# Software Engineering

# 1
Software Quality and Principles

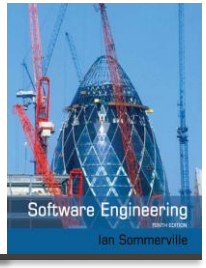
School of Computer Science
Prof. Euijong Lee

# Advise from the creator of C++


https://www.youtube.com/watch?v=-QxI-RP6-HM

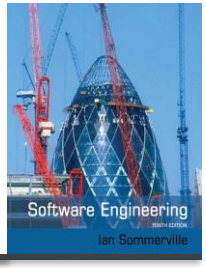
https://www.stroustrup.com/index.html

# Advise from the creator of C++




- 본인의 커리어를 지나치게 전문화 하지 마세요. (don't be overspecialized)
- 미래를 알고 있다고 너무 확신하지 마세요.
- 융통성을 가지고 (Be flexible), 커리어와 직업은 장기적인 것이라는 것을 기억하세요.
- 너무 많은 젊은이들이 자신이 뭔가를 최적화 할 수 있다고 생각하고, Right thing이 아닌 것에 전문화 하면서 많은 시간을 소비했다는 것을 깨닫습니다. 그리고 그 과정에서 번-아웃 됩니다.
- **우정을 쌓고 컴퓨팅 밖에서의 삶을 사는데 충분한 시간을 보내지 않았기 때문이에요.**
- 나는 많은 종류의 사람을 만납니다. 뭐라고 부르는지 모르겠는데, " Junior Geeks " ?
- 그들은 **오직 프로그래밍, AI, 그래픽 등의 컴퓨터에 대한 전문성만이 중요하다고 생각**합니다.
- 그렇지 않아요. 만약 그들이 다른 일을 하지 않는다면…
- 아이디어를 소통하지 않는다면, (혼자 하는) 스도쿠를 할 수 있습니다.
- **소통(Communication) 을 해야 해요.**
- 많은 **괴짜(Nerd)들이 이 사실을 잊고 있습니다**.
- 그들은 최고의 코드를 작성하기만 하면 세상을 바꿀 수 있다고 생각합니다.
- 하지만 경청할 줄 알아야 합니다.
- **사용자와 소통하고 그들로 부터 배울 수 있어야 합니다.**
- 그리고 **자신의 아이디어를 사용자에게 전달할 수 있어야 합니다.**

# Advise from the creator of C++




- 그러니 코딩만 할 수는 없습니다.
- **문화와 아이디어를 표현하는 방법에 대해 뭔가를 해야 합니다.**
- 내 말은, 나는 역사와 수학에 보낸 시간을 결코 후회하지 않는다는 것입니다.
- 수학을 정신을 날카롭게 하고, 역사는 자신의 한계에 대한 아이디어와 세상에 무슨 일이 일어나고 있는지를 알려줍니다.
- 그러니 너무 확신하지 마세요. **균형 잡힌 삶을 살기 위한 시간**을 가져보세요.
- **그리고 기회를 잡을 준비를 하세요.**
- 내 말은, 폭넓은 교육, 폭넓은 기술 세트(교육을 통해 기본적으로 기술 포트폴리오를 구축하는 것)는 기회가 왔을 때 이를 활용할 수 있다는 것을 의미합니다.
- 가끔 기회를 알아차릴 수 있습니다. 우리에게는 많은 기회가 있습니다.
- 하지만, 그 중 상당수는 우리가 활용하지 못하거나 알아차리지 못합니다.
- **저는 표준 컴퓨터 과학, 컴파일러, 여러 언어를 공부하는 등 상당히 폭넓은 교육**을 받았습니다.
- 당시 제가 알고 있었던 언어는 약 20여가지 였던 것 같아요. 그리고 머신 아키텍처와 운영체제도 다뤄봤죠.
- **이 기술 세트는 유용했어요(skill set turned out to be useful)**

# Topics covered

## Professional software development

What is meant by software engineering.

## Software engineering ethics

A brief introduction to ethical issues that affect software engineering.

