneity

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- Applications can be implemented as microservices where different microservices can be implemented using different languages and databases.

✧ 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.
  - For example, the recent development of AI, Big Data, FinTech, and IoT technologies and their applications could largely change business and society

General issues that affect software

- ✧ Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
- ✧ Scale
  - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

Software engineering diversity

- 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.
- There are many different **types** of software system 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**.
- Perhaps the **most significant factor** in determining which software engineering methods and techniques are most important is **the type of application being developed**.

Application types

- ✧ Stand-alone applications
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- ✧ Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- ✧ Embedded control systems
  - These are software control 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
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- ✧ Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- ✧ Systems for modeling and simulation
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

Application types

- ✧ Data collection systems
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- ✧ Systems of systems
  - These are systems that are composed of a number of other software systems, e.g., an ERP system.
- ✧ Boundaries between these system are **blurred**. However, you use different software engineering techniques for <u>each type of system</u> because the software has **<u>quite different characteristics</u>**

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**. Of course, **different processes are used for different types of software**.
  - Dependability and performance are important for all types of system. (i.e., *quality of software*)
  - Understanding and managing the software specification and requirements (what the software should do) are important. Managing user's expectations so that a useful system can be delivered within budget and to schedule.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Software engineering vs. emerging technology

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
  - The mobile is another platform, including smartphone, wearable and IoT devices
- Web services (discussed in Chapter 19) allow application functionality to be accessed over the web.
  - Microservices
- Cloud computing is an approach to the provision of computer services where applications run remotely on the 'cloud'.
  - Users do not buy software, but pay according to use.
- AI and Machine Learning
  - Deep learning for Software engineering, or vice versa

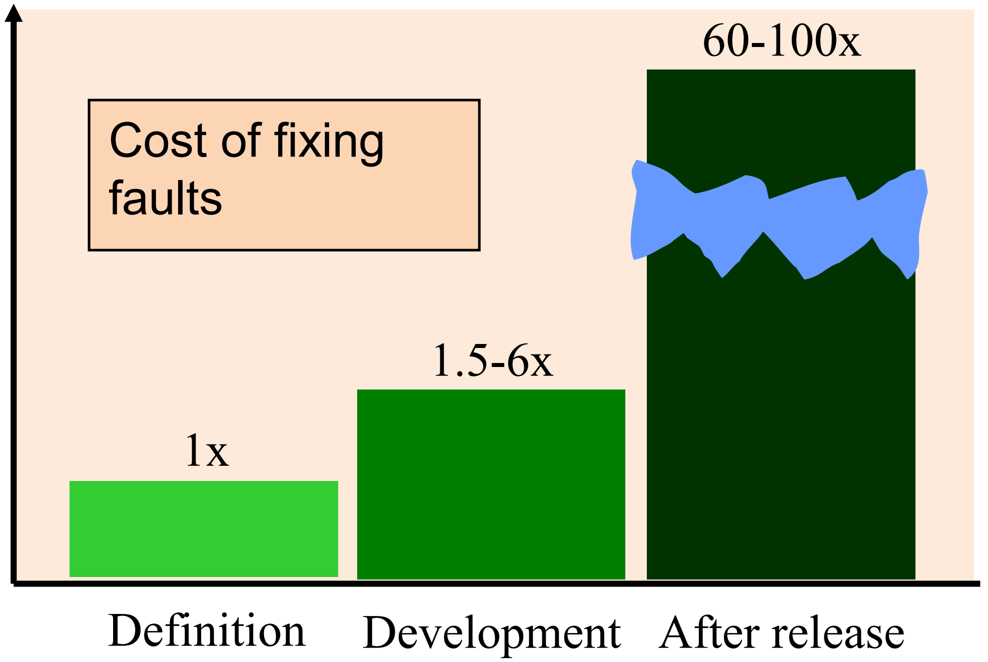
Management Myths

- *Myth*: We already have a book that's full of standards and procedures for building software, won't that provide my people with everything they need to know?
  - *Reality*: The book of standards may very well exist, but is it used? Not if:
    - Are software practitioners aware of its existence?
    - Does it reflect modern software engineering practice?
    - Is it complete?
    - Is it streamlined to improve time to delivery while still maintaining a focus on quality?
- *Myth*: If we get behind schedule, we can add more programmers and catch up
  - *Reality*: Software development is not a mechanistic process like manufacturing. As Brooks said: "adding people to a late software project makes it later"
- *Myth*: If I decide to outsource the software project to a third party, I can just relax and let them build it
  - *Reality*: If an organization does not understand how to manage and control software projects internally, it won't be able to outsource effectively

