SE - Syllabus

Topics
Introduction Software, Process Models, Software Engineering
Reguirement Engineering - Phases, SRS, Analysis
Design & Architecture , UML Diagrams
Agile SDLC & Introduction to JIRA Tool
SE MCQ Test & Submission

Chapter 1- Introduction

Day 1 - Points Covered

- Software
- Difference between Software and Hardware
- Software Engineering, need , Importance
- Types of Software Products, Costs
- Software Engineering Principles/Ethics
- Process /Product/Project
- Software Process Framework
- Software Process Model & Case Studies
- Requirement Engineering

What is Software?

- ☐ Set of programs

- ☐ Data Structures

- ☐ Documentation :

What are the characteristics of Software?

 Logical rather than Physical

 Software is developed or engineered, it is not manufactured (Quality, People, Process, Cost)

 Software does not wear out (Bathtub Curve)

 Moving towards component based construction, however most of the software is custom built

Hardware vs. Software

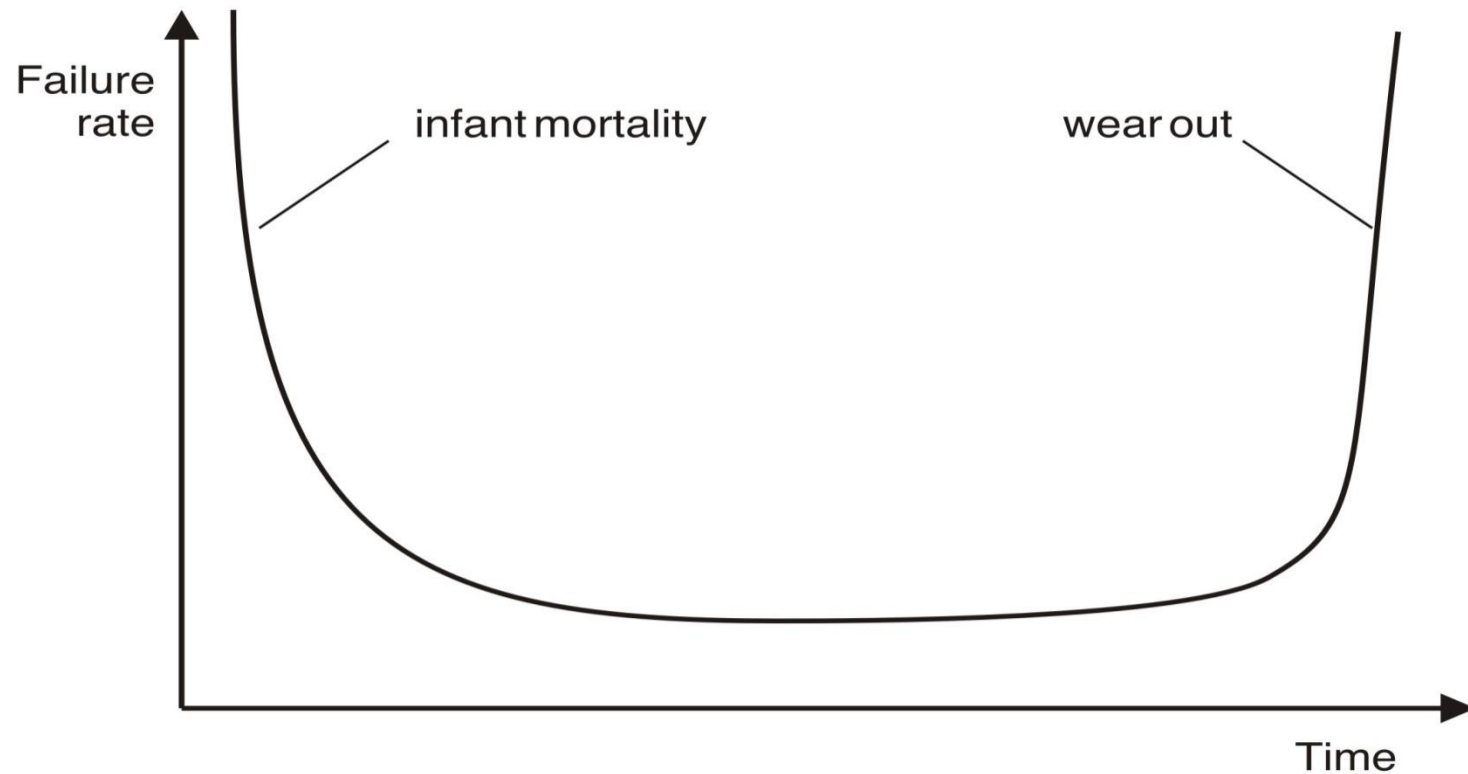| Hardware	Software
■ Manufactured ■ Wears out ■ Built using components ■ Relatively simple	■Developed/Engineered ■ Deteriorates ■ Custom built ■ Complex