# Topics covered

**01** What is Software?

**02** What is Software Engineering?

**03** Characteristics of developing software

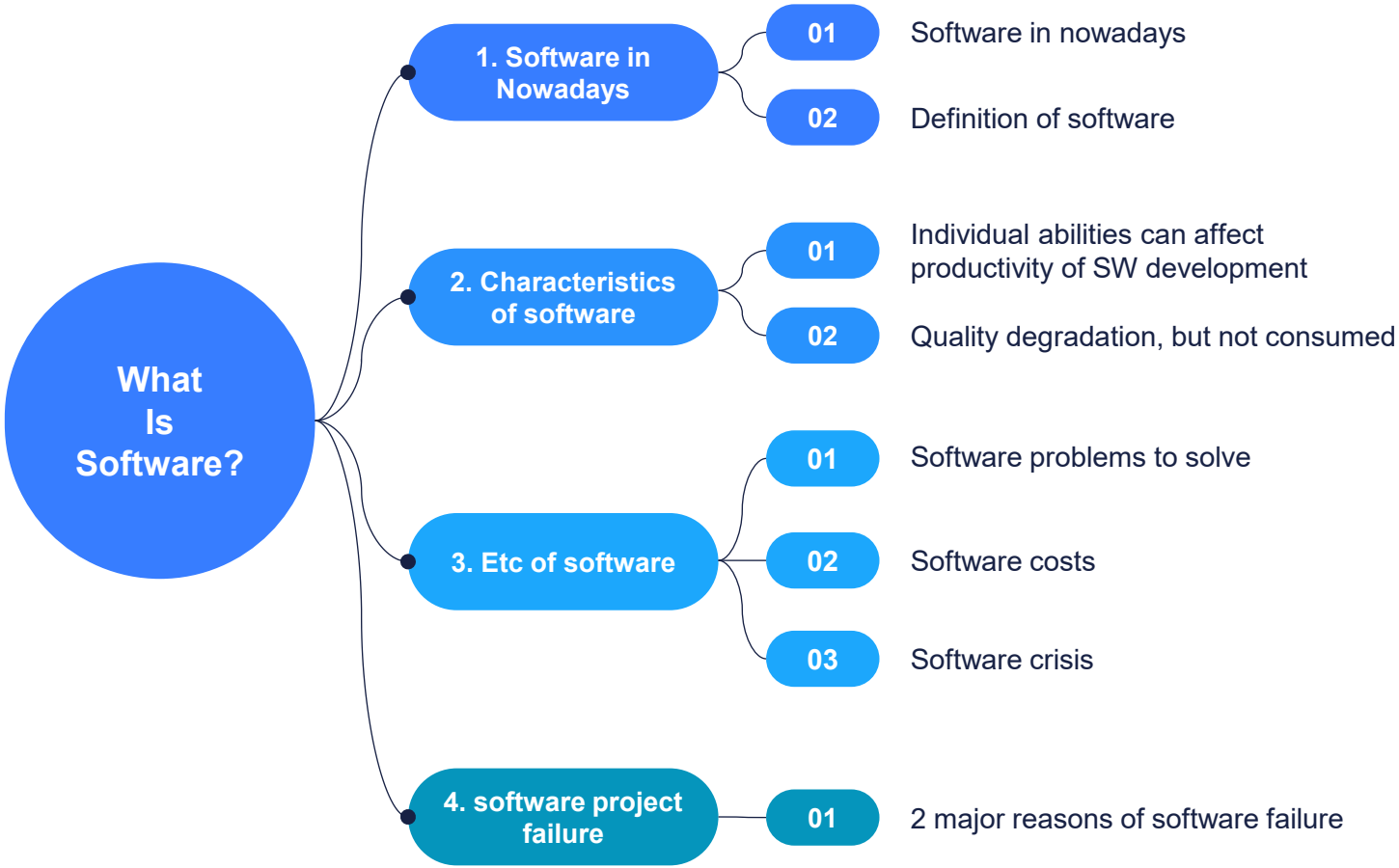
**04** Details of Software Engineering

**05** Divergence of Software

**06** Software engineering ethics

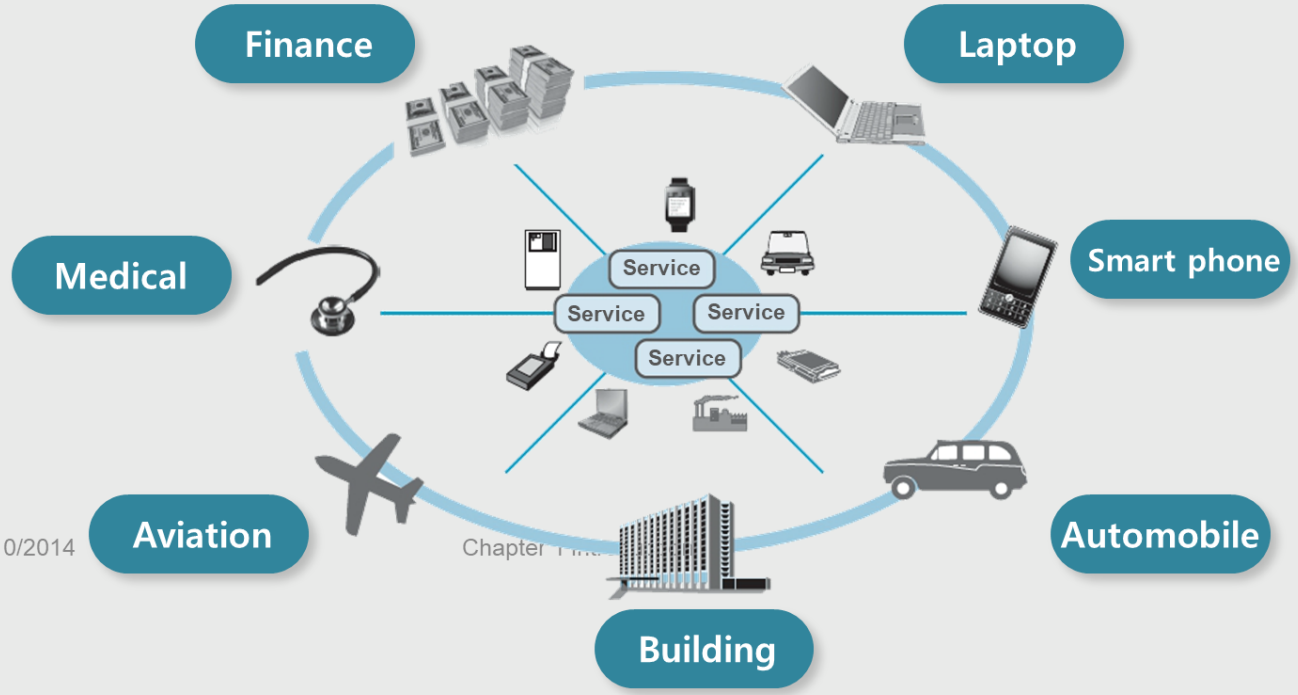
**07** Additional Slides

# 1-1: What is software?



# Software in nowadays

**[ Nowadays, software is used in several fields of society ]**



# Software in nowadays

- The economies of ALL developed nations are dependent on SW.
- 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(: Gross National Product) in all developed countries.

# Definition of software

## [ Program VS Software ]

**Program**

a collection of instructions executed by a computer,
including compiled code as well as source code

**Software**

not only programs (codes),
but all artifacts created during software development

**Software e.g.**

Data structure, database, structure of database, test results,
documents from each development stages, user manual

**“ Software is a comprehensive concept including more than just a program ”**

# Characteristics of Software

**Individual abilities (developer's ability) can affect productivity of software development.**

**[ No, manufacture ]**

- to make something
- to produce goods in large numbers
- usually in large numbers in a factory

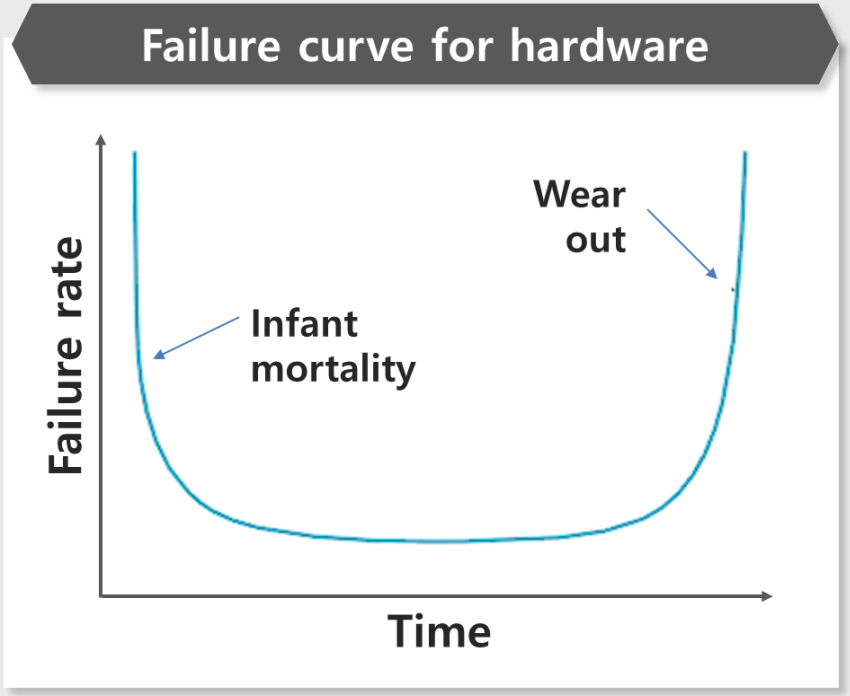
**[ Yes, development ]**

- the process of developing something new

# Characteristics of Software

**[ Quality degradation, but not consumed ]**

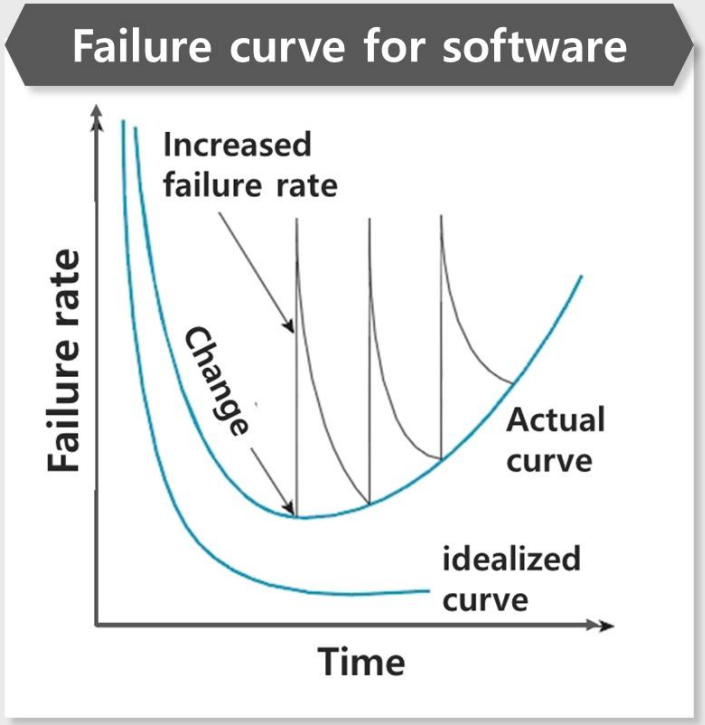
- Hardware has a high initial failure rate but can be used for a long time after error correction
- Hardware failure is increased rapidly because of environmental condition (e.g., dust, vibration, and heat)



# Characteristics of Software

## [ Quality degradation, but not consumed (2) ]

- Unlike hardware, software has no damage by parts (i.e., Environments have little impact on the software)
- In the idealized software curve, the failure rate is decreased after error corrections
- Software can reflect requirements in stage of use (i.e., software is updated to reflect user's requirement)
- The changes may cause increasing of failure rate



# Software problems to solve

**Problem #1** **Slow advancement of software development**

- Slow development speed compared to hardware
- Hardware has evolved beyond imagination since the 1970s
- Software is steadily evolving, but slower than hardware

**Problem #2** **Increasing user demand for new software**

- Hardware has high productivity
  - manufacturing by assembling parts already made
- Software has low productivity
  - development processes

# Software problems to solve

**Problem #3**

## Partial use of management techniques

- Software development requires management techniques
- Exhaustive management via
  Project Management Knowledge System (PMBOK)
- Using project management techniques and
  providing systematic strategies.

