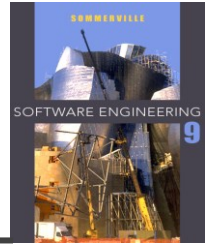


Faculty of Engineering & Information Technology

Software Engineering
110814230

Thaer Thaher
Thaer.Thaher@aaup.edu

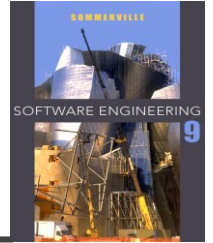
Fall 2021/2022



Chapter 1- Introduction

Lecture 1

Topics covered



✧ Professional software development

- What is meant by software engineering.

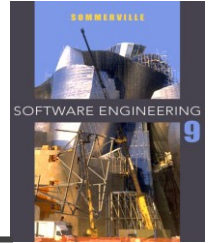
✧ Software engineering ethics

- A brief introduction to ethical issues that affect software engineering.

✧ Case studies

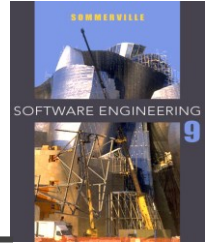
- An introduction to three examples that are used in later chapters in the book.

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.

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 engineering is intended to support professional software development, rather than individual programming. How??

Types of 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 (or *bespoke 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.



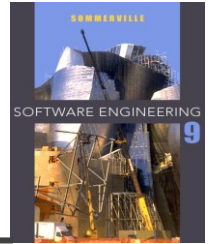
More and more systems are now being built with a generic product as a base, which is then adapted to suit the requirements of a customer (e.g. Enterprise Resource Planning (ERP) systems)

Let's discuss



- ✧ In Generic products Specifications and decision on change are made by
- ✧ In Customized products Specifications and decision on change are made by

Product specification



✧ Generic products

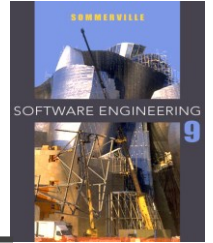
- The specification of what the software should do is owned by the **software developer** and decisions on software change are made by the **developer**.

✧ 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.

Professional software development

Frequently asked questions about SE



✧ 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.

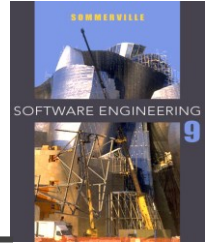
✧ What is engineering?

- Systematic and managed approaches used for solving problems

Many people think that software is simply another word for computer programs.



However, when we are talking about software engineering, **software is not just the programs themselves but also all associated documentation and configuration data that is required to make these programs operate correctly**



Software Engineering

✧ What is software engineering?

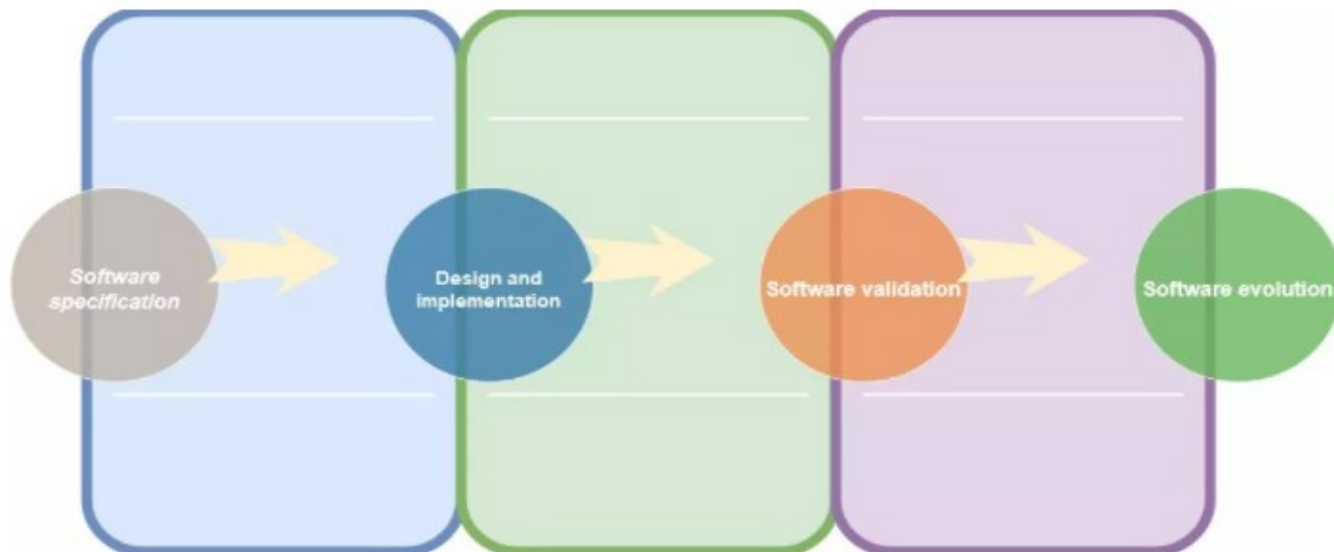
- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- A collection of techniques, methodologies, and tools that help with the production of high-quality software system developed with a given budget before a given deadline while change occurs.