Manufacturing vs. Development

- Once a hardware product has been manufactured, it is difficult or impossible to modify.
- In contrast, software products are routinely modified and upgraded.
- In hardware, hiring more people allows you to accomplish more work, but the same does not necessarily hold true in software engineering.
- Unlike hardware, software costs are concentrated in design rather than production.
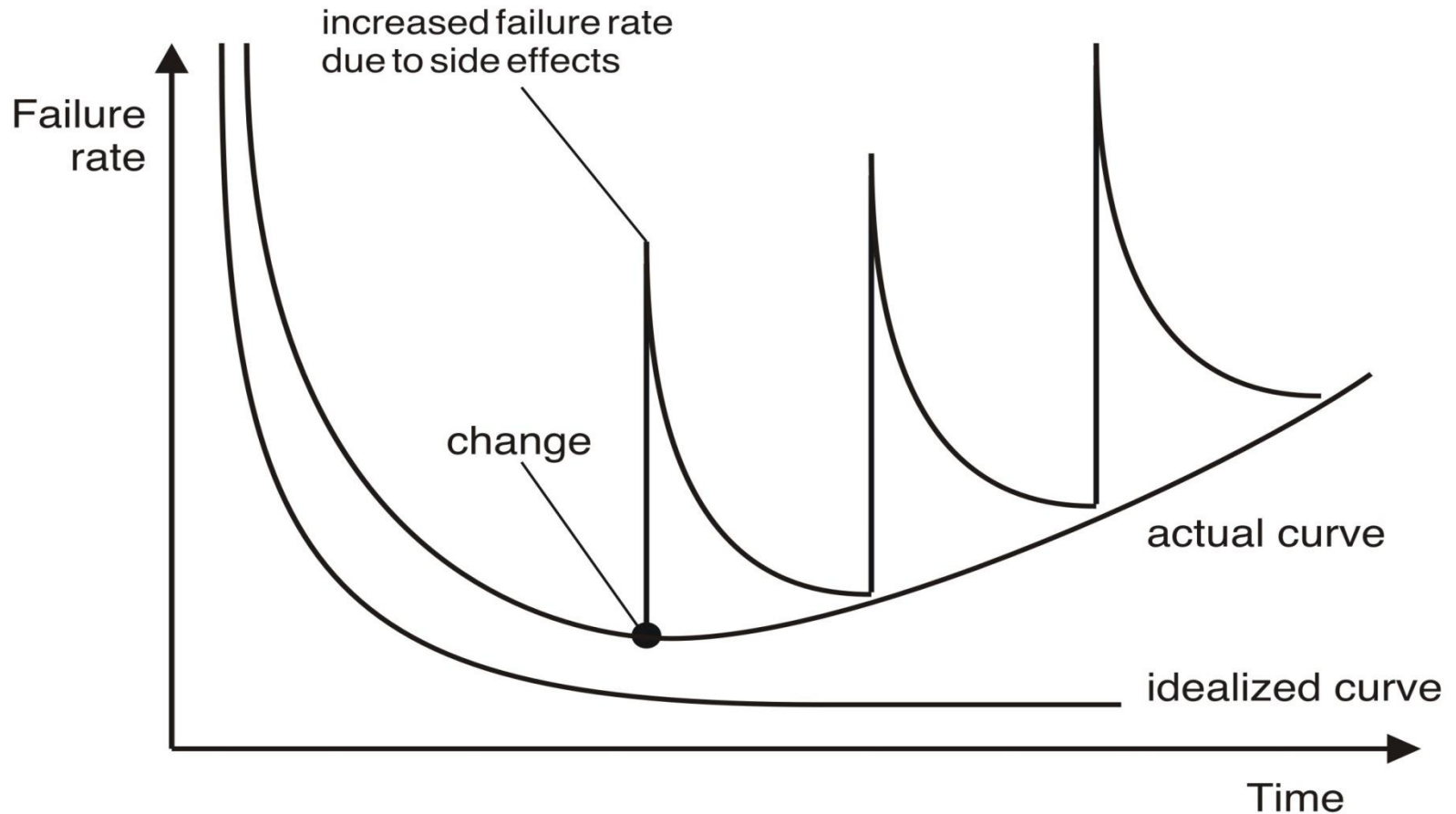
Wear vs. Deterioration