Etc of Software

# Software costs

- **Software costs often dominate computer system costs.**
  - The costs of software on a PC are often greater than the hardware cost.

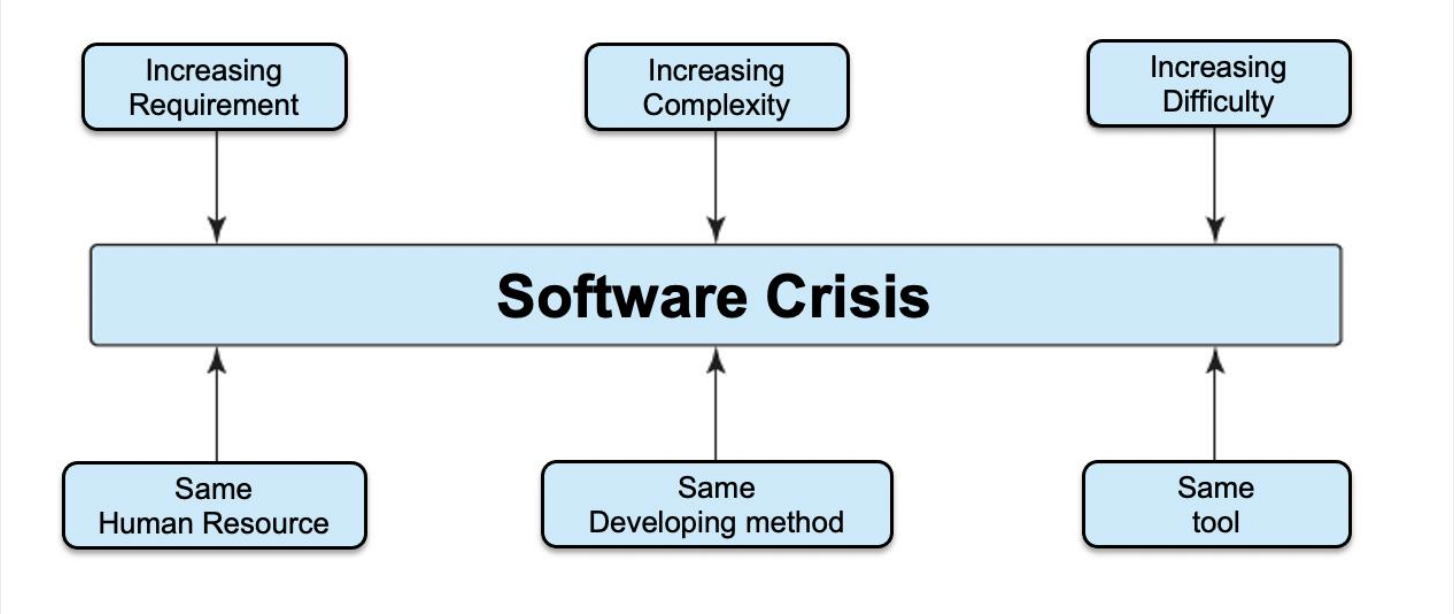
- **Software costs more to maintain than it does to develop.**
  - For systems with a long life, maintenance costs may be several times development costs.

- **Software engineering** is concerned with **cost-effective** software development.

# Software crisis

## Software crisis?

**Software crisis** is a term used in the early days of computing science for **the difficulty of writing useful and efficient computer programs** in the required time.




Source: <All about software engineering>. 생능출판사. 최은만, & wikipedia

Software project failure

# 2 major reasons of SW project failure (1)

**Increasing system complexity** (시스템 복잡도의 증가)

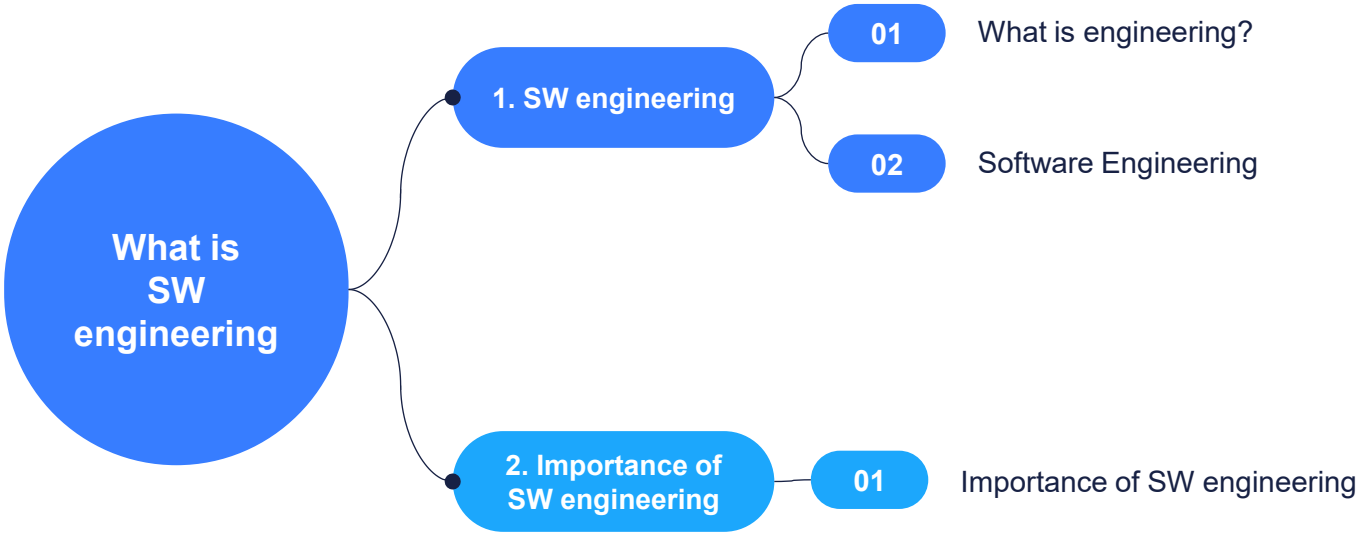
- As new **software engineering techniques help** us to build larger, more complex systems, the demands change.
- Systems have to be **built and delivered more quickly**; **larger**, even more **complex** systems are required;
- Systems have to have **new capabilities** that were previously thought to be impossible.