Customer Myths

- ✧ *Myth:* A general statement of objectives is enough to start writing programs—we can fill in the details later
  - *Reality:* A poor up-front definition is the major cause of failed software efforts. If you don't know what you want at the beginning, you won't get it

- *Myth*: Project requirements continually change, but change can be easily accommodated because software is flexible
  - *Reality*: It is true that software requirements change, but the impact of change varies with the time at which it is introduced



Practitioner's Myths

- *Myth*: Once we write the program and get it to work, our job is done
  - *Reality*: Someone once said that "the sooner you begin 'writing code', the longer it'll take you to get done." Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer.
- *Myth*: Until I get the program "running" I have no way to assess its quality
  - *Reality*: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the formal technical review. Software reviews are more effective than testing for finding certain classes of software defects.
- *Myth*: Software engineering will make us create voluminous and unnecessary documentation and will always slow us down
  - Reality: Software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times. Document should be used if and only if it increases quality

Software engineering ethics

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 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 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 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 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.
  - The problem here is that there are **no absolutes** when its comes to safety
  - The appropriate ethical position **depends entirely on the views of the individuals who are involved**
- Participation in the development of military weapons systems or nuclear systems.
- Collecting user's information and tracing system usage information
  - Privacy protection vs. improving system performance and user satisfaction
- Application of AI and network technology
  - Fake news and social media manipulation, user data collection and misuse, face and voice recognition and misuse

Case studies

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
  - Mentcare. 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.
- ✧ iLearn: a digital learning environment
  - A system to support learning in schools

Insulin pump control system

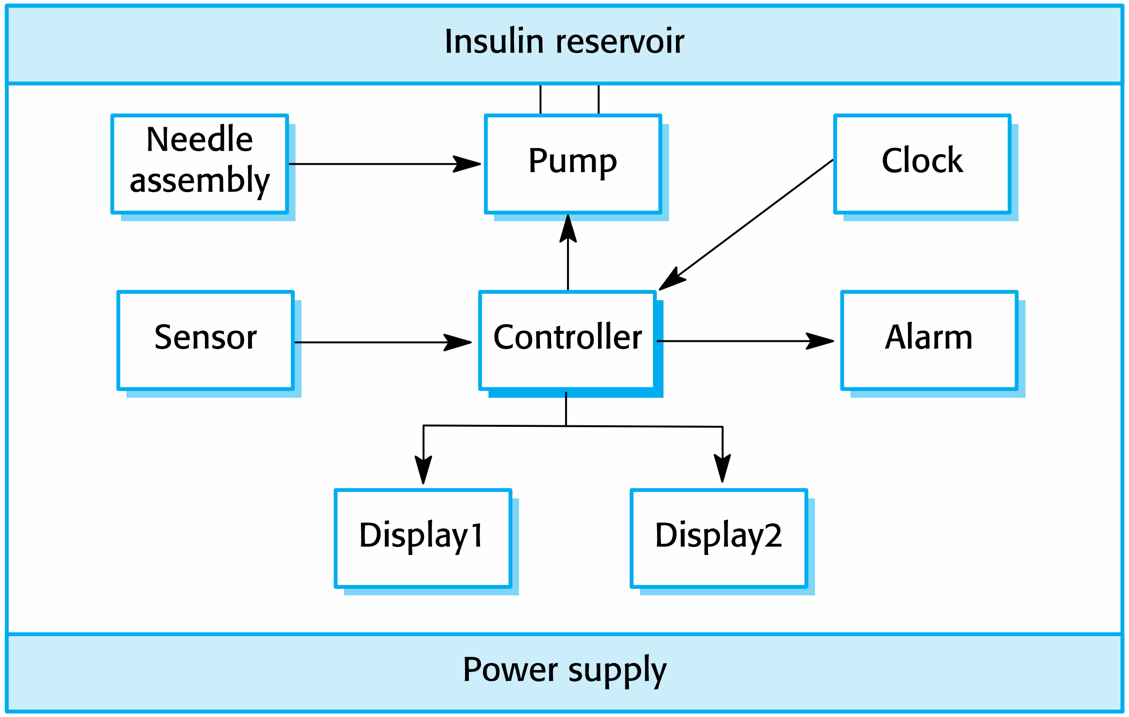
✧ Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.