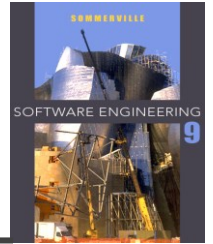
Frequently asked questions about SE

✧ What are the fundamental SE activities?

- Software specification
- Software development.
- Software validation.
- Software evolution.



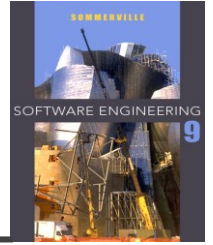
Frequently asked questions about SE



✧ What are the fundamental SE activities?

- **Software specification:** The main functionalities of the software and defines the restrictions in its operation.
- **Software development** (design and implementation): The software must be developed to meet its specification.
- **Software validation:** The software must be validated to assure that it performs what the customer requires.
- **Software evolution:** The software must be able to adapt to the customer's needs (changes).

Frequently asked questions about SE



✧ What are the key challenges facing software engineering?

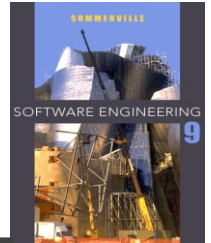
- **Coping with increasing diversity** (different devices and platforms.....)
- **Developing trustworthy software:**
- **demands for reduced delivery times**



Continuous Change

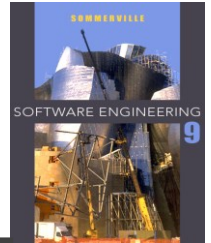


Frequently asked questions about software engineering



Question	Answer
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 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



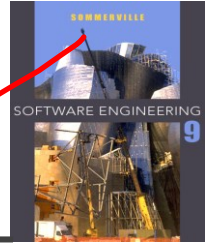
Question	Answer
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have 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 <u>distributed service-based systems</u> . Web-based systems development has led to important advances in programming languages and software reuse.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers . This is a critical attribute (why??) 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.

✓ Note

*The specific set of attributes that you might expect from a software system obviously depends on its application. Therefore, a **banking system must be secure**, an **interactive game must be responsive**, a **telephone switching system must be reliable**,*



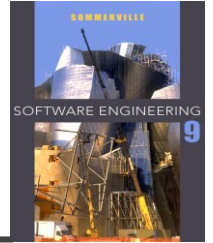
✧ What is Reliability?

- Reliability is the probability that a system performs correctly during a specific time duration. During this correct operation
 - No repair is required or performed
 - The system adequately follows the defined performance specifications



How to improve reliability?

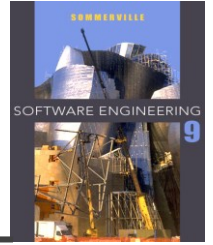
Software engineering



- ✧ **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 concerned with the technical process of development. Also **project management (why??)** and the development of tools, methods etc. to support software production.