Hardware wears out over time

Wear vs. Deterioration

Software deteriorates over time

Background 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.

Software Myths

☐ Management Myths

☐ Customer Myths

☐ Practitioners Myth

Software Myths

❑ Management Myths

    ❑ We are already having a book that is full of Standards & Procedures for building software. It will provide my people everything they need to know.

    ❑ If we get behind schedule, we can add more developers & catch up.

    ❑ We can outsource the software project to a third party. I can just relax & let that firm build it.

Software Myths

Customer Myths

 A general statement of objectives is sufficient to begin writing programs. We can fill in the details later

 Project requirements continually change, but the change can be easily accommodated because software is flexible

Software Myths

Practitioners Myth

    ☐ Once we write the program and get it to work our job is done

    ☐ Until I get the program running, I have no way of accessing its quality

    ☐ The only deliverable work product for a successful project is the working program

    ☐ Software Engineering will make us create voluminous & unnecessary documentation & invariably slow us down

Software project failure

- *Increasing system complexity*

  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.

Software costs

- Software costs often dominate computer system costs.

- 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.

☐ **Nature of Software**

- ☐ Systems Software
- ☐ Applications Software
- ☐ Engineering / Scientific software
- ☐ Embedded Software
- ☐ Product Line Software
- ☐ Web Applications
- ☐ Artificial Intelligence Software
- ☐ Ubiquitous Software
- ☐ Net sourcing
- ☐ Open Source

Software products & Specification

- 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.
  - 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
  - Software that is commissioned by a specific customer to meet their own needs.
  - Examples – embedded control systems, air traffic control software, traffic monitoring systems.
  - The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.

Software engineering diversity

- Due to diversity in software there is no fix pattern /principle/policy or method to develop a software

- Selection depends on many factors like user, stake holders, business needs, feasibility , budget, resources etc

Application Types

- Stand-alone applications
- Interactive transaction-based applications
- Embedded control systems
- Batch processing systems
- Entertainment systems
- Systems for modeling and simulation
- Data collection systems
- Systems of systems

Application brief

- ## Stand-alone applications
  - Run on a local computer, such as a PC, not need to be connected to a network.

- ## Interactive transaction-based applications
  - E-commerce
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals.

- ## Embedded control systems
  - Software that control and manage hardware devices.

Application types

- Batch processing systems
  - Process large Businees data
- Entertainment systems
  - For Personal entertainment
- Systems for modeling and simulation
  - Developed by Scientist , Engineers

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.