✧ Calculation based on the rate of change of blood sugar levels.

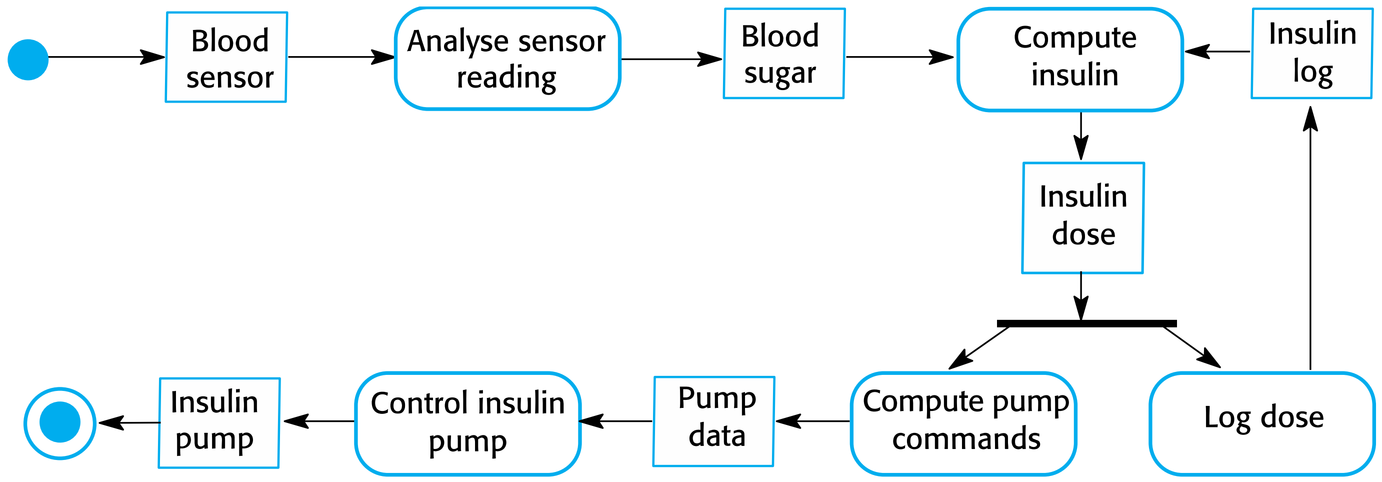
✧ Sends signals to a micro-pump to deliver the correct dose of insulin.

✧ 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.

Mentcare: 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.