SE is about getting results of the required quality within the **schedule and budget**

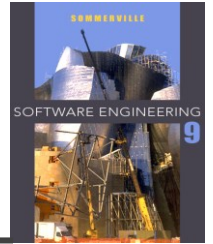
Importance of software engineering



SE is important for two reasons:

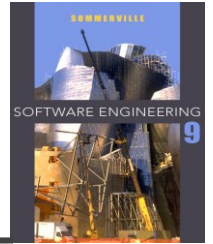
- ✧ More and more, individuals and society rely on advanced software systems. **We need to be able to produce reliable and trustworthy systems economically and quickly.**
- ✧ **It is usually cheaper, in the long run, to use software engineering methods** and techniques for software systems rather than 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



- ✧ The systematic approach that is used in software engineering is sometimes called a **software process**
- ✧ A software process is a sequence of activities that leads to the production of a software product
- ✧ **There are four fundamental activities that are common to all software processes. These activities are:**
 - Software specification
 - Software development.
 - Software validation.
 - Software evolution.

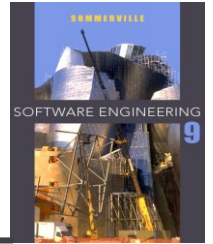
Software process activities



✧ What are the fundamental SE activities?

- **Software specification:** The main functionalities of the software and defines the restrictions (constraints) in its operation.
- **Software development** (design and implementation): The software must be designed and programmed to meet its specification.
- **Software validation:** The software must be validated/checked to assure that it performs what the customer requires.
- **Software evolution:** The software must be able to adapt to the customer's needs (changes). (the software is modified to reflect changing customer and market requirements)

Software process activities

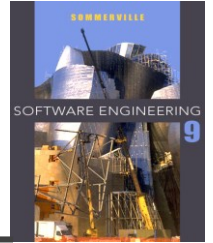


Different types of systems need different development processes. For example, **real-time software** in an aircraft has to be completely specified before development begins. In **e-commerce systems**, the specification and the program are usually developed together. Consequently, these generic **activities may be organized in different ways** and described at different levels of detail depending on the type of software being developed. I describe software processes in more detail in Chapter 2.



There is no universal software engineering method or technique that is applicable for all of types of software

General issues that affect most software (challenges)



✧ Heterogeneity

- Increasingly, systems are required to operate as distributed systems across networks that include **different types of computer and mobile devices**.

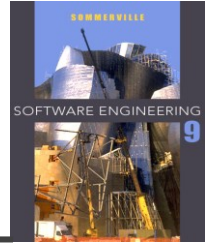
✧ Business and social change

- Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to **rapidly develop** new software.

✧ Security and trust

- As software is intertwined with all aspects of our lives, it is essential that we can **trust that software**.

Software engineering diversity

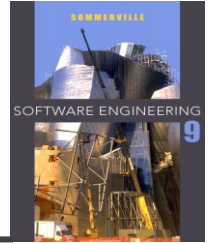


- ✧ There are many different types of software system and **there is no universal set of software techniques that is applicable to all of these.**
- ✧ The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.



The most significant factor in determining which software engineering methods and techniques are most important

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

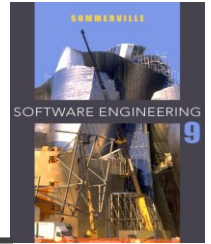
- Applications that **execute on a remote computer** and are accessed by users from their own PCs or terminals. These include **web applications** such as e-commerce applications, mail, photo sharing.... etc.

✧ 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.

example: software in a microwave oven to control the cooking process

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.

