

Software Engineering

JUNBEOM YOO

Dependable Software Laboratory
KONKUK University

<http://dslab.konkuk.ac.kr>

Ver. 5.0 (2017.09)

- ※ This lecture note is based on materials from Ian Sommerville 2014. (10th edition)
- ※ Anyone can use this material freely without any notification.

Contents at a Glance

Part I.
Basic Principles of Software Engineering

Chapter 1.

Introduction to Software Engineering

Topics Covered

1. Professional software development
2. Software engineering ethics
3. Case studies

Software Engineering

- **Software engineering** is concerned with **theories**, **methods** and **tools** for professional software development.
 - The economies of all developed nations are dependent on software.
 - Expenditure on software represents a significant fraction of GNP in all developed countries.
 - More and more systems are software controlled.
- **Software engineering** is concerned with cost-effective software development.
 - Software costs often dominate computer system costs.
 - The costs of software on a PC are often greater than the hardware cost.
 - Software costs is more to maintain than to develop.
 - For systems with a long life, maintenance costs may be several times development costs.

1. Professional Software Development

FAQs 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.
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. You can't, therefore, say that one method is better than another.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Software Products

- **Generic products**

- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
- Examples :
 - PC software such as graphics programs, project management tools, CAD software
 - Software for specific markets such as appointments systems for dentists
- The software requirements specification 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 software requirements specification is owned by the customer for the software and they make decisions on software changes that are required.

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

- **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.

Software Project Failures

- A number of software projects fail due to
 - Increasing system complexity
 - While new software engineering techniques help us to build larger and more complex systems, the demands change constantly.
 - 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 do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

Importance of Software Engineering

- More and more, individuals and society rely on advanced software systems.
- We need to be able to produce reliable and trustworthy systems economically and quickly(on time).
- 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 (→ maintenance cost).

Software Process Activities

- **Software specification**
 - Customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software development**
 - The software is designed and programmed.
- **Software validation**
 - The software is checked to ensure that it is what the customer requires.
- **Software evolution**
- The software is modified to reflect changing customer and market requirements.

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

Software Engineering Diversity

- No silver bullet for software engineering
 - 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.

Application Types

Type	Description
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.
Batch processing systems	These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
Entertainment systems	These are systems that are primarily for personal use and which are intended to entertain the user.
Systems for modeling and simulation	These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
Data collection systems	These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
Systems of systems	These are systems that are composed of a number of other software systems.

Software Engineering Fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
 - Systems should be developed using a managed and understood development process.
 - Different processes are used for different types of software.
 - Dependability and performance are important for all types of system.
 - Understanding and managing the software specification and requirements are important.
 - Where appropriate, you should reuse software that has already been developed rather than write new software.

Internet and Software Engineering

- The **Web** is now a platform for running application
 - 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

- The fundamental principles and ideas of software engineering are applicable to Web-based systems in the same ways.
 - They are another types of systems.
- Fundamental ideas for Web-based 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.
 - 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.

2. 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.
 - 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.
 - Intellectual property rights
 - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc.
 - Computer misuse
 - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (e.g., game playing on an employer's machine) to extremely serious (e.g., dissemination of viruses).

ACM/IEEE Code of Ethics

- The Code contains **8 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.

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.

3. Case Studies

Case Studies

1. A personal insulin pump

- An embedded system in an insulin pump used by diabetics to maintain blood glucose control

2. A mental health case patient management system

- **Mentcare** : A system used to maintain records of people receiving care for mental health problems

3. A wilderness weather station

- A data collection system that collects data about weather conditions in remote areas