# 2 major reasons of SW project failure (2)

**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**.

# 1-2: What is Software Engineering?



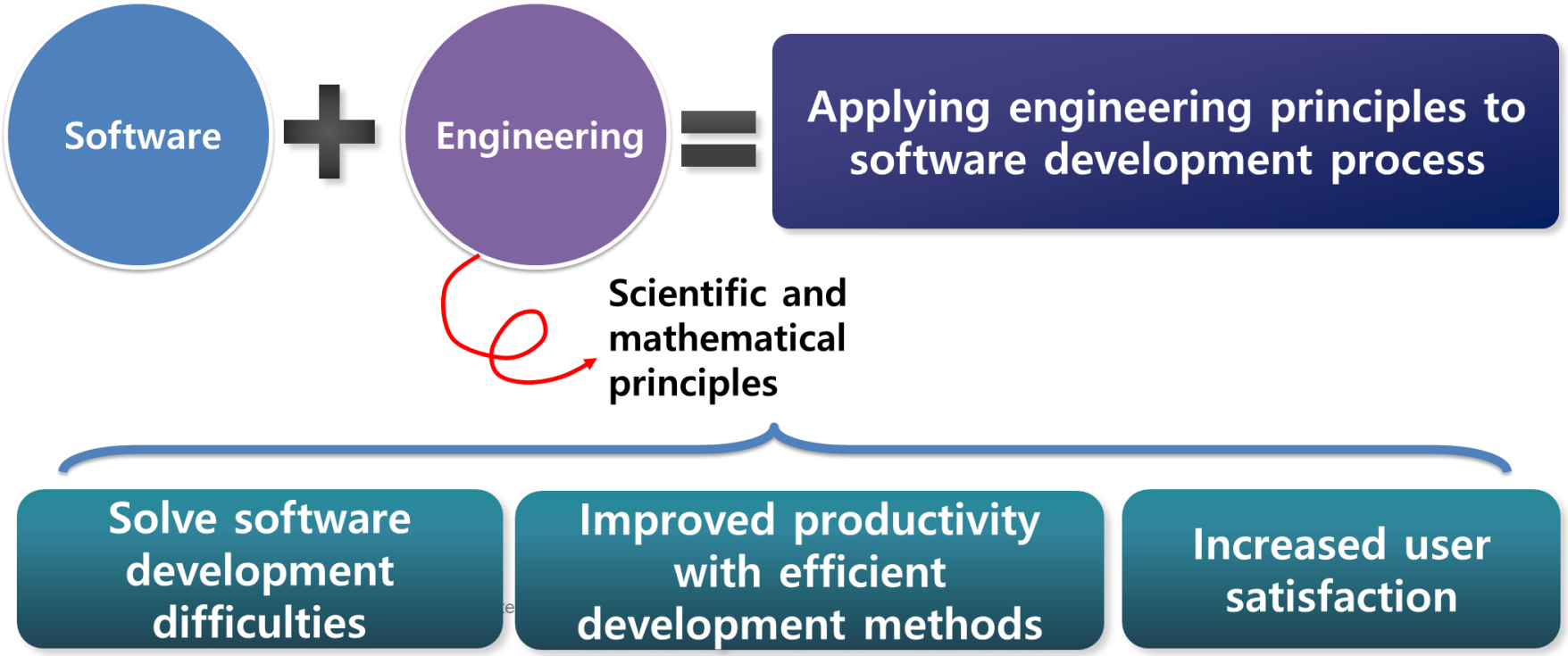
# What is engineering?

- the study of using **scientific principles** to design and build machines, structures, and other things, including bridges, roads, vehicles, and buildings:

- Developing engineering accumulate skills to solve problems, and it is developed **engineering principles**

- Applying engineering principles make **procedures** and **standards** to solve specific problems

# Software Engineering



# Software Engineering

**[ Software engineering ]**

- 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.

# Importance of SW engineering

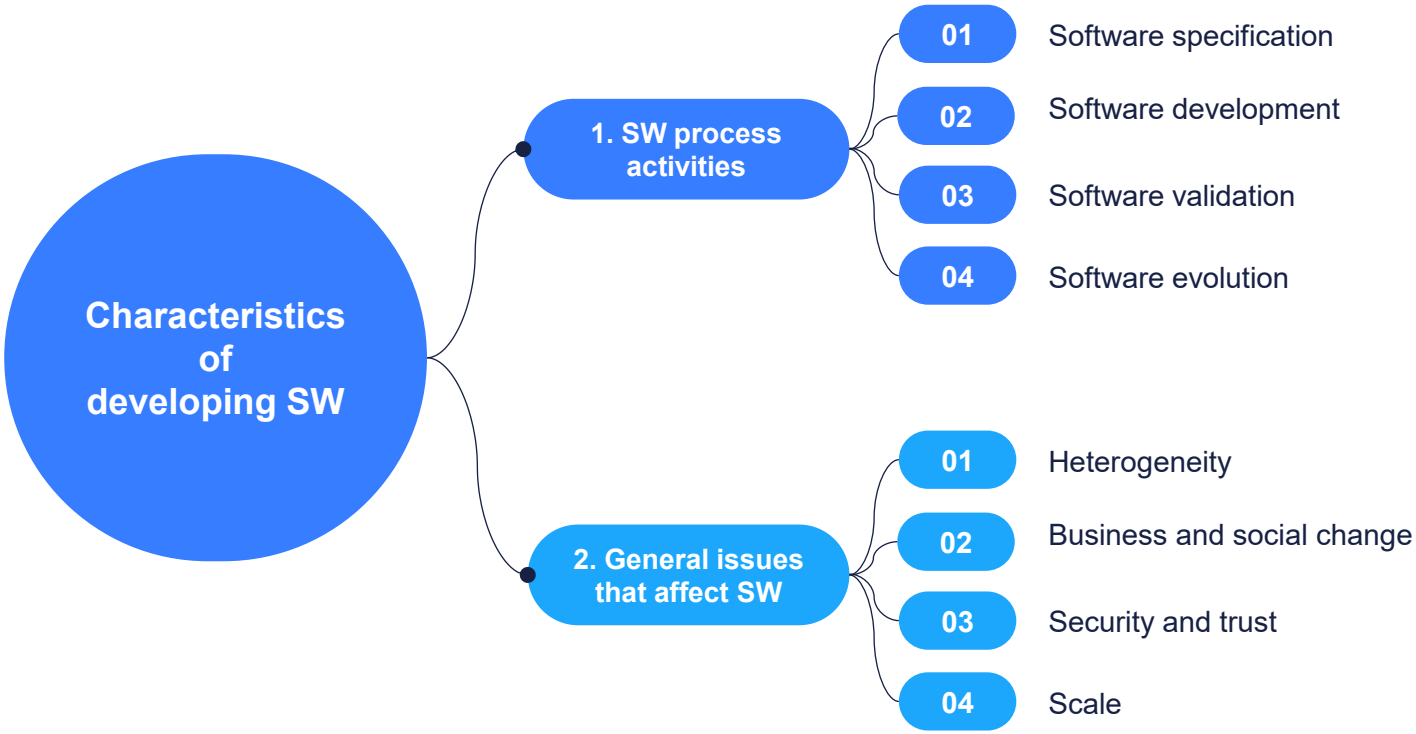
## Dependency of Software

- 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**.

## Cost saving

- 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, **most costs** are the costs of **changing the software** after it has gone into use.

# 1-3: Characteristics of developing software



SW process activities

# Software Process Activities

**Software specification** (명세화)

where customers and engineers **define the software** that is to be produced and the **constraints** on its operation.

**Software development** (개발)

where the software is **designed** and **programmed**.