Examples: periodic billing systems, such as phone billing systems, and salary payment systems

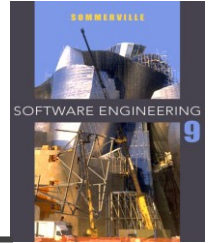
✧ Entertainment systems

- 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. These are often computationally intensive and require high-performance parallel systems for execution

Application types



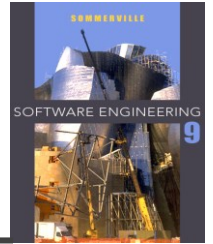
✧ Data collection systems

- 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.

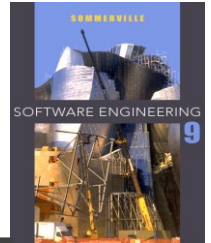
Application types



- ✧ You use different software engineering techniques for each type of system because the software has quite different characteristics.
 - For example, an **embedded control system** in an automobile is safety-critical and is burned into ROM when installed in the vehicle. It is therefore very expensive to change. Such a system needs very extensive verification and validation so that the chances of having to recall cars after sale to fix software problems are minimized.
 - For a **web-based system**, an approach based on iterative development and delivery may be appropriate, with the system being composed of reusable components.

Nevertheless, there are software engineering fundamentals that apply to all types of software system

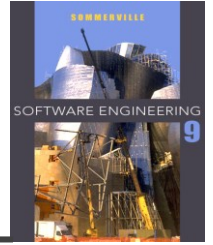
Software engineering fundamentals



- ✧ 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. (behave without failures, should be available for use when it is required, safe, secure against external attack, should not waste resources)
 - 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.**

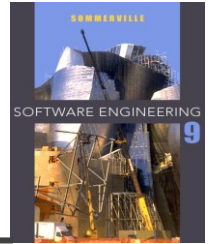
What differences has the web made to software engineering??

Software engineering and the web



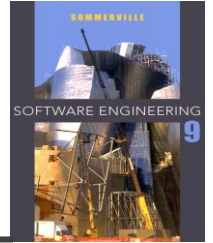
- ✧ The Web is now a platform for running application and organizations are increasingly developing **web-based systems rather than local systems**.
- ✧ **Web services** (discussed in Chapter 19) allow application functionality to be accessed over the web.
 - Web services are software components that deliver specific, useful functionality and which are accessed over the Web
 - Applications are constructed by integrating these web services, which may be provided by different companies.
- ✧ **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.**

Web software engineering



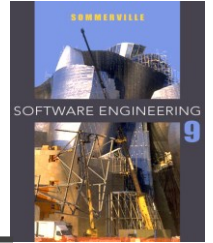
- ✧ **Software reuse** is the dominant approach for constructing web-based systems.
 - When building these systems, **you think about how you can assemble them from pre-existing software components and systems.**
- ✧ Web-based systems should be developed and delivered **incrementally (why??)**.
 - It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

Web-based software engineering



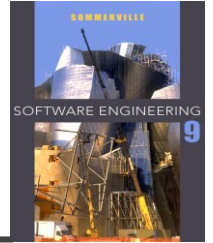
- ✧ **Web-based systems are complex distributed systems** but the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.
- ✧ The fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.

Key points

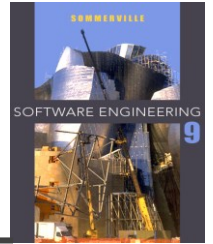


- ✧ Software engineering is an engineering discipline that is concerned with all aspects of software production.
- ✧ Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- ✧ The high-level activities of specification, development, validation 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



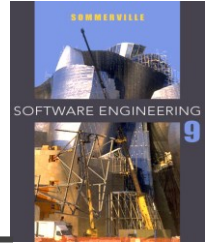
- ✧ There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- ✧ The fundamental ideas of software engineering are applicable to all types of software system.



Chapter 1- Introduction

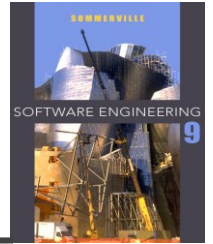
Lecture 2

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



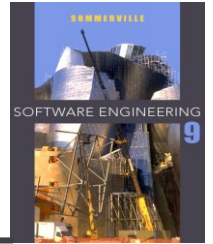
✧ Confidentiality

- Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- e.g., Every company has some knowledge and can identify the individuals and groups that might have access to a particular set of information. The members of such groups share the responsibility of maintaining confidentiality.