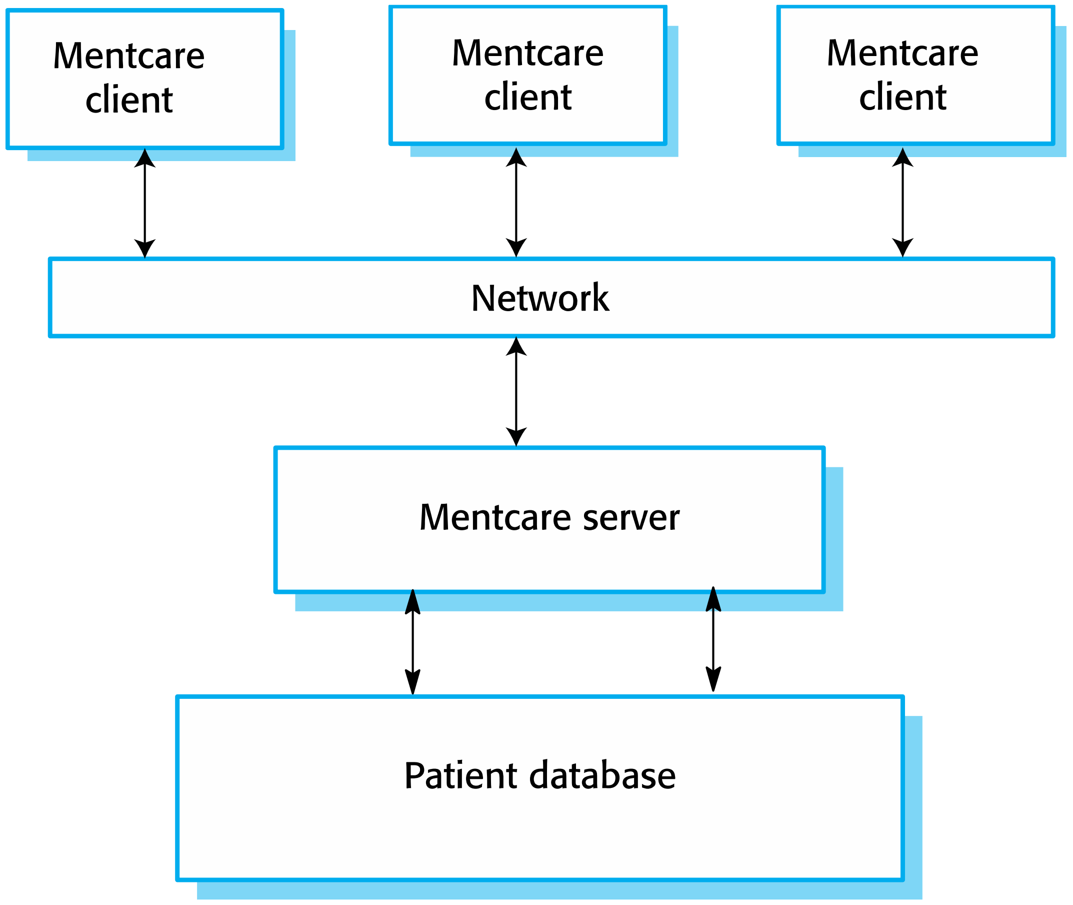
Mentcare

- ✧ Mentcare 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.

Mentcare 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 Mentcare system



Key features of the Mentcare system

- ✧ 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.

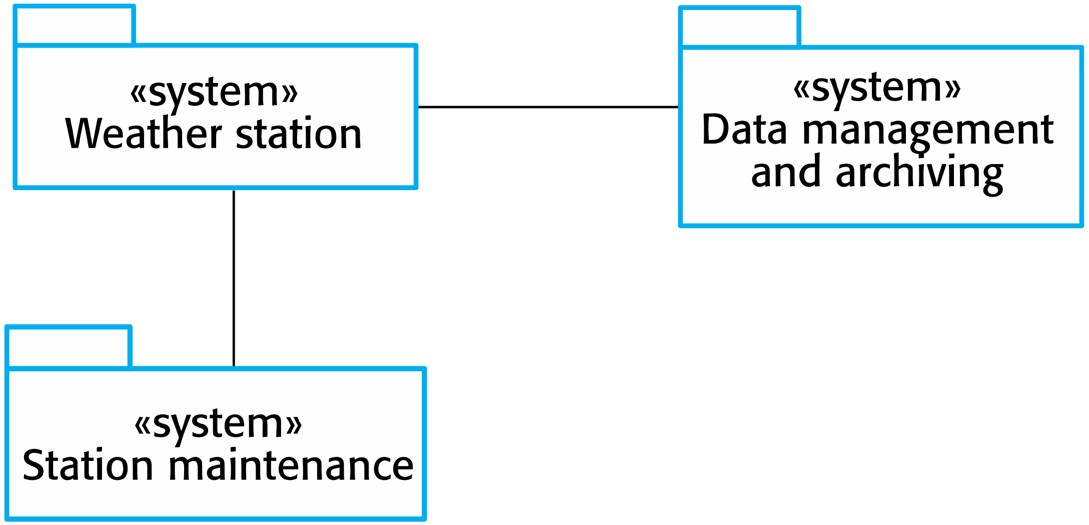
Mentcare system concerns

- 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.

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.

iLearn: A digital learning environment

- ✧ A digital learning environment is a framework in which a set of general-purpose and specially designed tools for learning may be embedded plus a set of applications that are geared to the needs of the learners using the system.
- ✧ The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs.
  - These can be general applications such as spreadsheets, learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games and simulations.

Service-oriented systems

✧ The system is a service-oriented system with all system components considered to be a replaceable service.