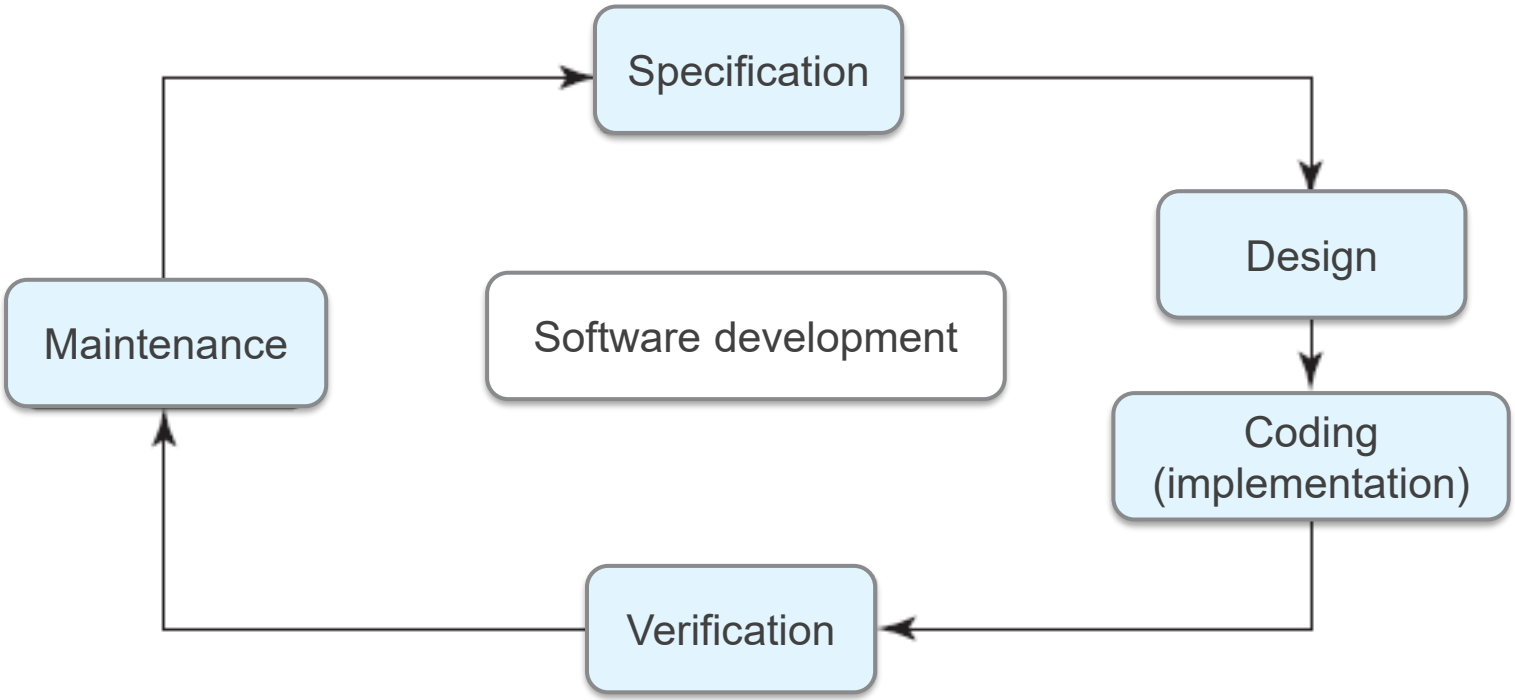
**Software validation** (검증)

where the software is checked to **ensure** that it is what the customer **requires**.

**Software evolution** (진화)

where the software is **modified** to **reflect changing** customer and market requirements.

# Software Process Activities



# General issues that affect SW (1)

## Heterogeneity (이질성)

- 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 SW (2)

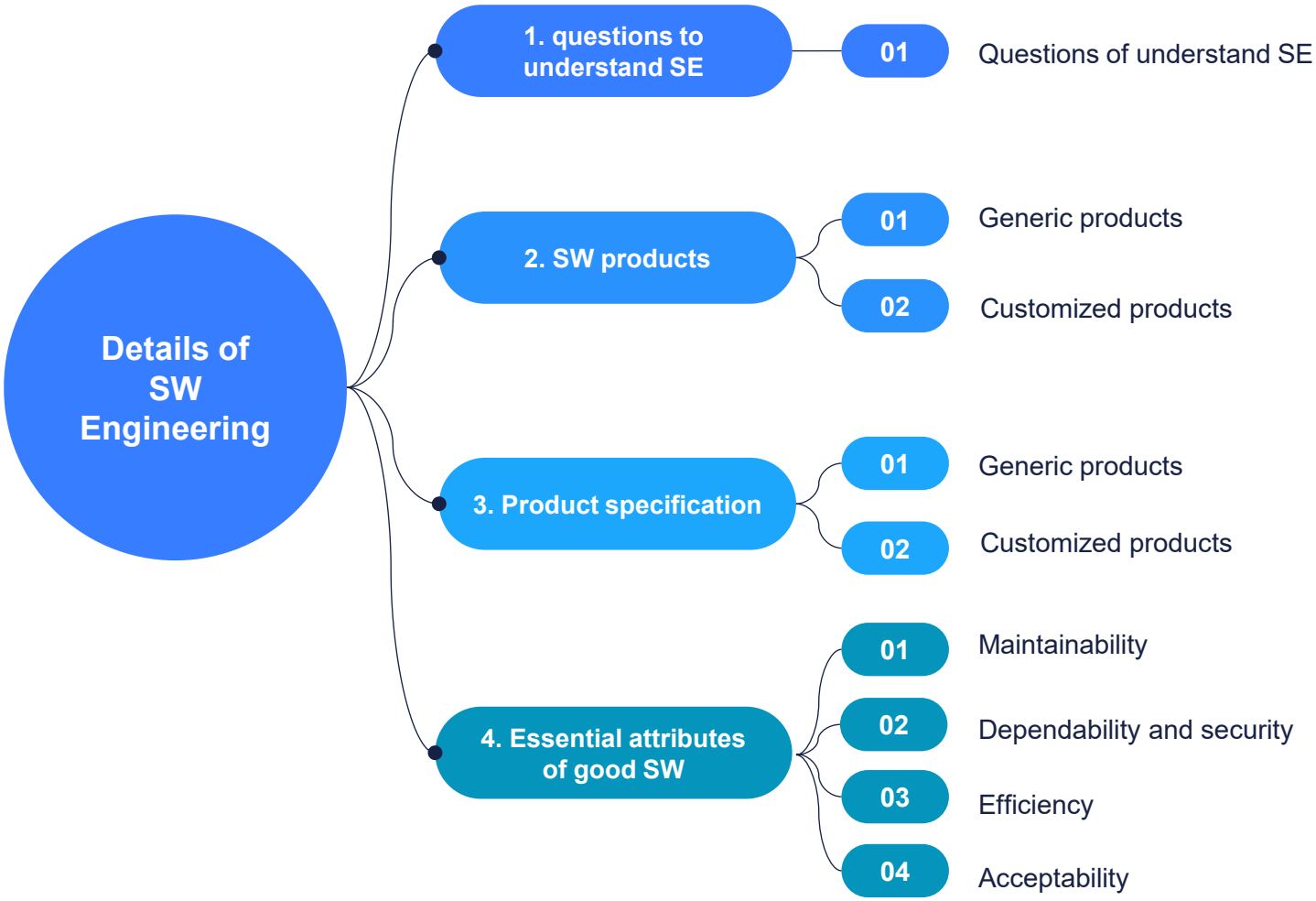
## Security and trust (보안과 신뢰)

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

## Scale (규모)

- Software must 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.