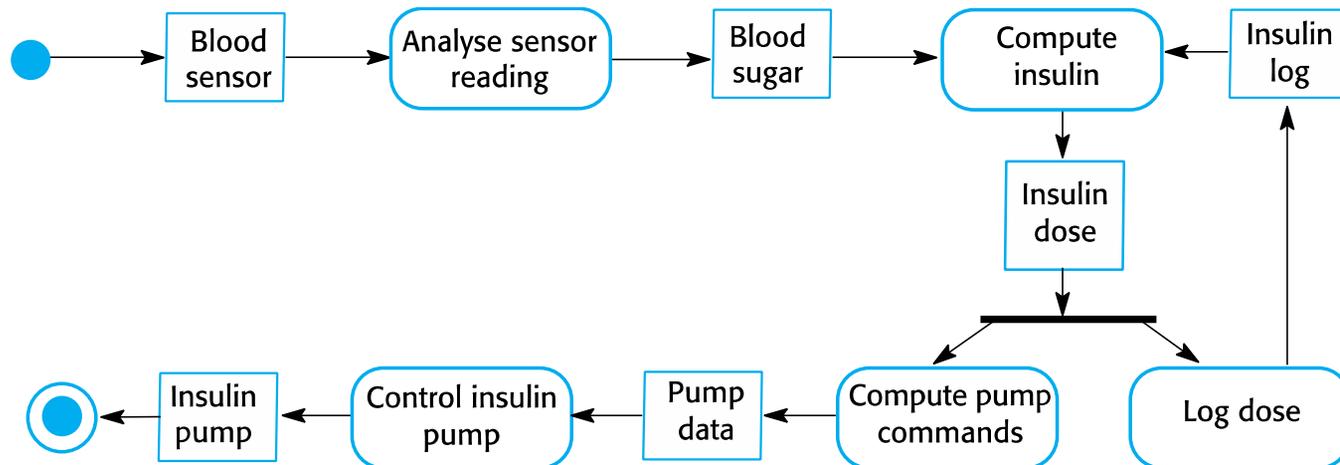
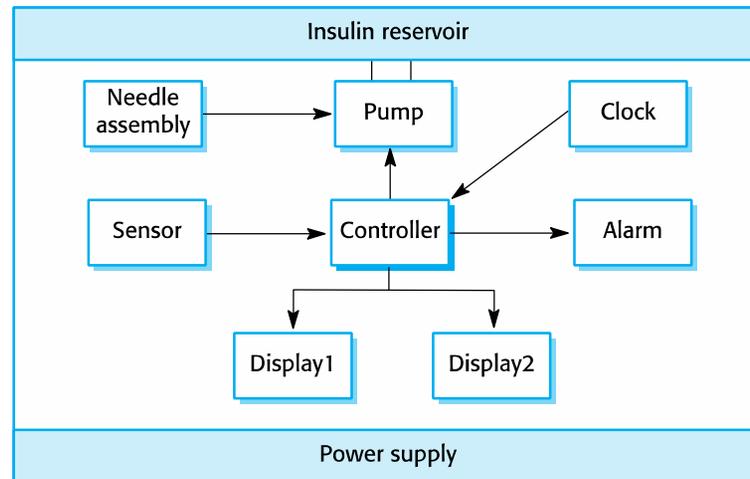
4. A digital learning environment

- **iLearn** : A system to support learning in schools

1. Insulin Pump Control System

- Requirements
 - 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 and Activity Model



Essential High-Level Requirements

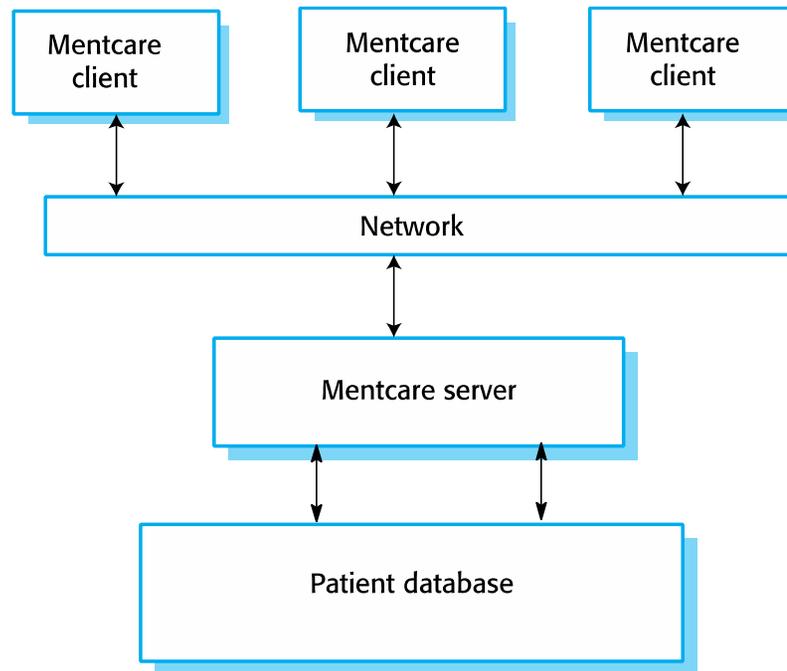
- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

2. Mentcare

- A patient information system to support mental health care
 - 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 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 and the Organization

- 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.



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

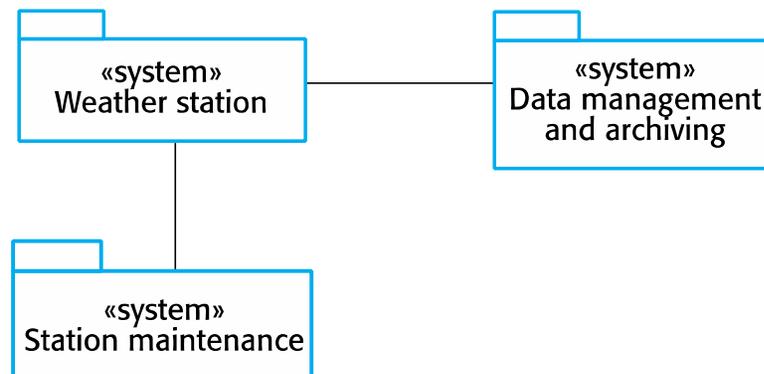
- Privacy
 - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
 - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should 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.

3. 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

- 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.

4. 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

- iLearn 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.
- 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.

iLearn Services

- Utility services
 - Provide basic application-independent functionality and which may be used by other services in the system.
- Application services
 - 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
 - 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

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

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.
- 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.