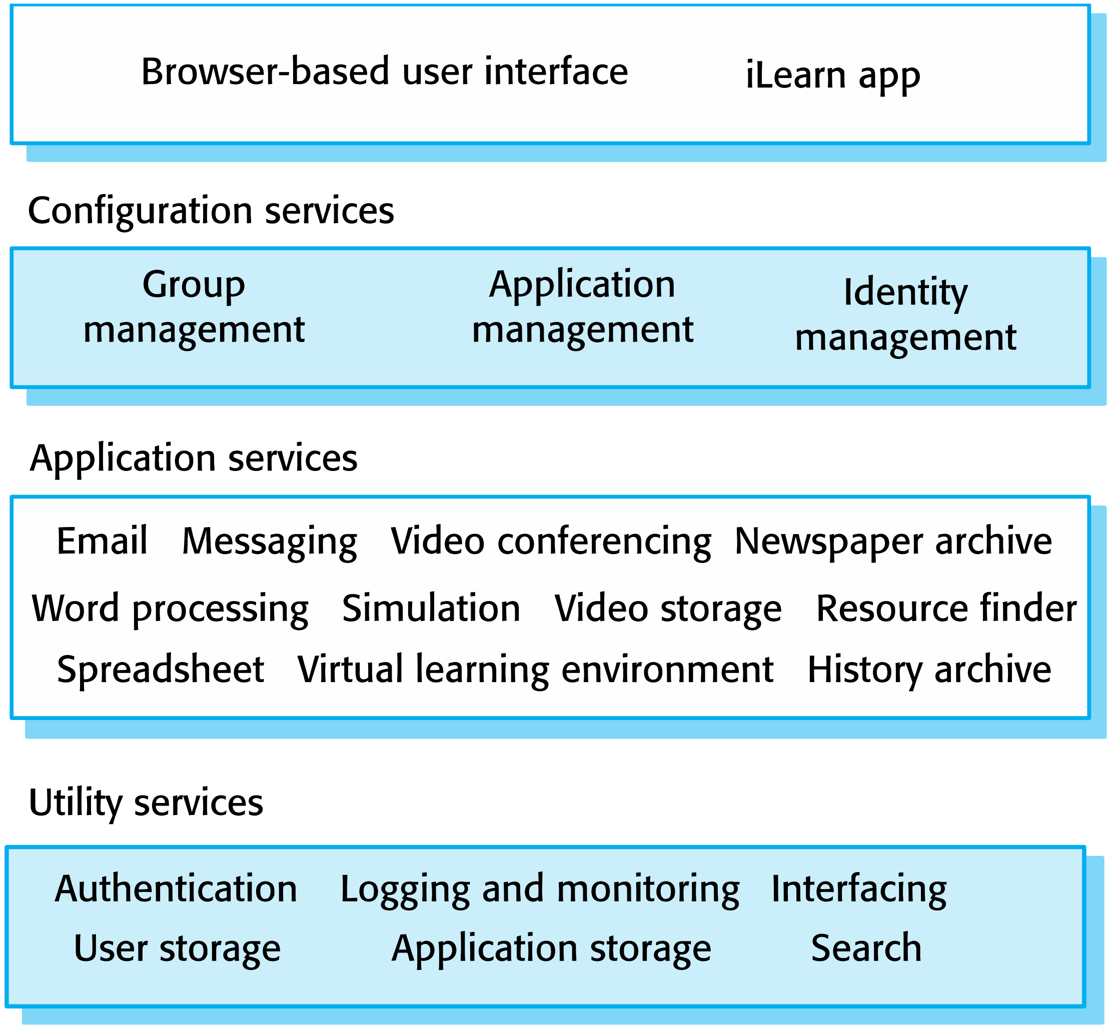
✧ This allows the system to be updated incrementally as new services become available.

✧ It also makes it possible to rapidly configure the system to create versions of the environment for different groups such as very young children who cannot read, senior students, etc.

iLearn services

- ✧ *Utility services* that provide basic application-independent functionality and which may be used by other services in the system.
- ✧ *Application services* that provide specific applications such as email, conferencing, photo sharing etc. and access to specific educational content such as scientific films or historical resources.
- ✧ *Configuration services* that are used to adapt the environment with a specific set of application services and do define how services are shared between students, teachers and their parents.

iLearn architecture



iLearn service integration

- *Integrated services* are services which offer an API (application programming interface) and which can be accessed by other services through that API. Direct service-to-service communication is therefore possible.
- *Independent services* are services which are simply accessed through a browser interface and which operate independently of other services. Information can only be shared with other services through explicit user actions such as copy and paste; re-authentication may be required for each independent service.

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 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.