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 engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

A Layered Technology

Software Engineering

tools

methods

process model
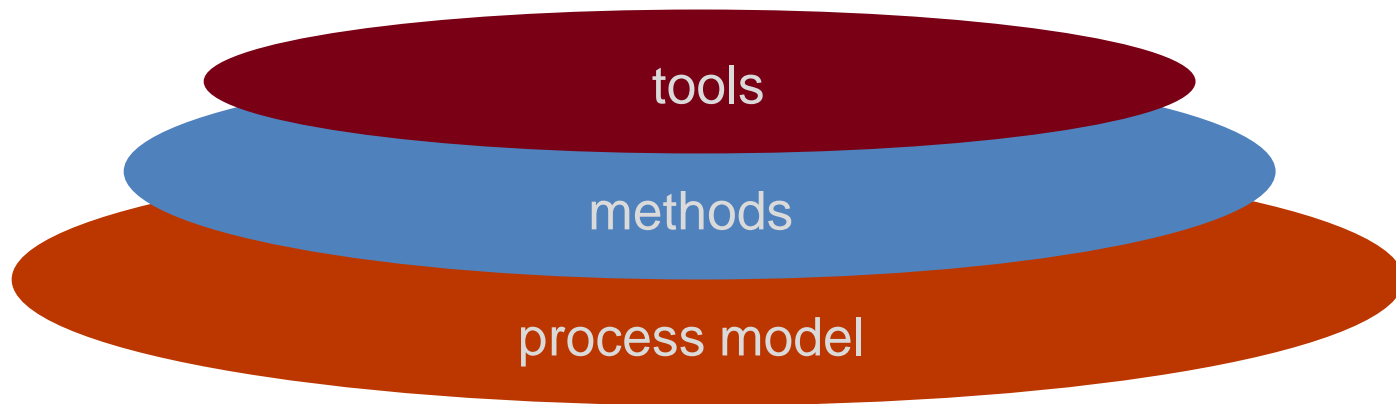
a "quality" focus

Software Engineering Layers

Quality Focus : Organizational commitment to quality (TQM, Six Sigma)

Process   : Defines a framework (Foundation for Software Engineering)

Methods   : "How to"'s for building software (Tasks)

Tools   : Automated or semi-automated support (Rational Rose, CASE tools)

Importance of software engineering

- Reliable & Trustworthy Software

- Economically cheap, can be used for Long run

- Society advancing in use of software

- Demanding Change in needs

Core Principles

1. The Reason it all Exists
2. Keep It Simple, Stupid (KISS!)
3. Maintain the Vision
4. What you Produce, others will Consume
5. Be Open to Future
6. Plan Ahead for Reuse
7. THINK

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.

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.