✧ 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.

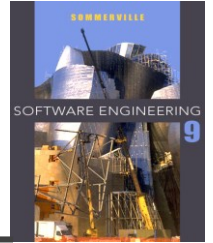
✧ Computer misuse

- 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).



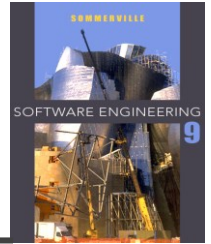
White hat (White hat hacker)?

ACM/IEEE Code of Ethics



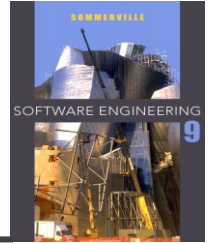
- ✧ The professional societies in the US have cooperated to produce a code of ethical practice.
- ✧ Members of these organisations sign up to the code of practice when they join.
- ✧ The Code contains **eight Principles** related to the behaviour of and decisions made by professional software engineers, 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

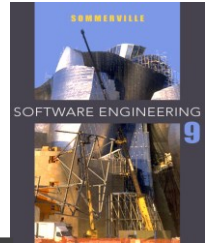
ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

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.

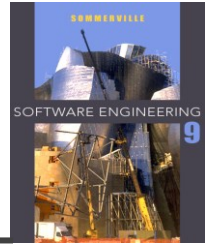
Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the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



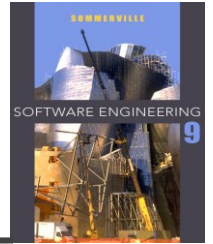
- ✧ Disagreement in principle with the policies of senior management.
 - Reaction depends on the particular individuals
 - argue? resign from their job??
- ✧ Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system. (**How should you react?**)
 - The problem here is that there are no absolutes when it comes to safety
 - Is the engineer's responsibility to maintain confidentiality or to alert the customer or publicize, in some way, that the delivered system may be unsafe?
- ✧ Participation in the development of military weapons systems or nuclear systems.

Case studies

- ✧ A personal insulin pump
- ✧ A mental health case patient management system
- ✧ A wilderness weather station

“The reason why I have not used a single case study is that one of the key messages in this book is that software engineering practice depends on the type of systems being produced”

Case studies



✧ A personal insulin pump

- An **embedded system** in an insulin pump used by diabetics to maintain blood glucose control.
- Issues include responsiveness, physical size, power management

✧ A mental health case patient management system

- An **information system** used to maintain records of people receiving care for mental health problems.
- Issues in information systems include security, usability, privacy, and maintaining data integrity

