Software Engineering

What is software?

1. program + documentation

2. off-shelf software
   customized software

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.

Software Engineering

IEEE: “The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.”

What are the fundamental software engineering 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.

What the key challenges of software engineering nowadays?

Ariane 5
Rocket

```
int counter = MAX_INT;
counter++;
```

Mariner Bugs Out (1962)

- **Cost**
  - **$18,500,000**
- **Disaster**
  - **Mariner 1 rocket with a space probe headed for Venus diverted from its intended flight**
  - **Mission Control destroyed the rocket 293 seconds after liftoff**
- **Cause**
  - **A programmer incorrectly transcribed a formula into software**
  - **The software interpreted normal variations of velocity as anomalies**
  - **It issued faulty correction commands that sent the rocket off course**

Hartford Coliseum Collapse (1978)

- **Cost**
  - **$90,000,000**
- **Disaster**
  - **Steel-latticed roof collapsed under the weight of wet snow**
- **Cause**
  - **CAD software was used to design the coliseum**
  - **A programmer incorrectly assumed the steel roof supports would only face pure compression**
  - **One of the supports unexpectedly buckled from the snow**
  - **This set off a chain reaction**

Healthcare.gov

Development cost

- Initial estimate: 93.7 million
- Actual: 1.7 billion

Problems

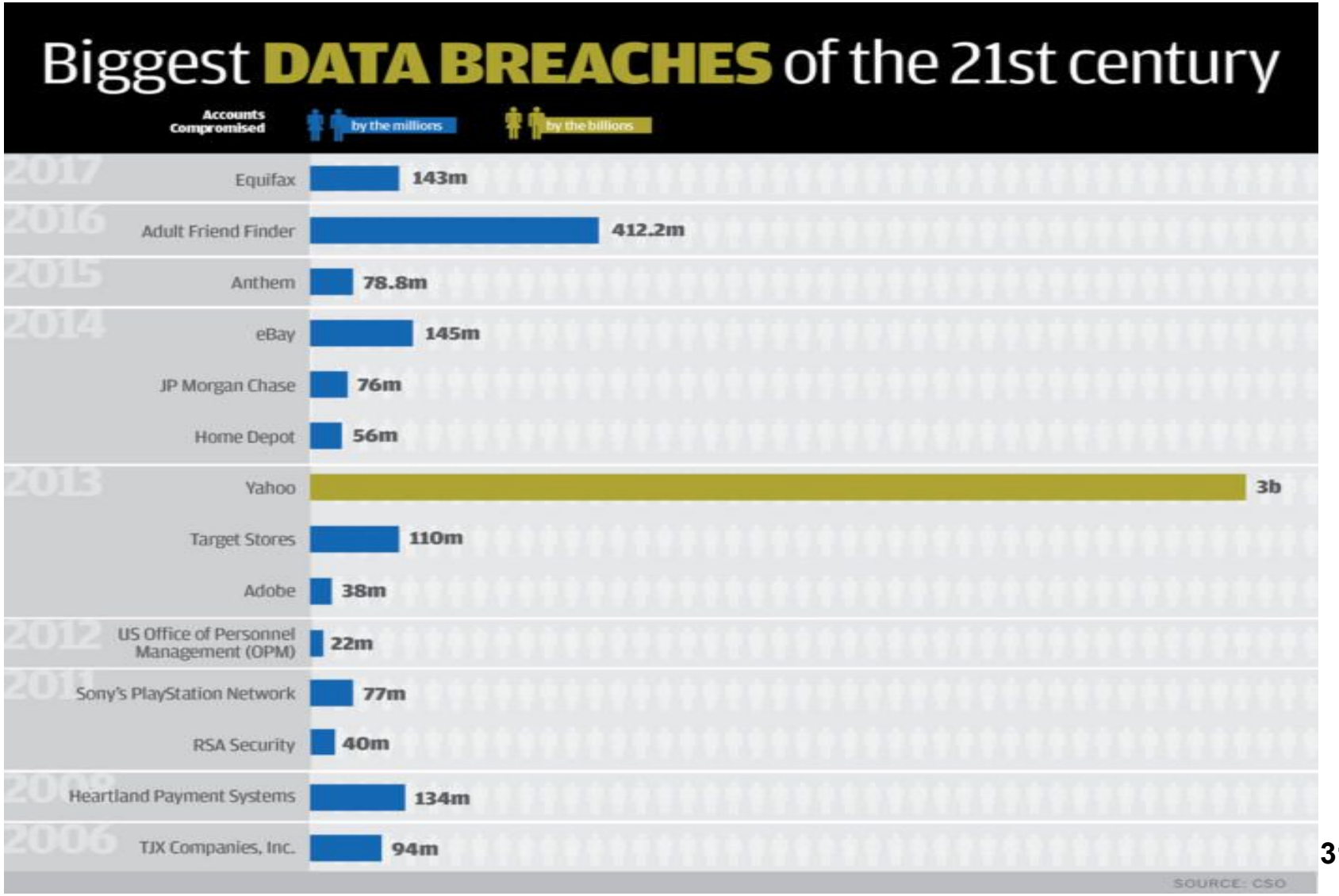
- Only 1% of all users in the first week could complete registration
- 1/10 applications had information garbled
- A political embarrassment

