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

- 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

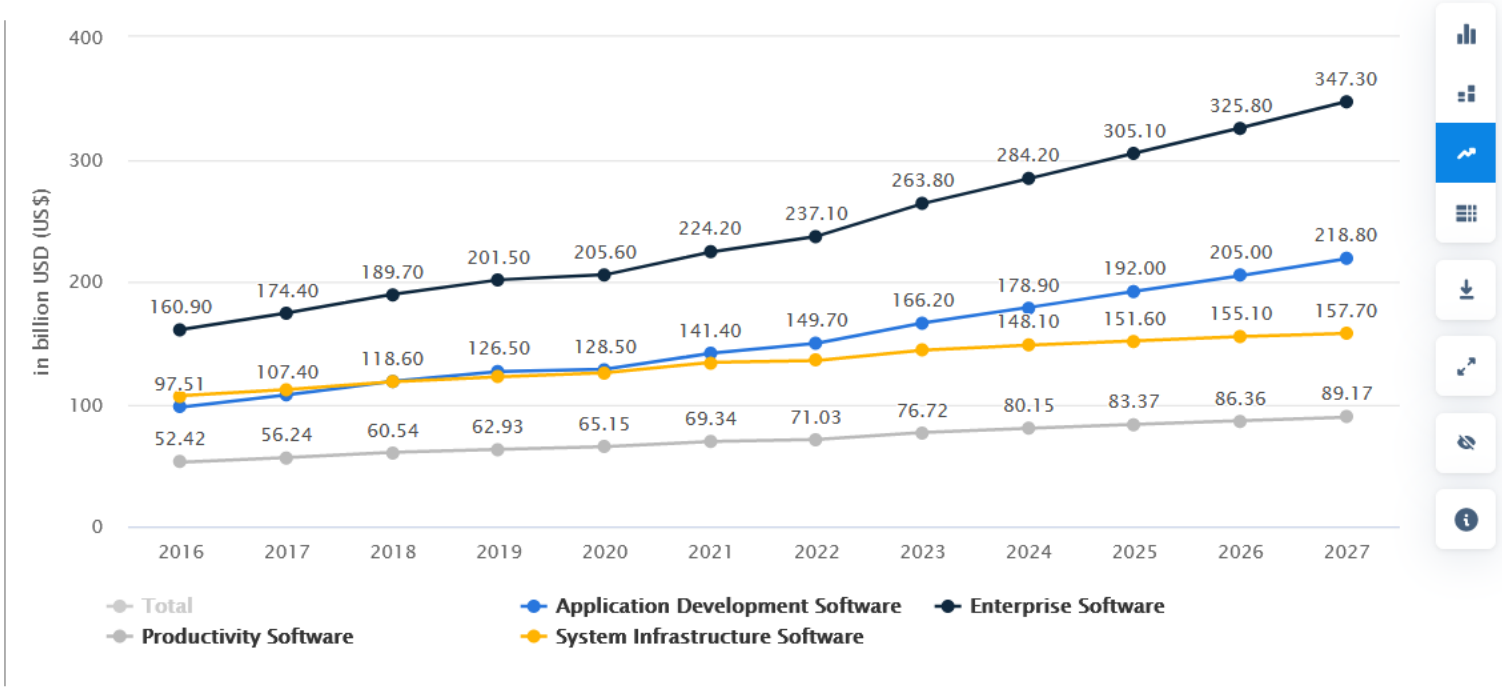
Software - Worldwide

- Revenue in the Software market is projected to reach US$593.40bn in 2022
- The market's largest segment is Enterprise Software with a projected market volume of US$237.10bn in 2022
- Revenue is expected to show an annual growth rate (CAGR 2022-2027) of 6.50%, resulting in a market volume of US$812.90bn by 2027
- In global comparison, most revenue will be generated in the United States (US$297.10bn in 2022)

https://www.statista.com/outlook/tmo/software/worldwide

Software - Worldwide

REVENUE BY SEGMENT



Notes: Data shown is using current exchange rates and reflects market impacts of the Russia-Ukraine war.

Most recent update: Jul 2022

Sources: Statista, Financial Statements of Key Players, National statistical offices

https://www.statista.com/outlook/tmo/software/worldwide

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

Software costs

- 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

Software project failure

✧ *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

Why Software Engineering?