- 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.
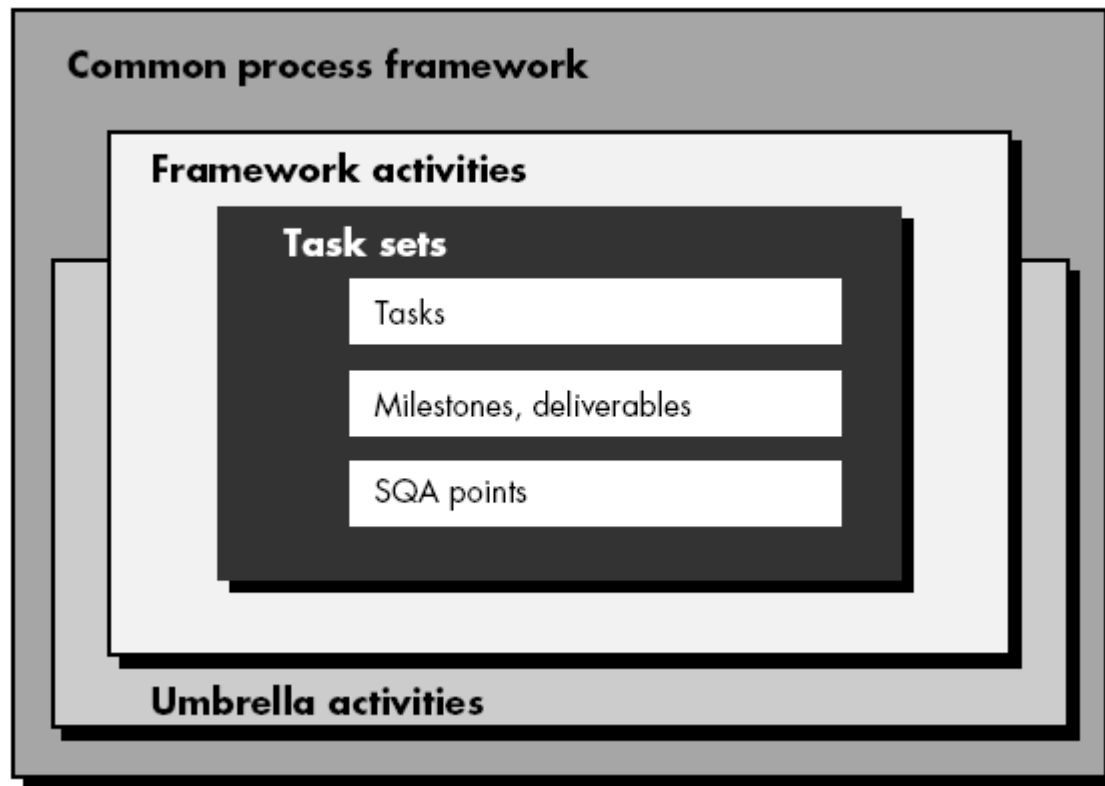