CSE4006 Software Engineering

01. Software & Software Engineering

Scott Uk-Jin Lee

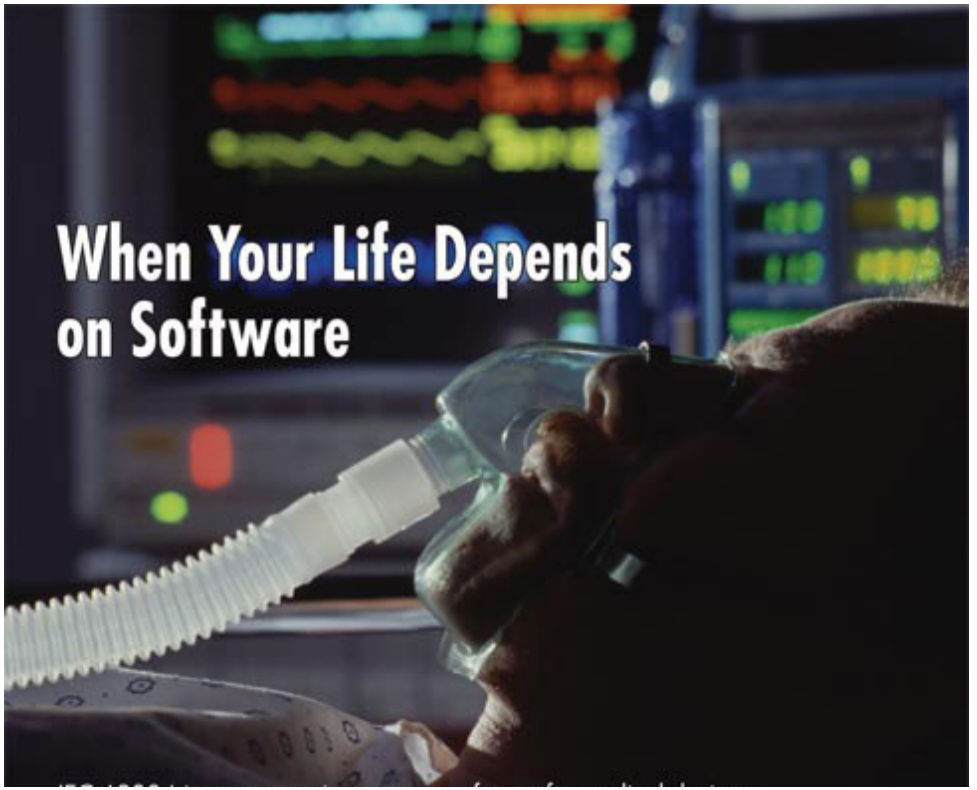
Department of Computer Science and Engineering
Hanyang University ERICA Campus

1^{st} Semester 2015

lab(se);

Software

- is the single most important technology in our world
- is everywhere
  - transportation, medical, telecommunication, military, industry, entertainment, education, office, ...



lab(se);

Software

- delivers the most important product of our time: **information**
  - transforms data suitable for specified context
  - manages business information to enhance competitiveness
  - provides gateway to worldwide information (Internet)
  - provides means to acquire information

Software

- changed significantly in the last half decade
  - hardware advancements:
    - dramatic increase in performance
    - profound changes in computer architecture
    - vast increase in memory and storage capacity
    - variety of exotic I/O options become available

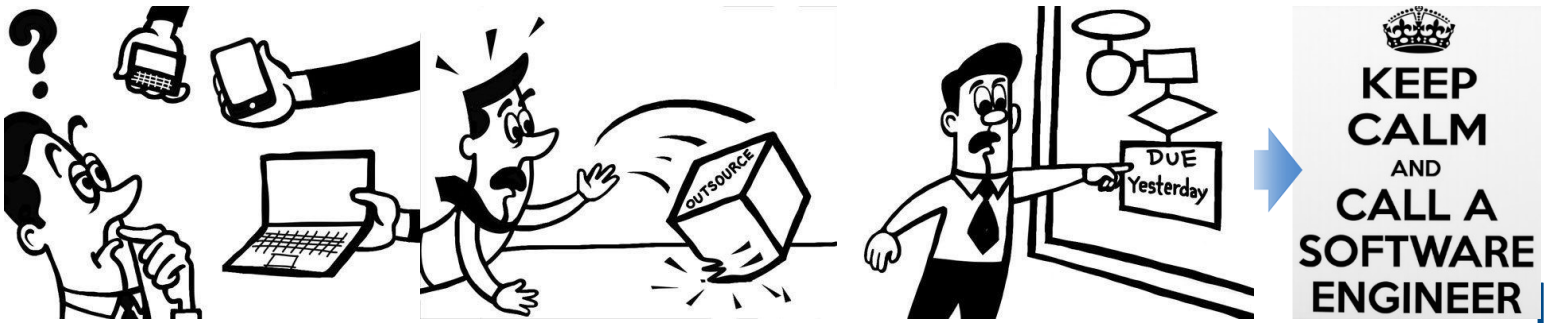
→ developments of more sophisticated & complex software
→ software industry become dominant factor in economic
→ lone programmer → team of software specialists

lab(se);

Software

- unchanged concern about software and its development
  - **why** does software development take so long to finish?
  - **why** is software development cost so high?
  - **why** can't we find all errors before delivering software to customers?
  - **why** spend so much money and time on maintaining existing software?
  - **why** is it so difficult to measure progress on software development or maintenance?
  - **why** ...