- The problem is **complexity**
- Many sources, but **size** is the key:
  - Windows XP and 7 contain 45 million lines of code
  - Mac OS X 10.4 contains 86 million lines of code
  - Debian 5.0 contains 324 million lines of code
  - Debian 7.0 contains 419 million lines of code
  - Linux kernel 3.6 has 15.9 million lines of code
  - Linux kernel 4.2.3 has 20 million lines of code

Software engineering is about managing this complexity

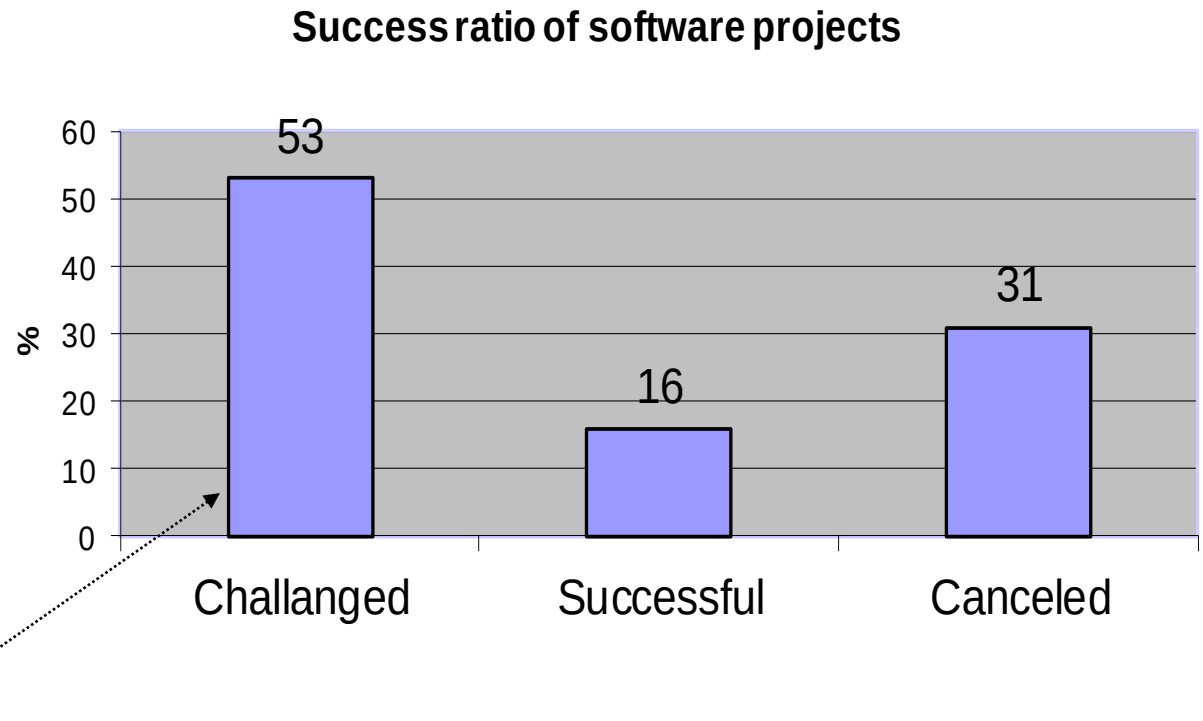
https://en.wikipedia.org/wiki/Source_lines_of_code#Example

State of the Practice

Estimate (C LOC)	Early	On time	Delayed	Canceled
13.000	6.06%	74.77%	11.83%	7.33%
130.000	1.24%	60.76%	17.67%	20.33%
1.300.000	0.14%	28.03%	23.83%	48.00%
13.000.000	0.0%	13.67%	21.33%	65.00%