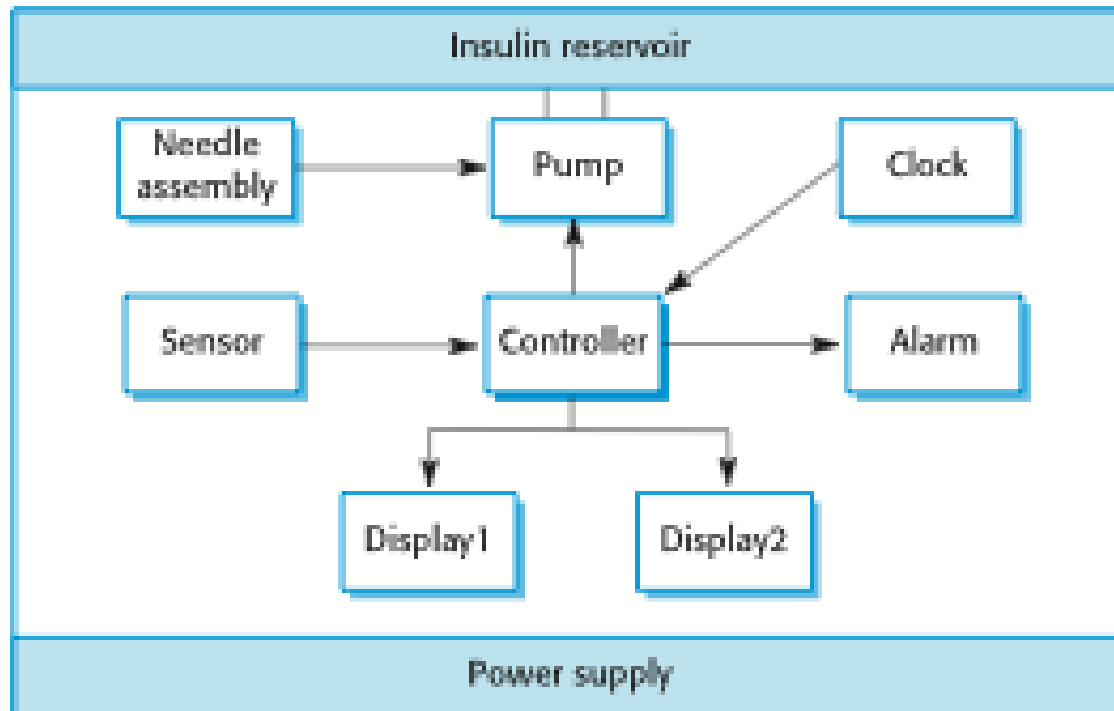
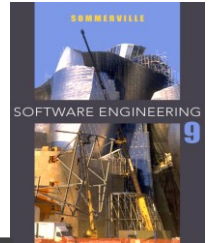
✧ A wilderness weather station (**sensor based data collection**)

- A data collection system that collects data about weather conditions in remote areas.
- The key requirements of such systems are reliability and maintainability