Cause

- Inadequate design to handle user load
- Testing was packed into last few weeks of project.

WannaCry Ransome attack

Biggest Data Breaches in 21st Century



What the key challenges of software engineering nowadays?

1. coping with diversities
2. demand for shortening delivery time
3. delivering trustworthy software

Software 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.

Software 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.

Software 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.

Why do projects fail?

- Estimation error. Projects that are estimated poorly are doomed from the start.
- Performance failure
  - Poor planning (bad methodology choice)
  - Incorrect and optimistic status reporting
  - Unrealistic schedule pressure
  - New and changing requirements during development
  - Inadequate quality control
  - Poor communication
  - New security challenges

Success with Software is Possible

- There are organizations that are succeeding with their software development efforts. Being successful with software means:
  - Having the ability to give accurate estimates that get more precise over time
  - Delivering on schedule and within budget
  - Being able to control software quality
  - Satisfied customers
  - High morale among staff members
  - High productivity

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.

Three Codes of Ethics

ACM Codes of Ethics and Professional Conduct:
http://www.acm.org/about/code-of-ethics

ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice:
http://www.acm.org/about/se-code

The Ten Commandments of Computer Ethics:
http://www.acm.org/about/se-code

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).

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

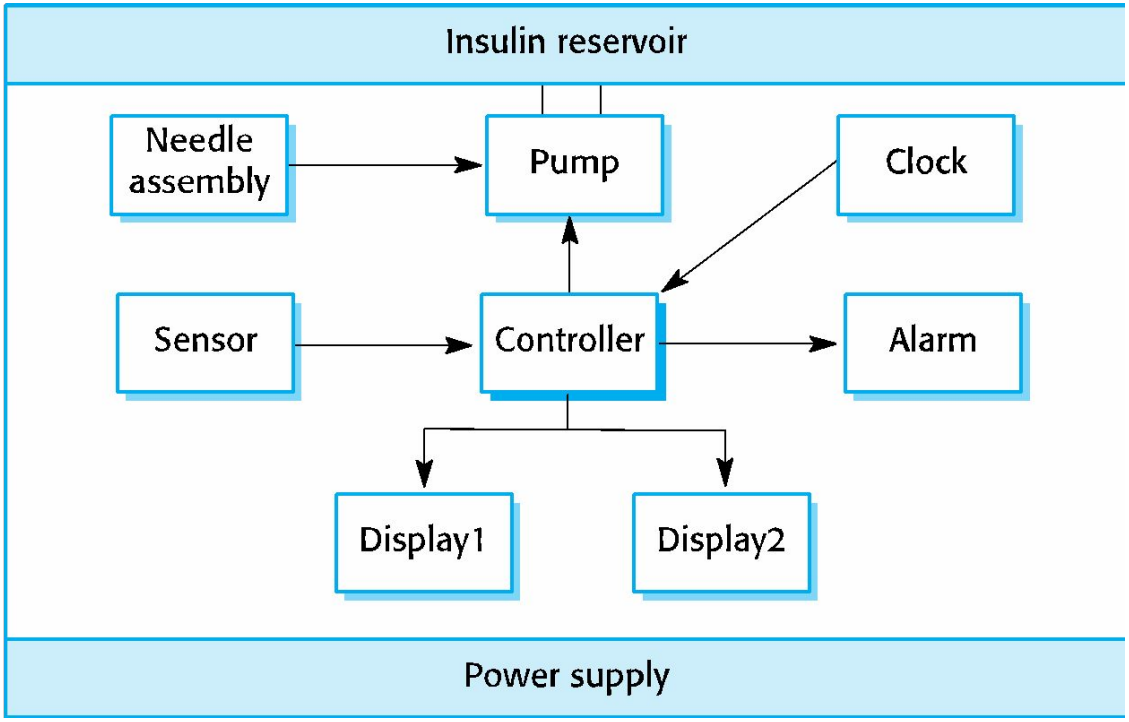
Case studies

- 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

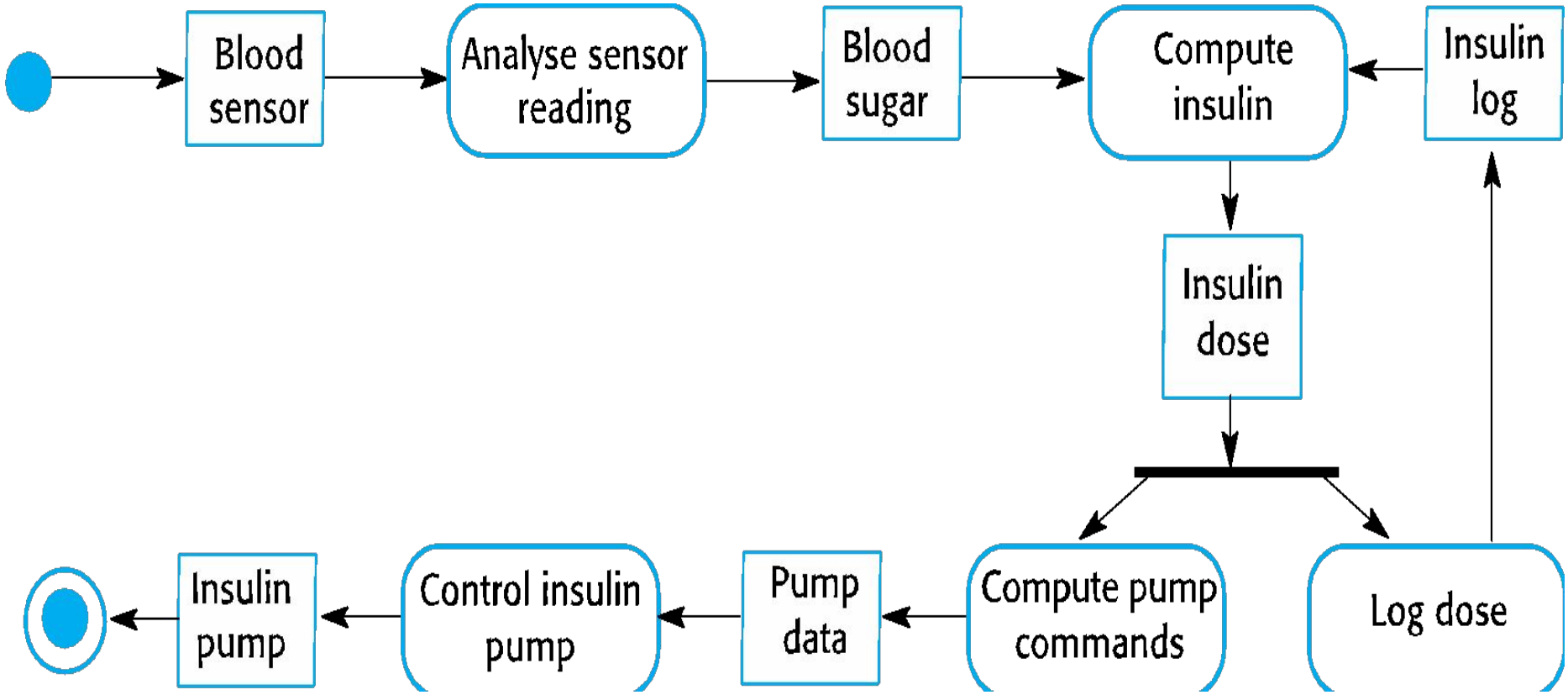
Insulin pump control system

- 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 and the time of last insulin injection
- 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