Patterns of Software Failure and Success, C. Jones, 1996

State of the Practice



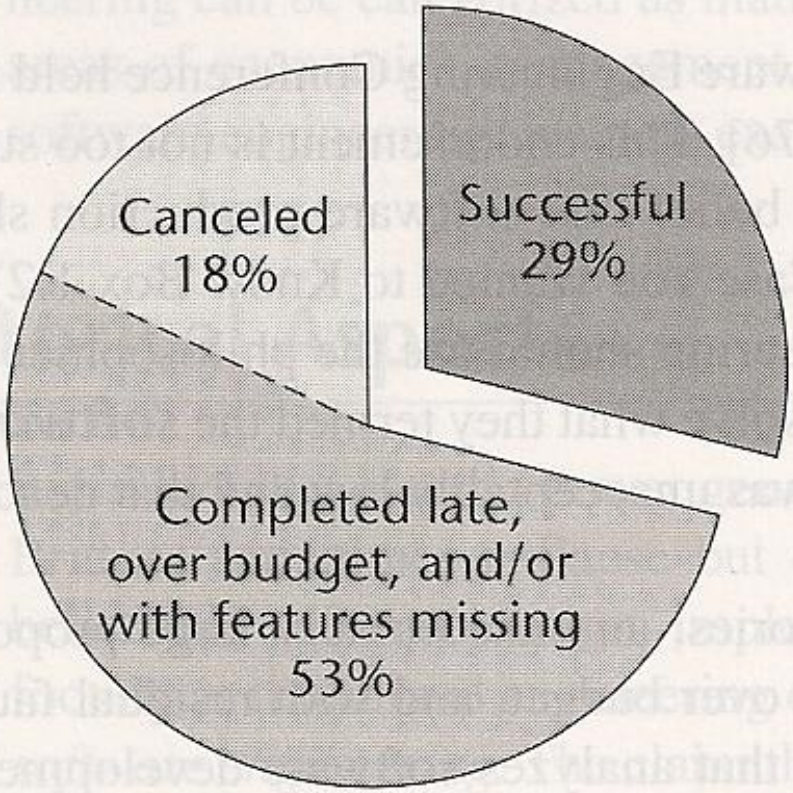
Cost overrun	178%
Late delivery	230%
Missing capability	58%

The Standish Group, 1994

State of the Practice

Figure 1.1
The outcomes of over 9000 development projects completed in 2004.

Source: [Hayes, 2004].



HORROR STORIES

Horror Stories

- Air traffic control (FAA modernization)
  - $5.6 million cost overrun
  - 8-year delay
  - 2 out of 4 systems are canceled, third system’s requirements are %48 decreased

Horror Stories

- ✧ USA navy finance system
  - 4 times cost overrun, $230 million
  - Canceled after 9 years
- ✧ Comanche Helicopter
  - In ten years, cost increased 10 times, $34.4 million
  - Requirements are %74 decreased

Catastrophic Computer Systems Development Examples

- ✧ The Denver Airport Baggage System
  - Approved in 1989, plan was to have a new airport with five runways operational by the end of 1993 – could support 3 simultaneous landings under any weather conditions
  - Baggage handling system was to support 20 major airlines
    - Proposed costing $193 billion and was based on 4000 independent ‘telecars’ which would carry baggage
    - Laser scanners were installed to read bar-coded luggage tags and 5000 photocells tracked the movement of telecars
    - The entire system was controlled by a network of about 300 computers

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

- ✧ The Denver Airport Baggage System
  - Beset by problems, mainly in SW
    - Telecars were mis-routed or crashed, and baggage damaged or misplaced
  - Without adequate baggage handling facilities, airport was unable to open
  - Delay estimated to cost $1.1 million per day
  - After spending a further $88 million to improve the system, the airport opened in February 1995, 15 months late

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

✧ The Clementine Space Mission

- In January 1994, US Ballistic Missile Defense Organization launched Clementine satellite
  - Mission is to use its sensors and cameras to survey the surface of the Moon, to conduct various military experiments and to visit the asteroid belt of the asteroid Geographos
- Clementine is distinctive in that it is a low cost project (less than $100 million) and uses off-the-shelf advanced technology
- In initial stages, mission was an outstanding success
  - Topographical survey of the Moon was performed and over 1.5 million images were recorded

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

✧ The Clementine Space Mission

- In May 1994, as Clementine was being readied for the next phase of its mission, on-board computer system independently activated several of its thruster rockets during a period when telemetry was interrupted
- The effect of 11 minute burn was to expend all the fuel tanks and it was unable to maneuver to allow it to visit the asteroid
- A computer system fault, believed to be a software bug, had resulted in the mission being prematurely curtailed