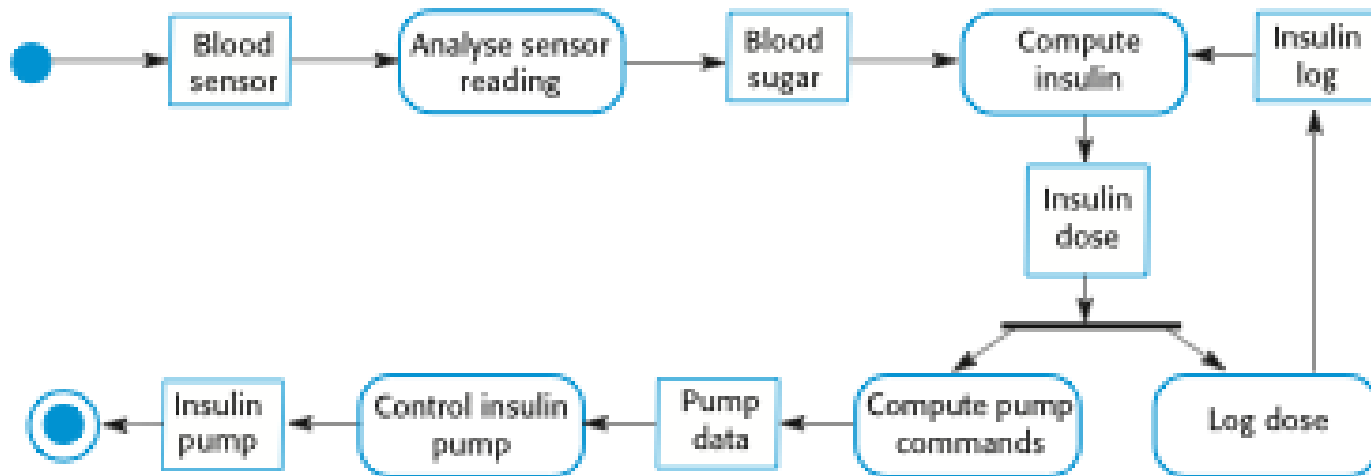
Insulin pump control system

- ✧ 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



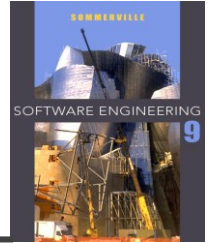
Activity model of the insulin pump



Essential high-level requirements

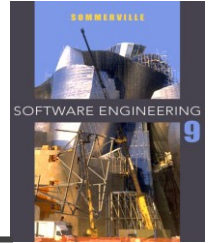
- ✧ The system shall be available to deliver insulin when required.
- ✧ The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- ✧ The system must therefore be designed and implemented to ensure that the system always meets these requirements.

A patient information system for mental health care



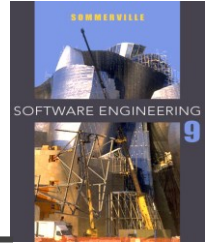
- ✧ A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- ✧ Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- ✧ To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

MHC-PMS



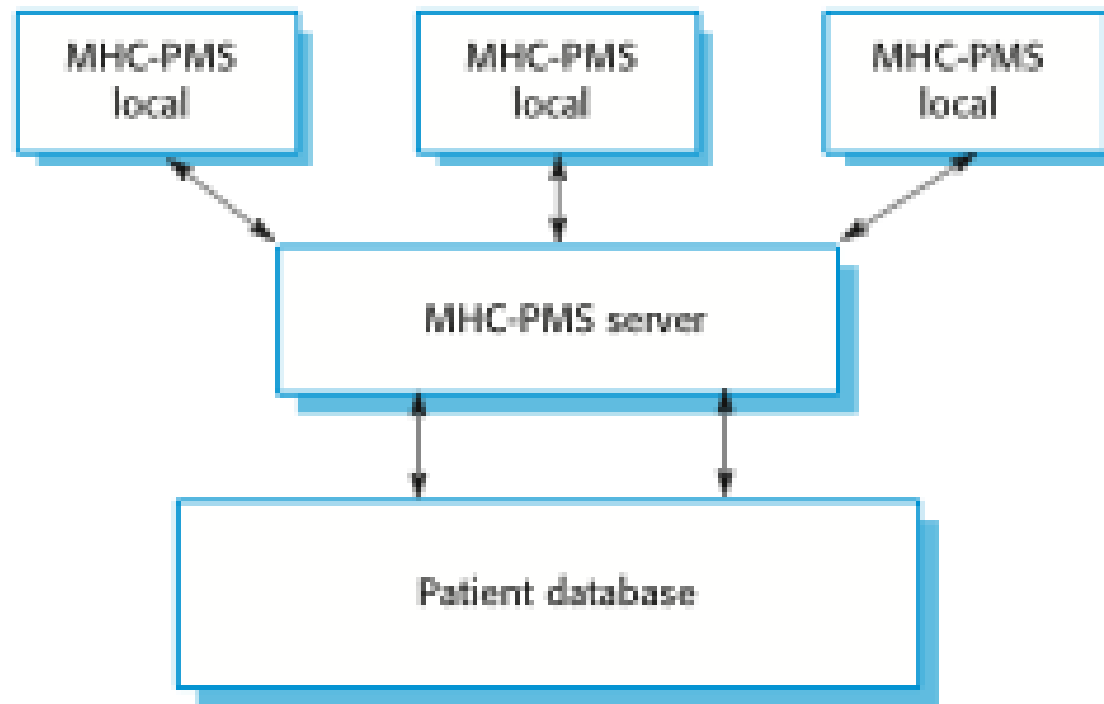
- ✧ The MHC-PMS (Mental Health Care-Patient Management System) is an information system that is intended for use in clinics.
- ✧ It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- ✧ When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

MHC-PMS goals

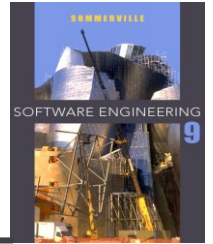


- ✧ To generate management information that allows health service managers to assess performance against local and government targets.
- ✧ To provide medical staff with timely information to support the treatment of patients.