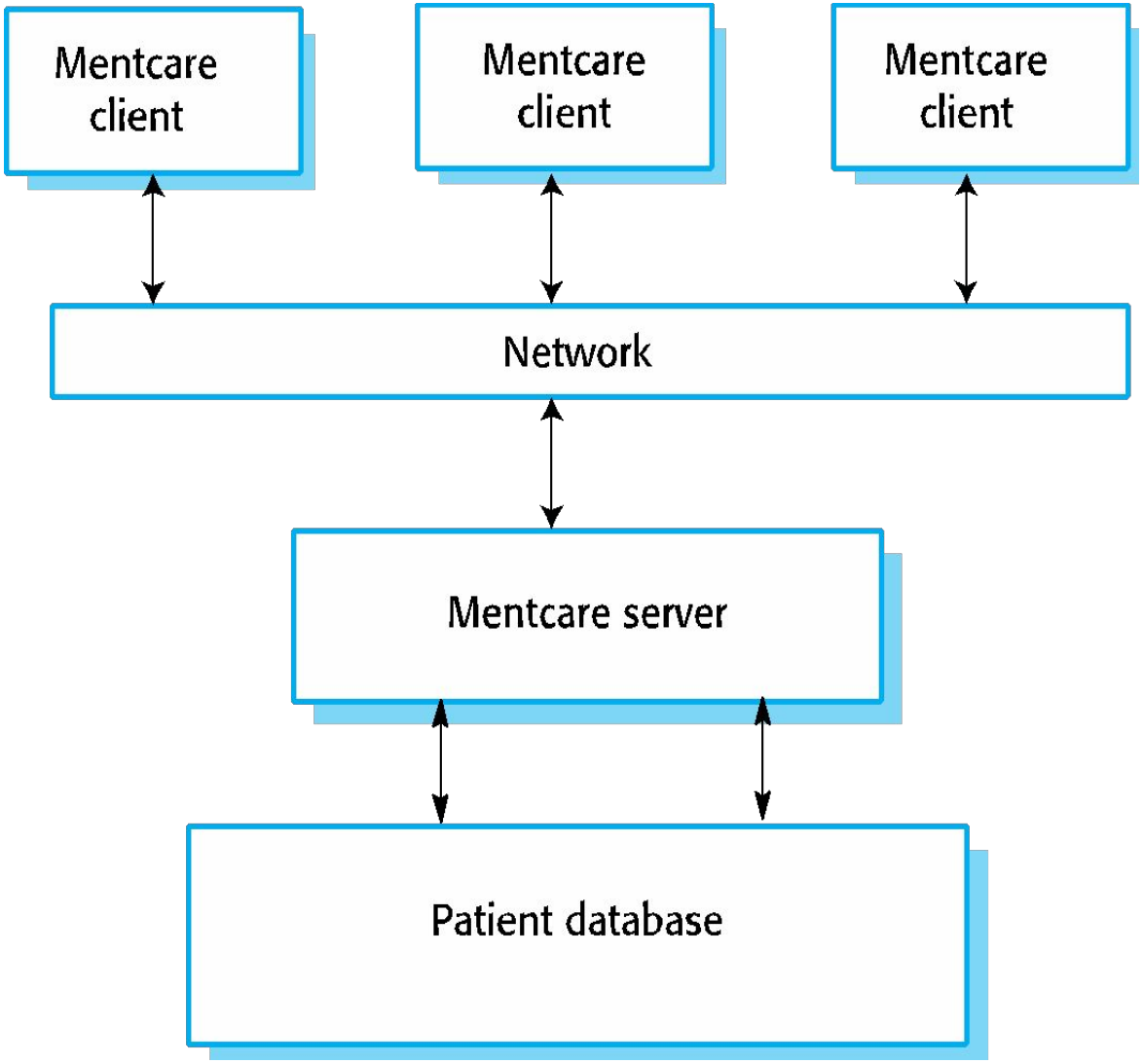
Mentcare

- 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.