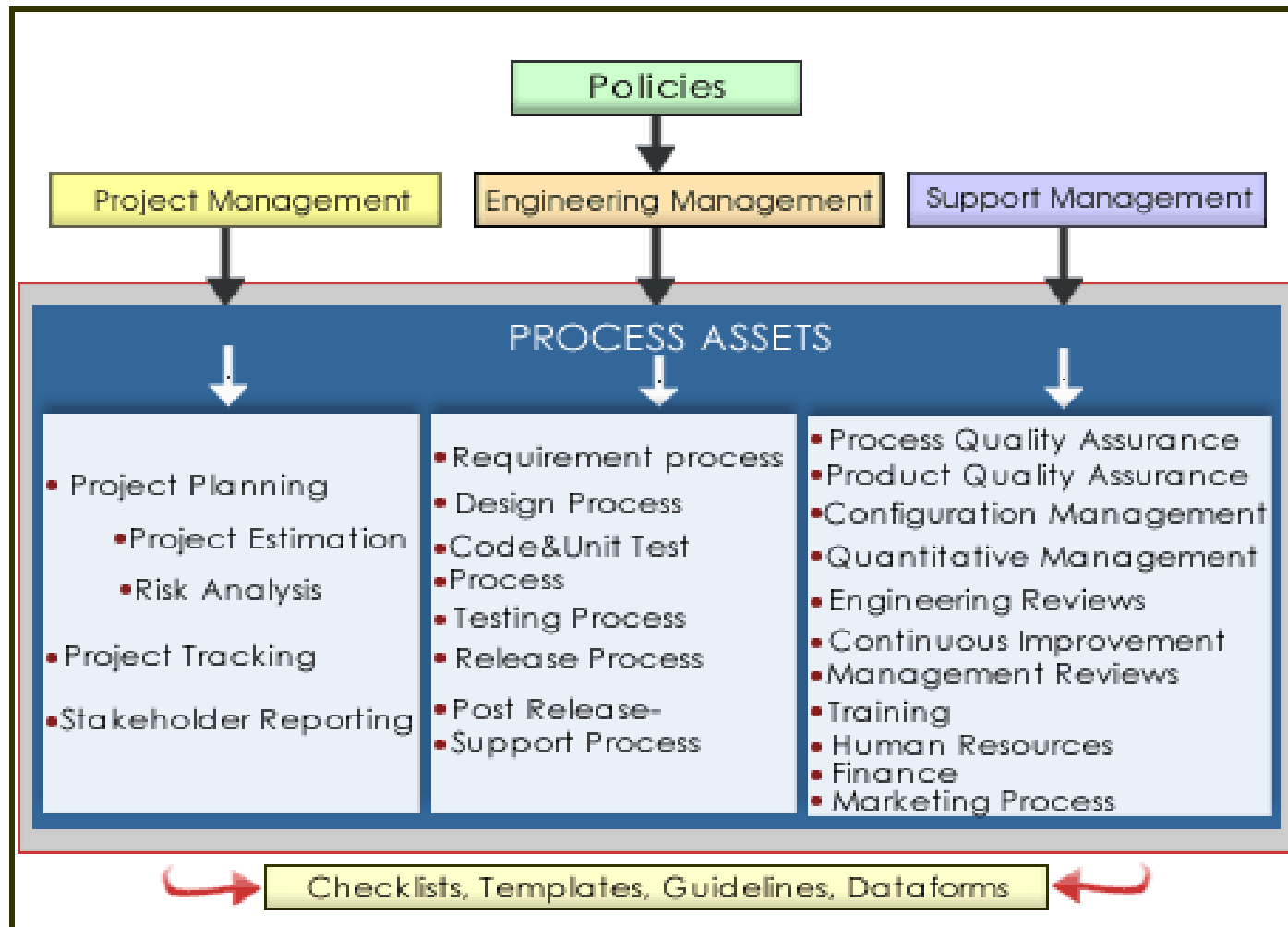
Product/Process/Project