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

- ✧ The Intel Pentium ™ Processor
  - "A software bug that's encoded in hardware" (Halfhill 1995)
  - Illustrates the problems that can occur when designs are committed to silicon, and high cost of remedying problems in mass produced systems
  - The problem lies in desire to implement a fast floating point arithmetic unit
  - "Divide" function is one of the complex ones, an algorithm that uses a lookup table is employed

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

- ✧ The Intel Pentium ™ Processor
  - Lookup table has 1066 relevant entries but, due to a programming error, only 1061 entries were loaded into PLA of Pentium's FPU
  - The PLA was not tested to see that the table had been copied correctly
  - A consequence of this mistake is that calculations involving a floating-point divide, for some by numerators and divisors, are likely to produce inaccurate answers
  - This defect has been estimated to have cost Intel upwards of $400 million

From: Morris et al., Object-Oriented Computer Systems Engineering, 1996

Catastrophic Computer Systems Development Examples

- ✧ Ariane Rocket Goes Boom (1996)
  - Ariane 5, Europe's newest unmanned rocket, was intentionally destroyed seconds after launch on its first flight
  - Also destroyed was its cargo of four scientific satellites to study how the Earth's magnetic field interacts with solar winds

From: from http://www.devtopics.com/20-famous-software-disasters-part-2/

Catastrophic Computer Systems Development Examples

- ✧ Ariane Rocket Goes Boom (1996)
  - 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
- ✧ See http://www.devtopics.com/20-famous-software-disasters/ and http://www5.in.tum.de/~huckle/bugse.html for more ...

From: from http://www.devtopics.com/20-famous-software-disasters/

Why ?

- The demand for software products are increasing exponentially
- Expectations from software products are increasing exponentially

Professional software development

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.

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

General issues that affect software

✧ 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

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

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

Software engineering diversity

- 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

- 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

Application types

- ✧ 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

Application types

✧ 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

Application types

- 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

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
  - 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

Internet software engineering

- 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

Internet software engineering

- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’
  - Users do not buy software buy pay according to use

Web software engineering

✧ Software reuse

- 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 software engineering

- Incremental and agile development
  - 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