# 1-4: Details of Software Engineering



# Question to understand SE (1)

**[ 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? ]**

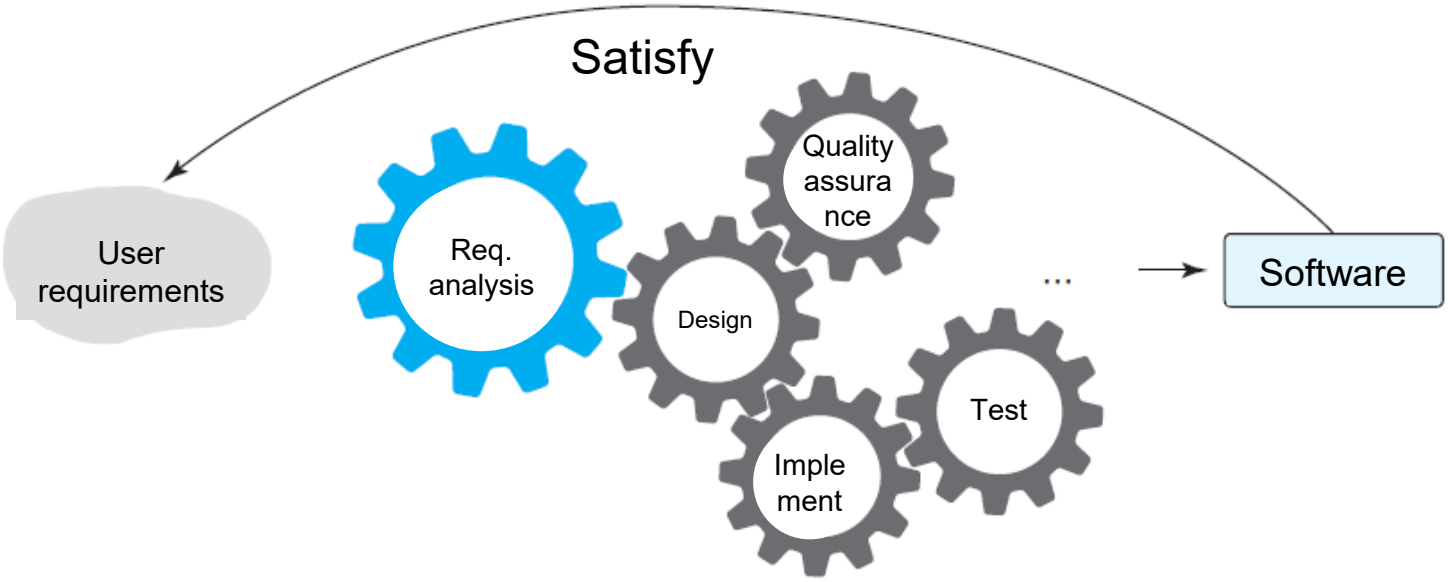
- Software engineering is an **engineering discipline** that is concerned with all aspects of software production.

# Question to understand SE (2)

**[ What are the fundamental software engineering activities? ]**

- Software specification, software development, software validation and software evolution.

Source: <All about software engineering>. 생능출판사. 최은만

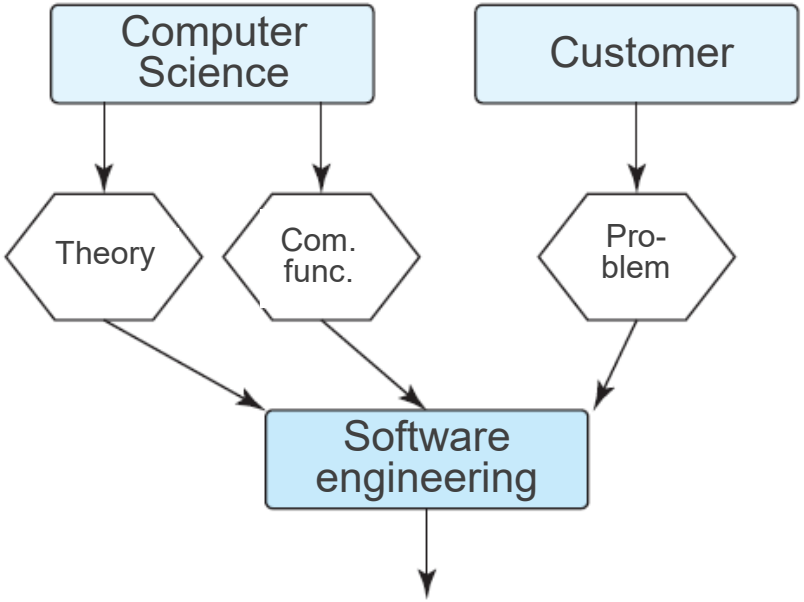


# Question to understand SE (3)

**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.

Source: <All about software engineering>. 생능출판사. 최은만

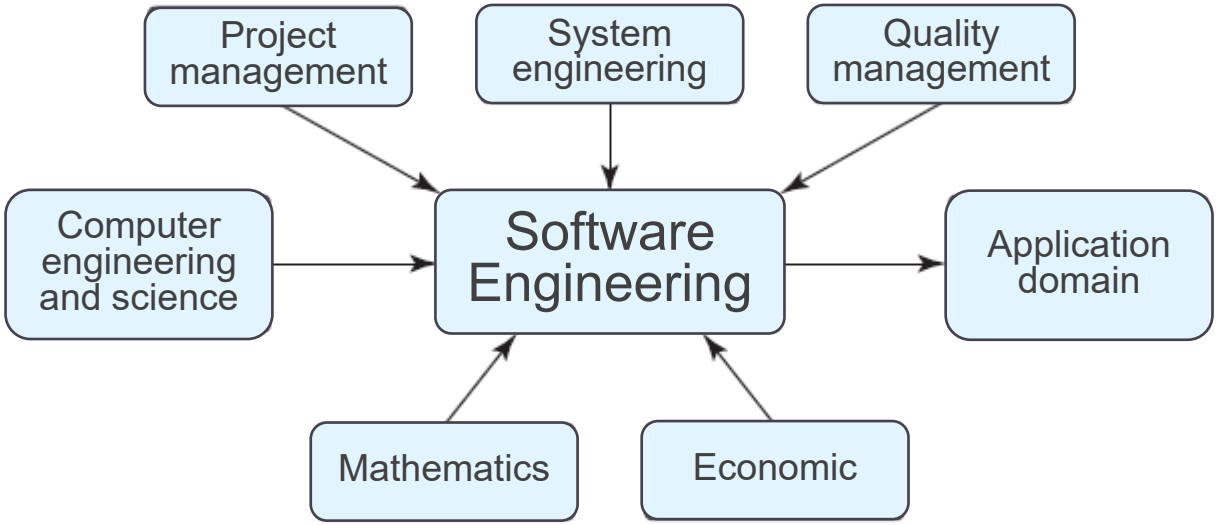


# Question to understand SE (4)

**[ 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**.

Source: <All about software engineering>. 생능출판사. 최은만



# Question to understand SE (5)

**[ 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.

# Question to understand SE (6)

**[ 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.

**[ 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.

## 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.

# 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.

# Essential attributes of good Software (1)

## Maintainability (유지보수성)

- 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 (확실성과 보안성)

- Software **dependability** includes a range of characteristics including reliability(신뢰성), security(보안성) and safety(안정성).
- 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.