- Product is Software (Outcome)
- Process (Technique)
- Project is (Workdone)

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.

Policies

Project Management | Engineering Management | Support Management

PROCESS ASSETS

- Project Planning
  - Project Estimation
  - Risk Analysis
- Project Tracking
- Stakeholder Reporting

- Requirement process
- Design Process
- Code&Unit Test
- Process
- Testing Process
- Release Process
- Post Release-
- Support Process

- Process Quality Assurance
- Product Quality Assurance
- Configuration Management
- Quantitative Management
- Engineering Reviews
- Continuous Improvement
- Management Reviews
- Training
- Human Resources
- Finance
- Marketing Process

Checklists, Templates, Guidelines, Dataforms

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 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 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.

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 (discussed in Chapter 19) allow application functionality to be accessed over the web.

- 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-based software engineering

- Web-based systems are complex distributed systems but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.

- The fundamental ideas of software engineering apply to web-based software in the same way that they apply to other types of software system.

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.

- 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.

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

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.
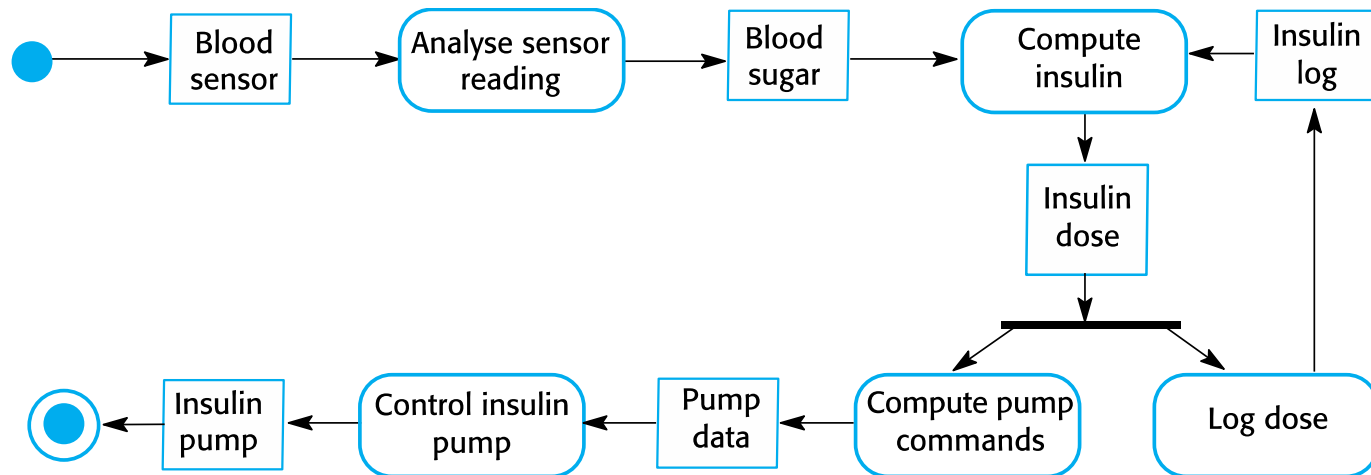
Insulin pump hardware architecture
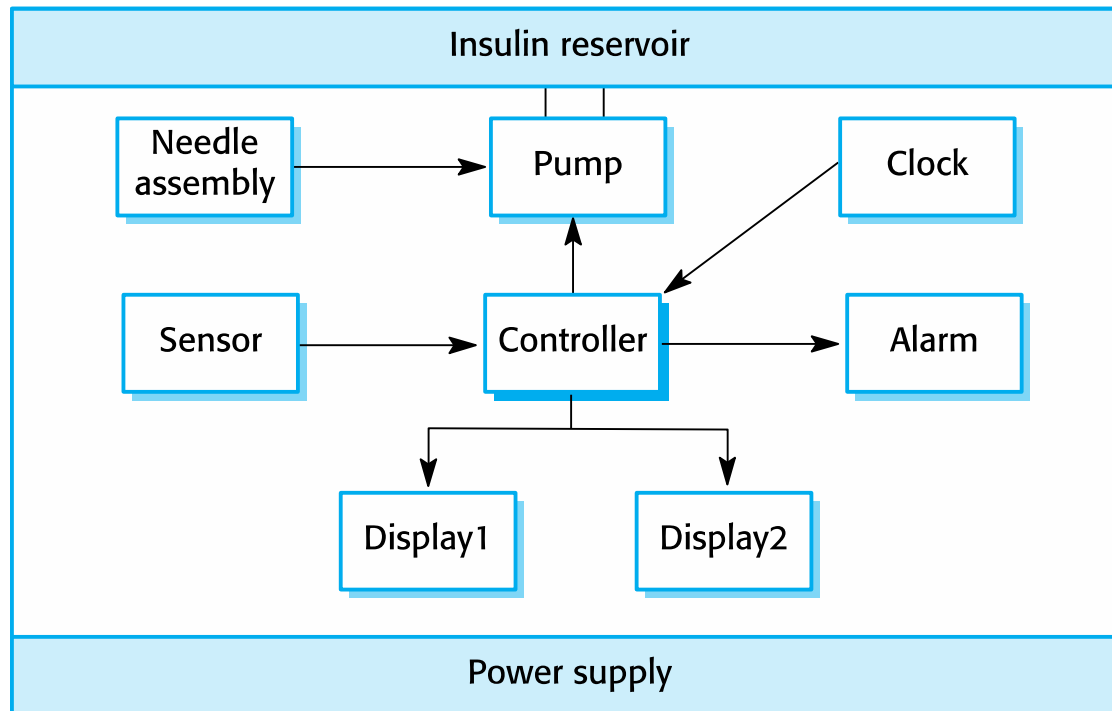
Activity model of the insulin pump