Web software engineering

✧ Service-oriented systems

- Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services

Web software engineering

✧ Rich interfaces

- Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser

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

Software engineering ethics

- 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

Issues of professional responsibility

- 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

Issues of professional responsibility

✧ 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

ACM/IEEE Code of Ethics

- 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*

Rationale for the code of ethics

- *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

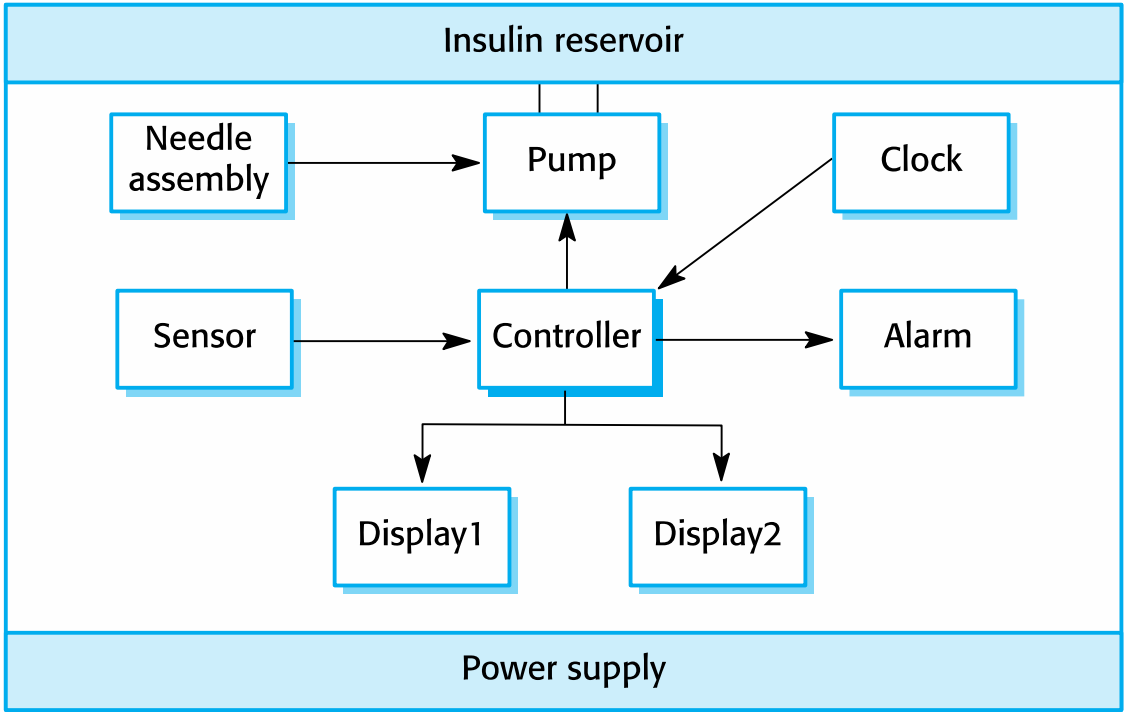
Case studies

- 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

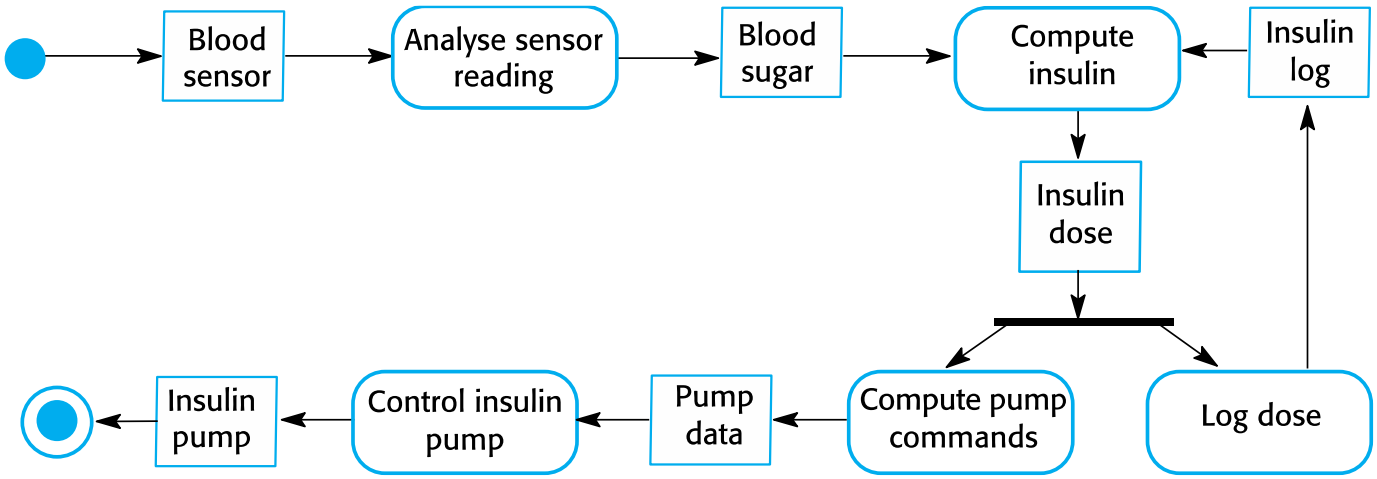
Insulin pump control system

- 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.