# Essential attributes of good Software (2)

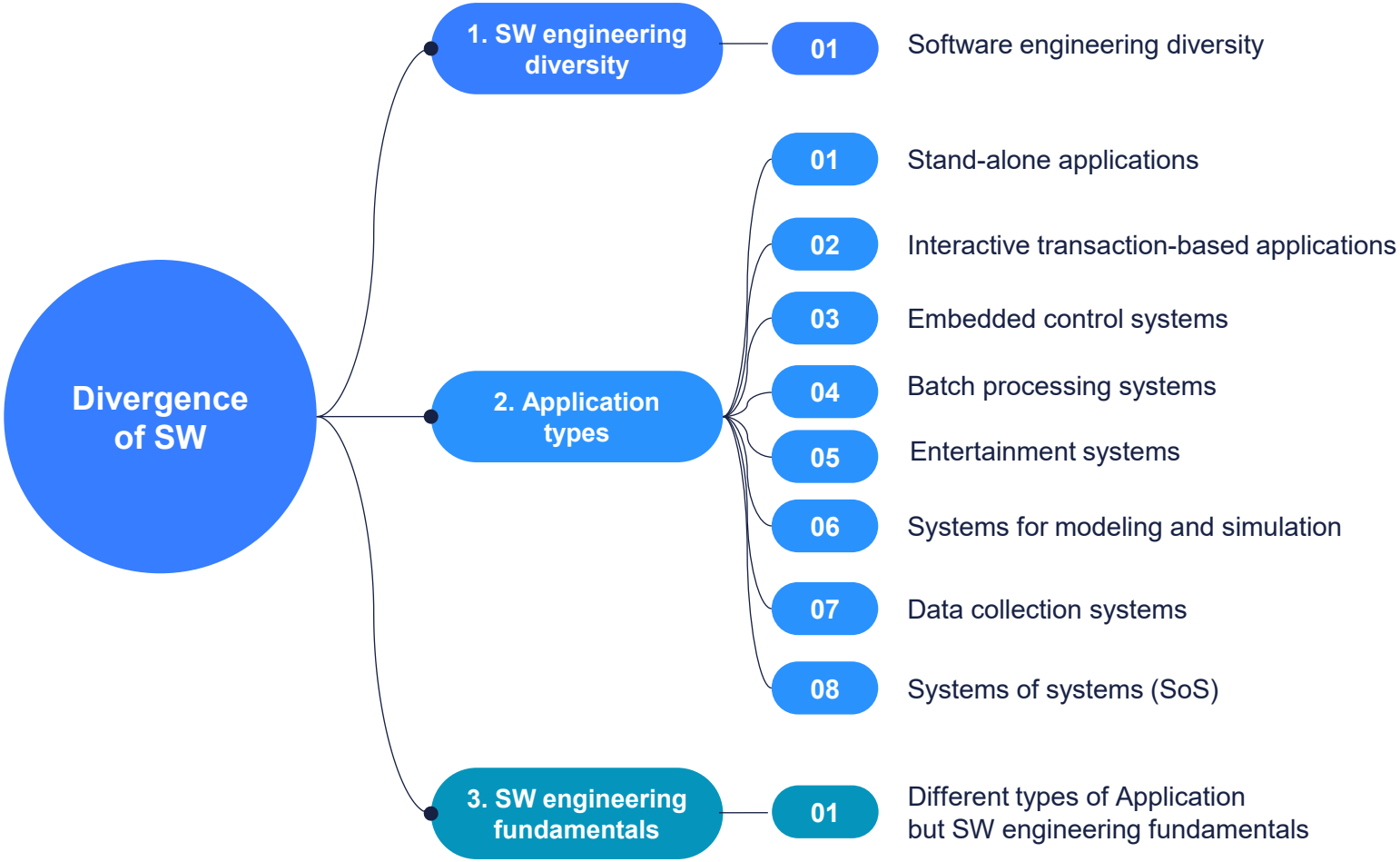
## Efficiency (효율성)

- 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 (수용성)

- 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.

# 1-5: Divergence of 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.

# Application types (1)

**[ 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.

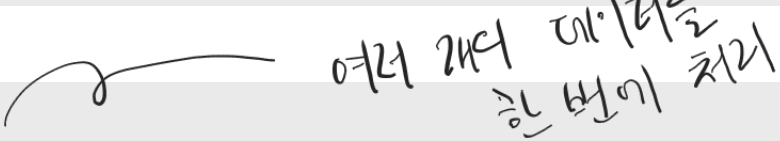
**[ Interactive transaction-based applications ]**

- 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.

# Application types (2)

**[ 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.

# Application types (3)

**[ Entertainment systems ]**

Netflix Youtube Steam ...

- These are systems that are primarily for personal use, and which are intended to entertain the user.

**[ Systems for modelling and simulation ]**

- These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

# Application types (4)

**[ 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 (SoS) ]**

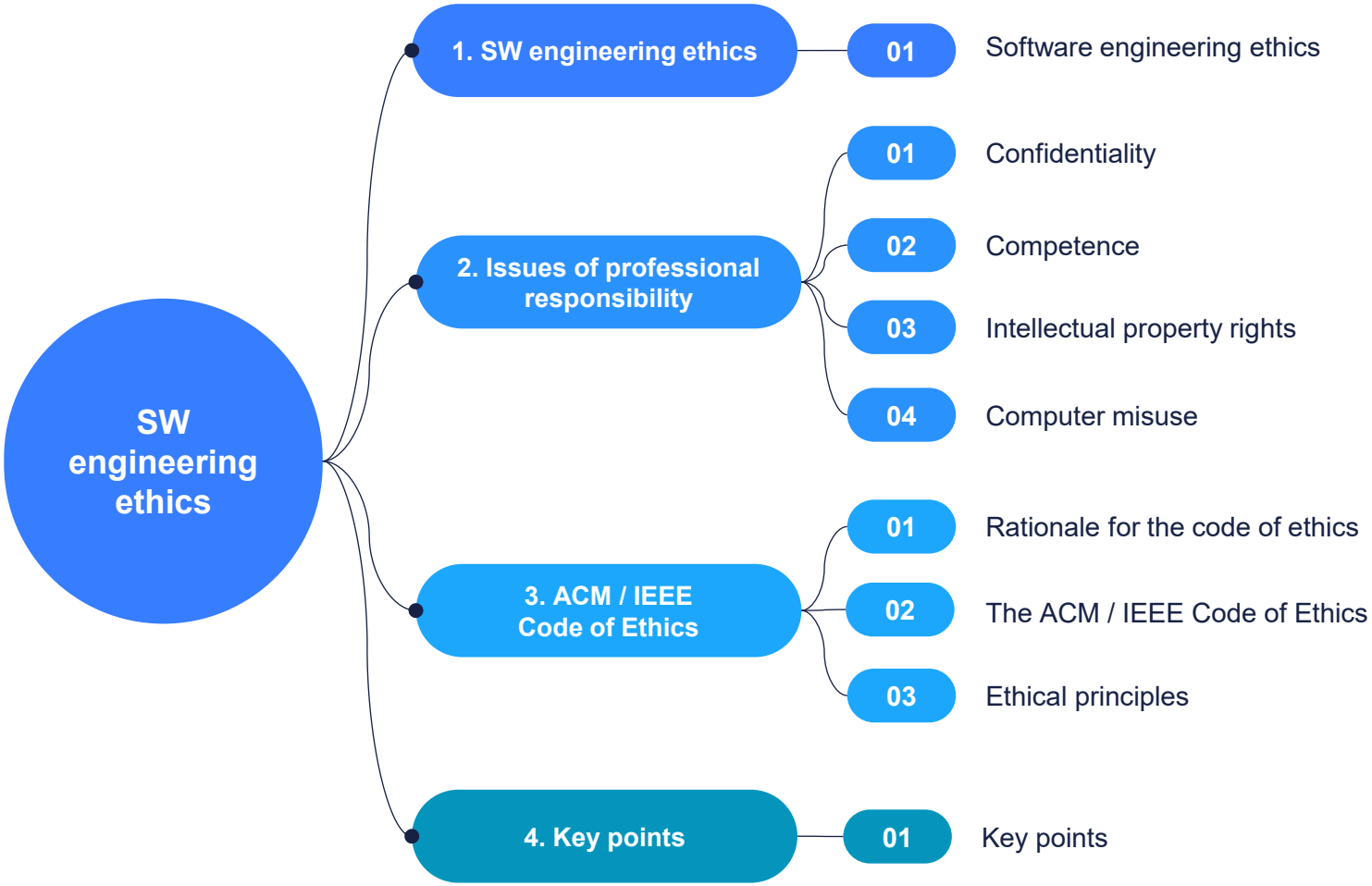
- These are systems that are composed of a number of other software systems.

# Different types of Application but SW engineering fundamentals

Some fundamental principles apply to **all types of software system**, irrespective of the development techniques used:

- Systems should be developed using a managed and understood development process.
  - Of course, different processes are used for different types of software.
- Dependability and performance are important for all types of system.
- Understanding and managing the software specification and requirements(what the software should do) are important.
- Where appropriate, you should reuse software that has already been developed rather than write new software.