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.
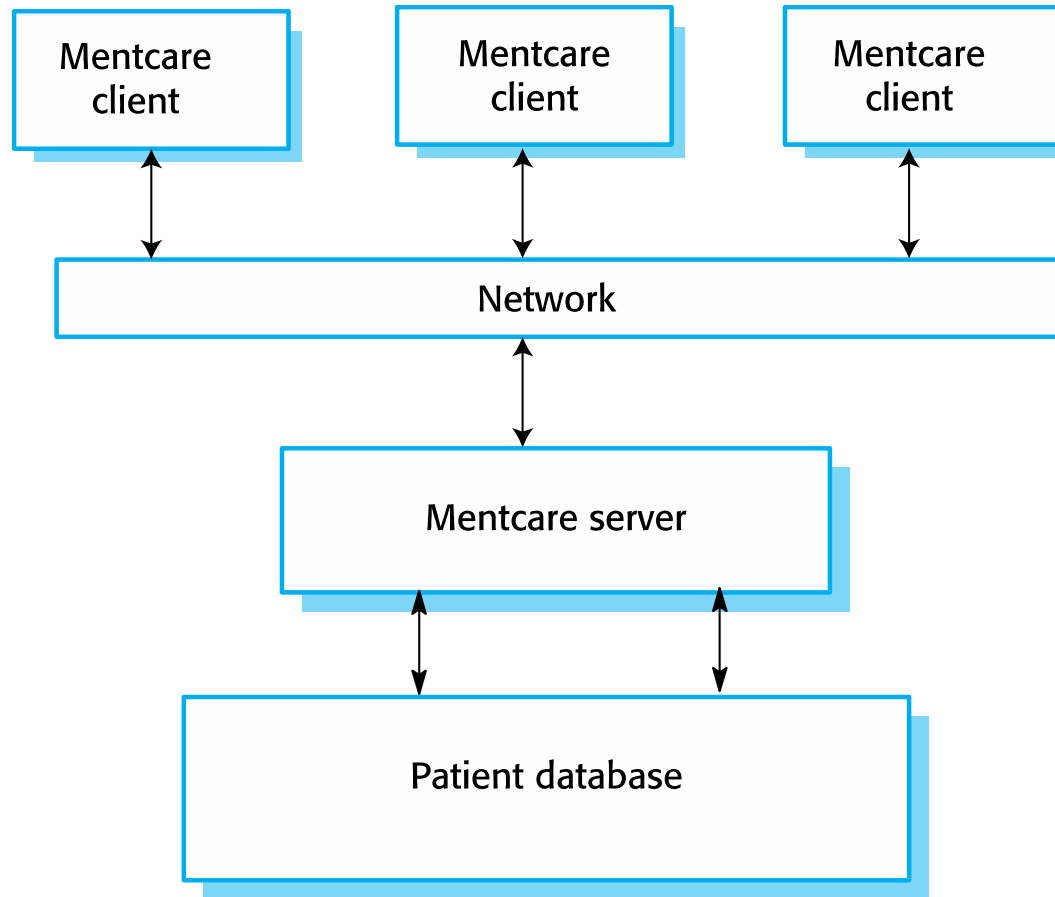
Mentcare goals

- 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
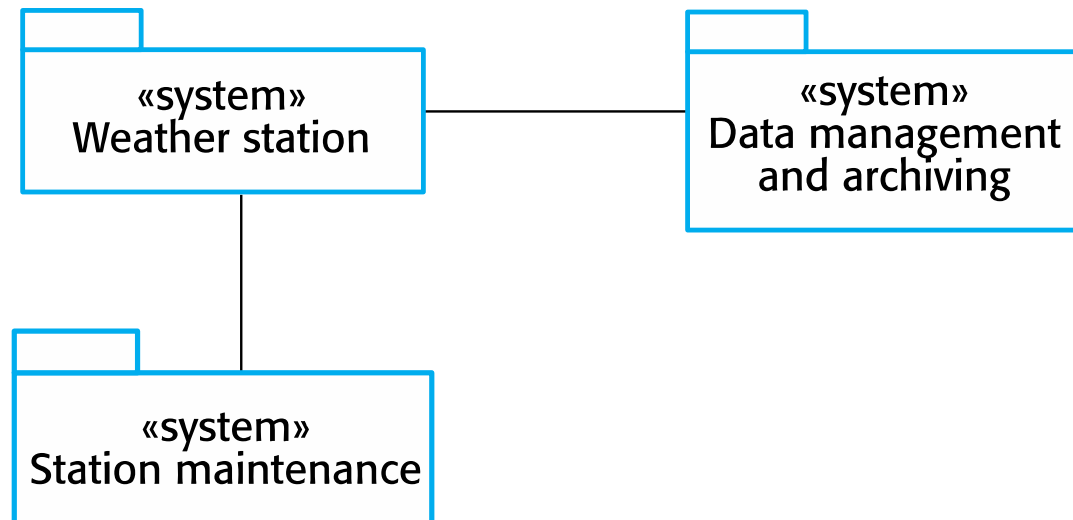
Weather information system

- 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 architecture

Browser-based user interface   iLearn app

Configuration services

Group management	Application management	Identity management

Application services

Email   Messaging   Video conferencing   Newspaper archive

Word processing   Simulation   Video storage   Resource finder

Spreadsheet   Virtual learning environment   History archive

Utility services

Authentication   Logging and monitoring   Interfacing

User storage   Application storage   Search

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.