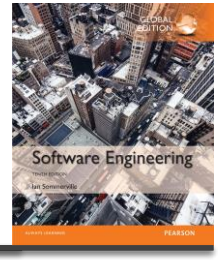
Mentcare

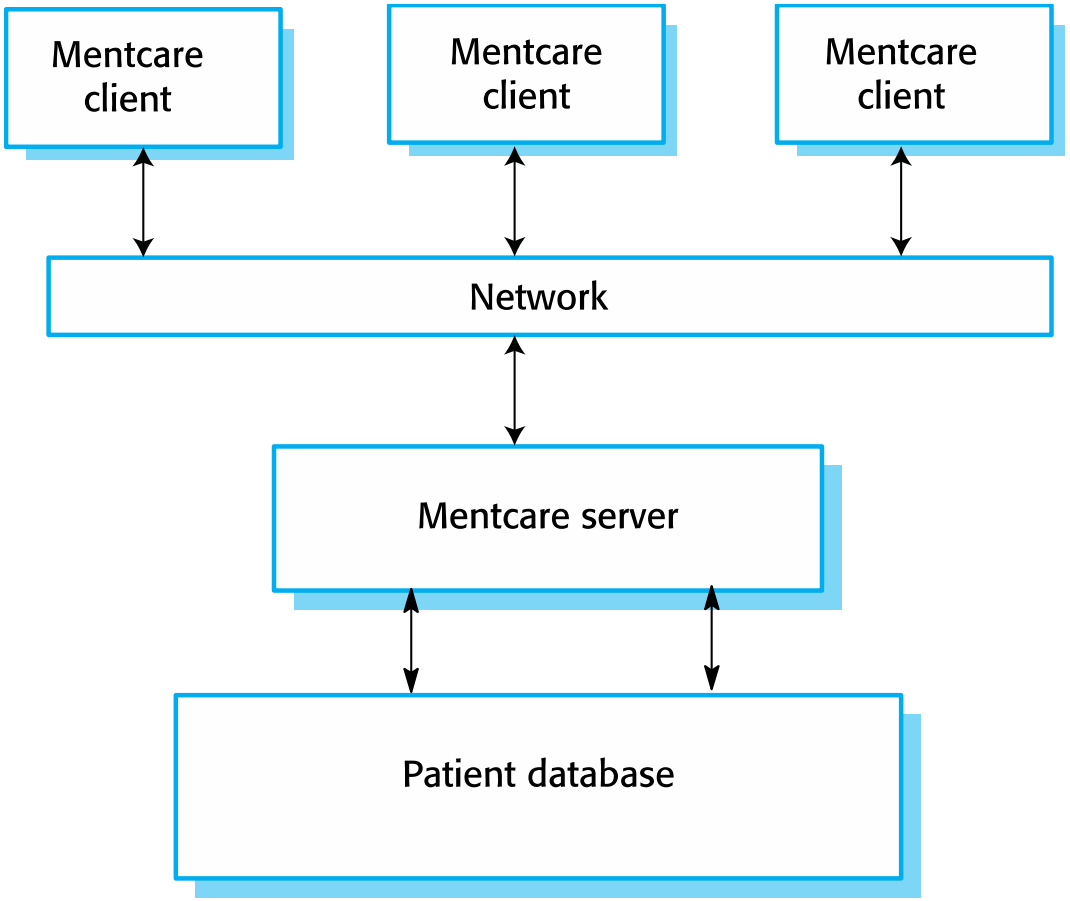
- Mentcare is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have 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.

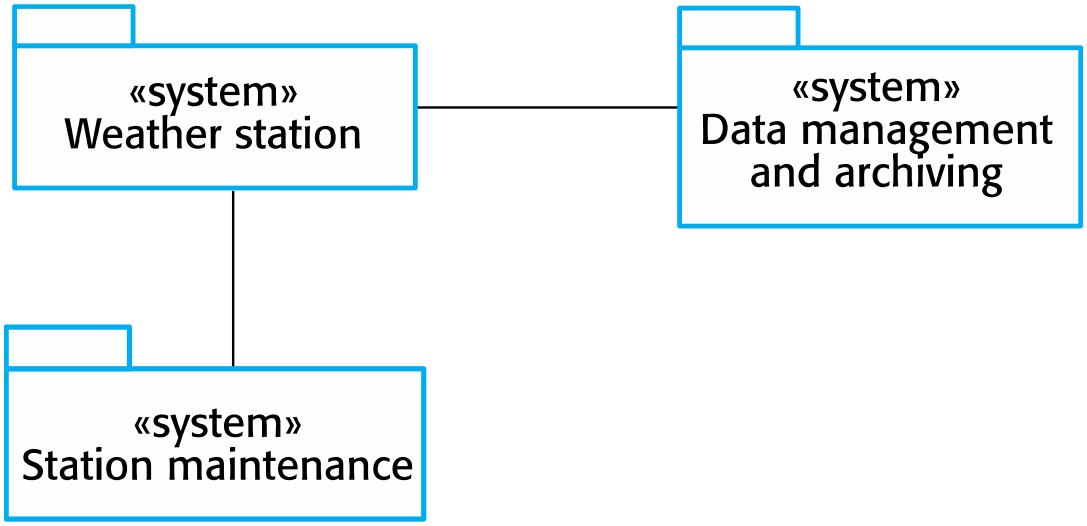
Mentcare system concerns

- 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.