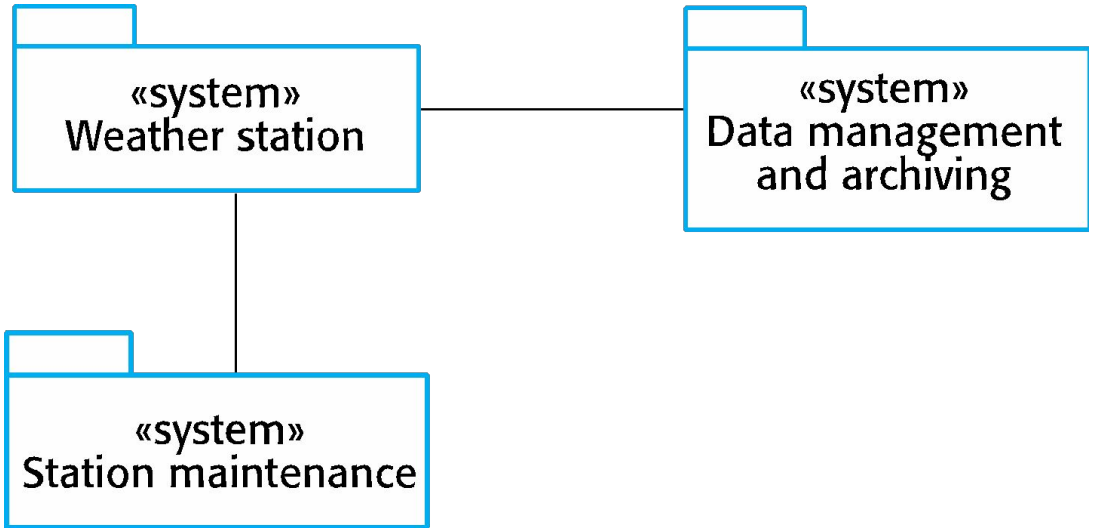
Mentcare system concerns

- 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

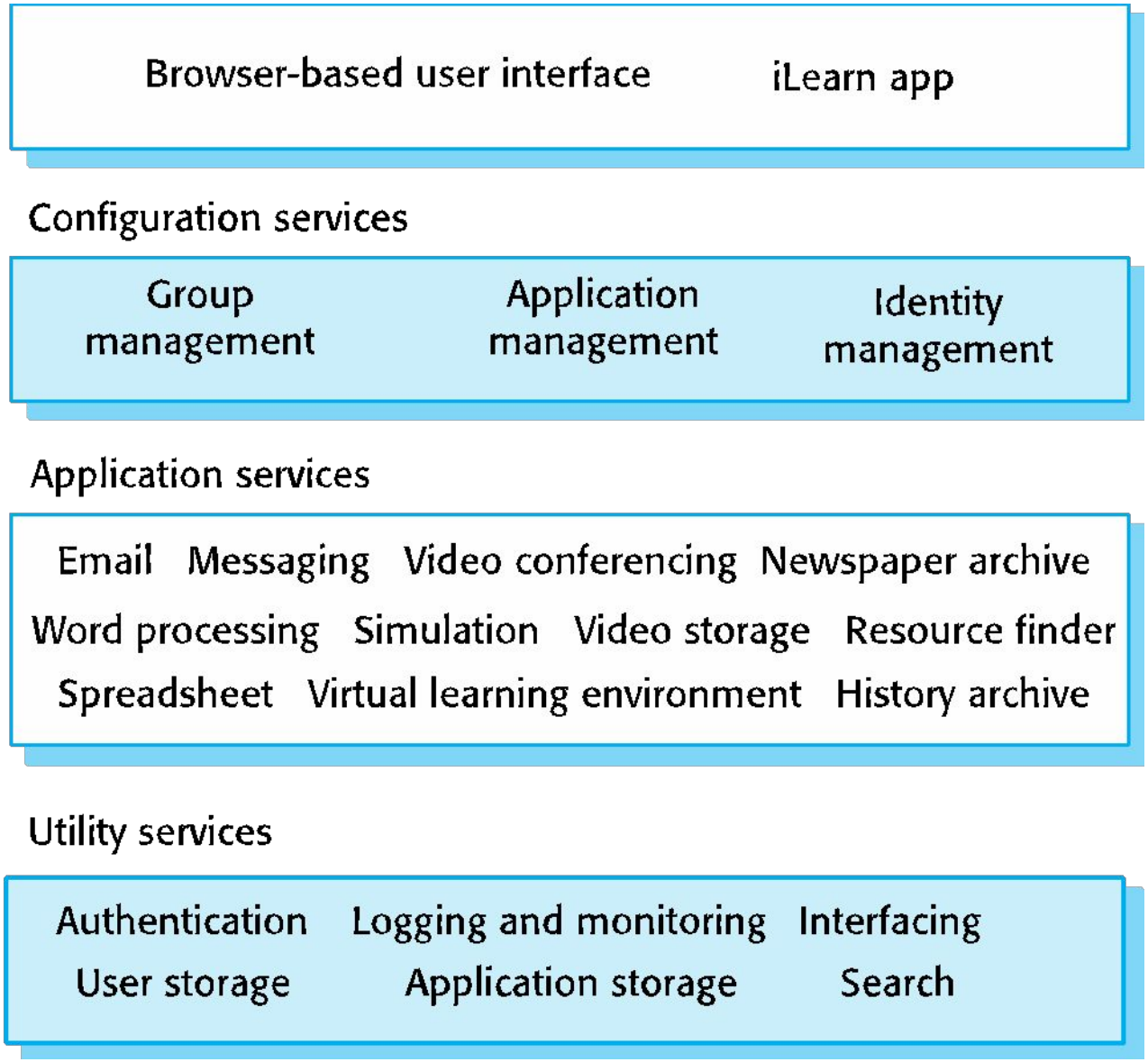
Service-oriented systems

- 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.

The difficulties in software development are rooted in the fundamental nature of software

- Intangibility – software is not far removed from pure thought stuff [Brooks].
- Complexity
  - Software is dynamic
  - Software automates complex processes (lower bound for complexity)
  - Software has a greater number of discrete states than any other human artifact and transitioning between these states is not a smooth and continuous process [tbd: find Fred Brooks ref]
  - Software complexity and project complexity grows more than linearly with size of product and size of team

Accidental Complexities

- The accidental complexities of software development are consequential to the tangible form of the solution, what you might call self-inflicted problems. If you choose Java as your implementation language, you will have to deal with the complications—large and small—that go along with programming in Java (e.g. how to convert a string to an int, how to read data from a file, etc.).
- Examples of accidental complexities:
  - Expressing conceptual constructs with a programming language
  - Learning to use IDE's such as Visual Studio and Eclipse
  - Learning to use configuration management tools such as SVN Git and ant gradle.
  - Testing the fidelity of the representation
  - Special tricks need to get C preprocessor macros to work correctly. For example, needing to wrap do { … }while(0) around statements in a macro definition in order to preserve the semantics when a semicolon is added after the macro use.

Essential Complexities

- Essential complexities are inherent in the problem. They can't be avoided.
- Examples:
  - Defining precisely what to build
  - Coming up with the conceptual construct of the solution
  - Validating the problem definition and solution construct

Which is better for your future?

- Specializing in tools that offer solutions to accidental complexities :
  - Java, C++, Objective-C
  - iPhone SDK, Android SDK
- Specializing in practices and methods used to deal with essential complexities:
  - Requirements engineering
  - Software design
  - Modeling (concepts and principles)
  - Usability engineering
- You can't be an expert in all areas. Each category of specialty has its own tradeoffs
- Distinguishing between essential and accidental complexities helps clarify the tradeoffs involved when making strategic decisions regarding what knowledge to pursue or skills to get.

Question:

Which skills you mostly like to have? Any reasons?