The organization of the MHC-PMS



MHC-PMS key features



✧ Individual care management

- Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.

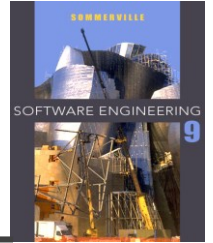
✧ Patient monitoring

- The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.

✧ Administrative reporting

- The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

MHC-PMS concerns



✧ Privacy

- It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.

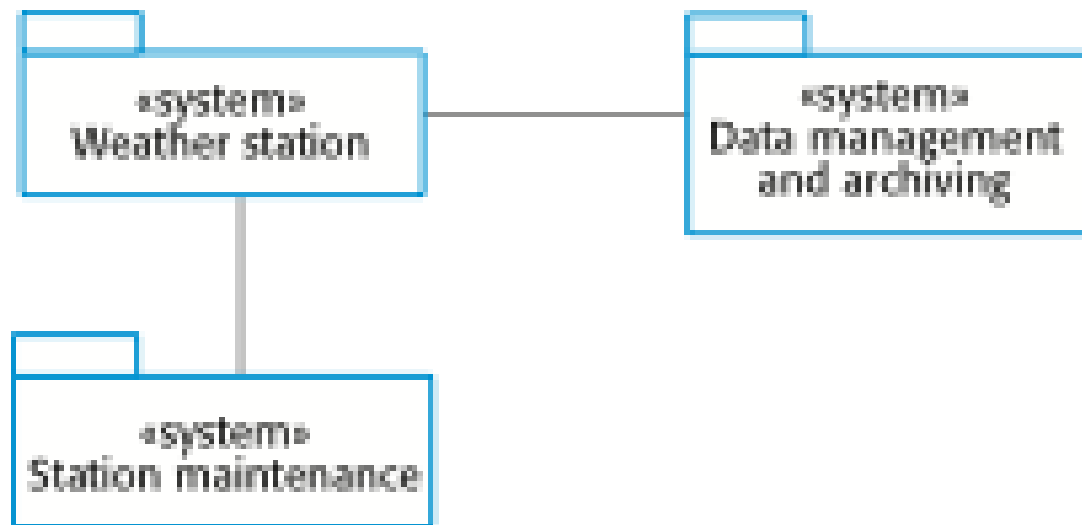
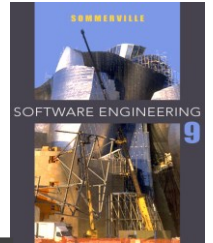
✧ Safety

- Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
- The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

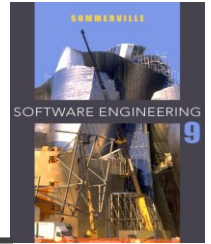
Wilderness weather station

- ✧ The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.
- ✧ Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
 - The weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period. Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

The weather station's environment



Weather information system



✧ The weather station system

- This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.

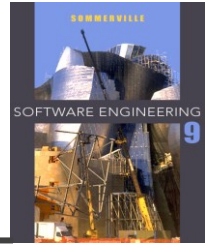
✧ The data management and archiving system

- This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.

✧ The station maintenance system

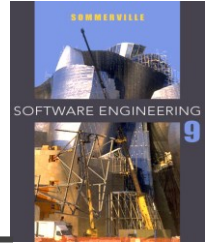
- This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.

Additional software functionality



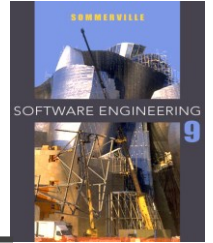
- ✧ Monitor the instruments, power and communication hardware and report faults to the management system.
- ✧ Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
- ✧ Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.

Key points



- ✧ Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- ✧ Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.
- ✧ Three case studies are used in the book:
 - An embedded insulin pump control system
 - A system for mental health care patient management
 - A wilderness weather station

Course structure and organization



✧ *Add your own material here about how you will be running the course*