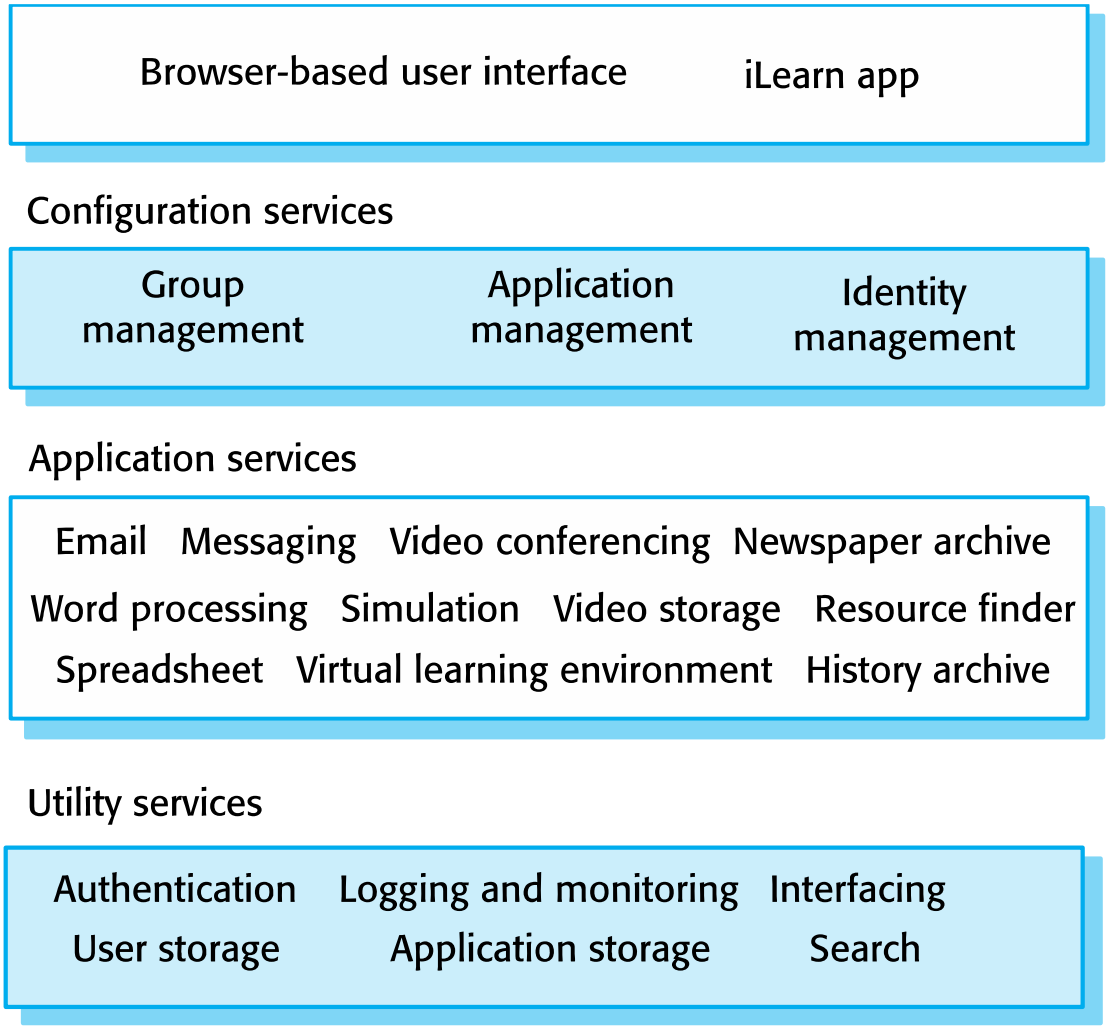
Service-oriented systems

- 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

Key points

- The principal approaches to process improvement are agile approaches, geared to reducing process overheads, and maturity-based approaches based on better process management and the use of good software engineering practice

Key points

- The SEI process maturity framework identifies maturity levels that essentially correspond to the use of good software engineering practice

Key points

- There are many different types of systems, 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

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