# 1-6: 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 (1)

## Confidentiality (비밀유지)

- Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

## Competence (능력)

- Engineers should not misrepresent their level of competence.
- They should not knowingly accept work which is outwith their competence.

# Issues of professional responsibility (2)

## Intellectual property rights (지적재산권)

- 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 (컴퓨터 남용)

- 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 Ethices

- 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:

ACM / IEEE Code of Ethics

# 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.

ACM / IEEE Code of Ethics

# Ethical principles (Korean)

## 소프트웨어공학 윤리강령

2016년 5월 16일 제정

출처: ACM/IEEE-CS Software Engineering Code of Ethics and Professional Practice

http://www.acm.org/about/se-code#short

소프트웨어 엔지니어는 소프트웨어의 분석, 명세, 설계, 개발, 테스트, 유지보수 전반의 일을 유익하고 존중받는 직업적 활동으로 만들기 위해 노력해야 한다. 공공의 건강, 안전, 복지를 위하여 소프트웨어 엔지니어는 다음 8개의 원칙을 준수해야 한다:

1. 소프트웨어 엔지니어는 공공의 이익에 부합하도록 일해야 한다.
2. 소프트웨어 엔지니어는 자신의 고객 및 고용자의 최선의 이익을 위하여 일하되 공공의 이익과 부합되는 방식으로 일해야 한다.
3. 소프트웨어 엔지니어는 자신이 개발한 제품과 그에 대한 수정이 가능한 최고의 전문적 수준에 이르도록 노력해야 한다.
4. 소프트웨어 엔지니어는 자신의 전문가적 판단에 있어서 언제나 정직성과 독립성을 유지해야 한다.
5. 소프트웨어공학 관리자와 리더는 소프트웨어 개발과 유지보수에 대한 관리를 함에 있어서 윤리적 방법을 따르고 이를 장려하여야 한다.
6. 소프트웨어 엔지니어는 공공의 이익과 부합되는 방식으로 자신의 직업의 청렴성과 명성을 발전시켜야한다.
7. 소프트웨어 엔지니어는 자신과 함께 일하는 동료들에 대해 언제나 공정하고 지원을 아끼지 않아야 한다.
8. 소프트웨어 엔지니어는 자신의 전문성 발전을 위해 평생 배우고, 자신의 전문분야의 일을 수행함에 있어서 윤리적인 방법의 사용을 장려해야 한다.

Source: http://sigsoft.or.kr/

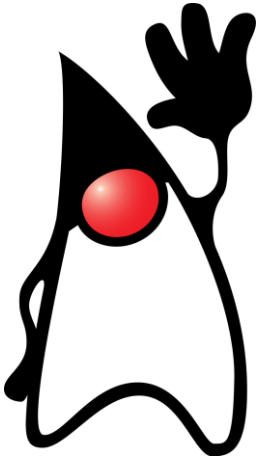
ACM / IEEE Code of Ethics

# Ethical thinking



**James Gosling**

**Father of Java**



https://www.youtube.com/watch?v=IT__Nrr3PNI

" I starting to **think about ethical choices in my life**.

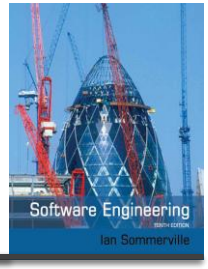
I'm a big science fiction fan.

I got to thinking about just about technical decision (that) I make.

In terms of "Are you building Blade Runner or Start Track" **which future would you rather live in?** "

ACM / IEEE Code of Ethics

# Ethical thinking




STAR TREK or BLADE RUNNER



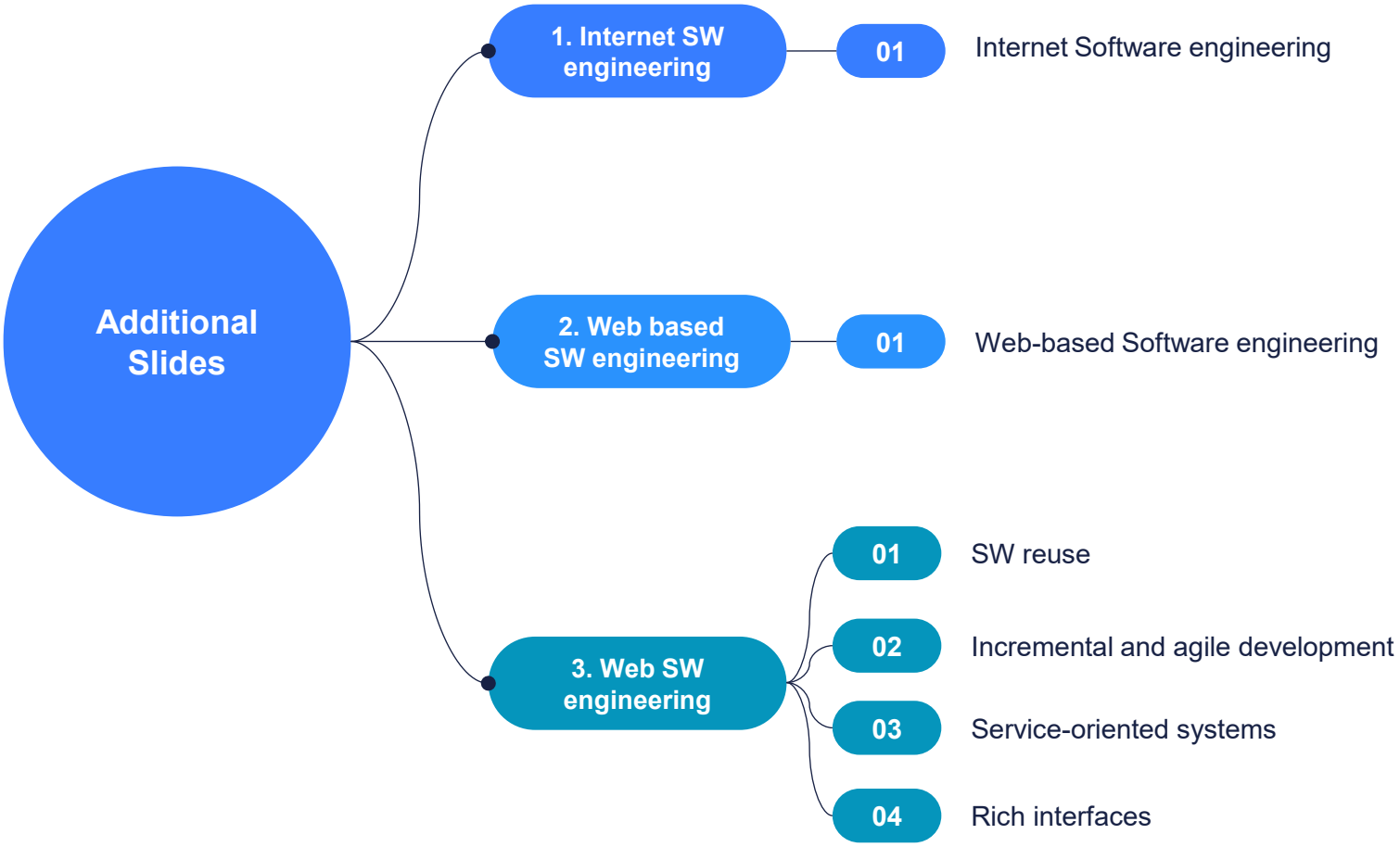
# Key points

- 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.

# 1-7: Additional Slides



# Internet Software Engineering

**[ Internet software engineering ]**

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.

- Web services allow application functionality to be accessed over the web.

- 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 ]**

- 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 (1)

## 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 (2)

## 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.