→ need to adopt **Software Engineering** practice



Software

Definition :

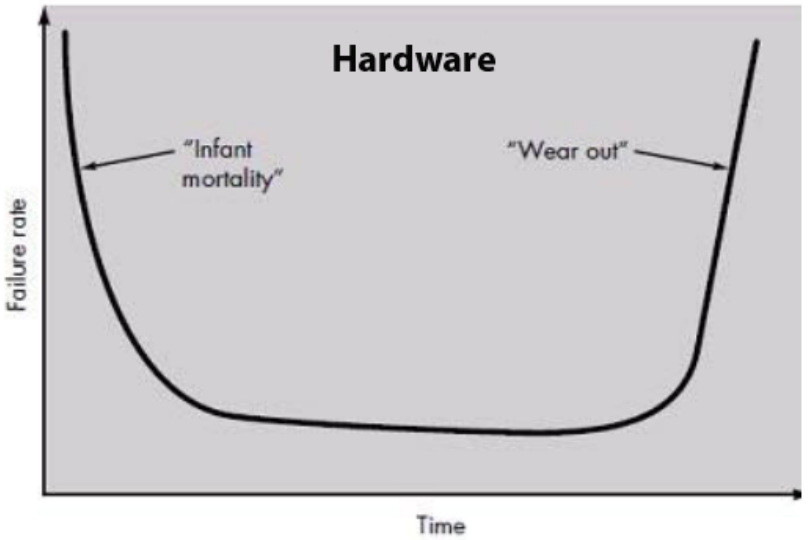
1) instructions(programs) that provides desired features, function, and performance when executed;
2) data structures that enables the programs to adequately manipulate information;
3) descriptive information in both hard copy and virtual form describing operations and use of the program

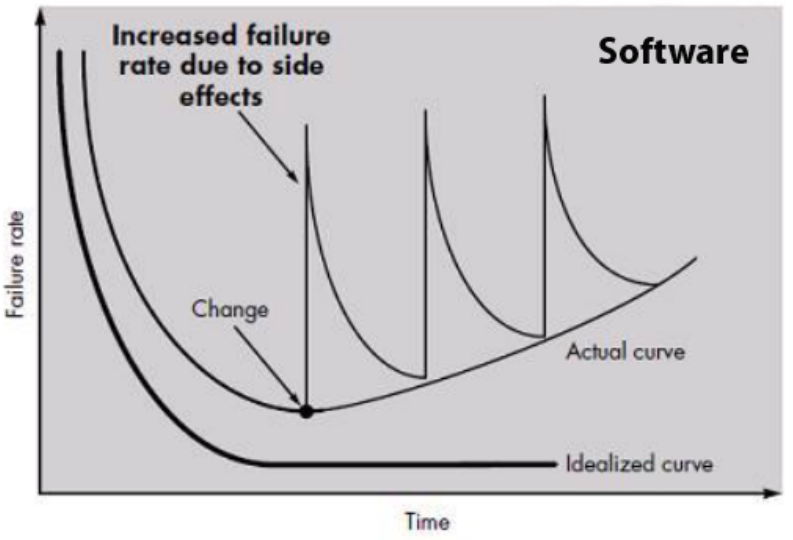
- does this definition (or any other from textbooks/dictionaries) give you a better understandings on **what Software is**?

lab(se);

Hardware vs. Software

- Failure curves





- Software is logical rather than physical
  - → **does not wear out** but deteriorate

lab(se);

Software is ...

- a set of items or objects that creates "**Configuration**"
  - programs, documents, data, ...

못 쓰는 프로그램 등은 소프트웨어가 아님

→ the entire information required for :

- a program +
  development, operation, and maintenance of the program

- intangible
- developed or engineered, not manufactured
- sophisticated and complex
- custom built

lab(se);

Software Application Domain

- System software OS/ editor/ file manager
- Application software
- Engineering/Scientific software
- Embedded software 엔드유저/시스템에 다른 feature나 funtionality를 마련해줌
- Product-line software
- Web/Mobile applications
- Artificial Intelligence (AI) software

lab(se);

Software Application Domain

evolving to dominate industry :

- Web application
- Mobile application
- Cloud Computing
- Product-line software

Software: New Challenges

- Internet of Things (IoT)
- Big data
- Mobility (Wearables)
- Security
- Also ...
  - Education
  - Legacy
  - License
  - Integration
  - Rejection

lab(se);

Emerging Software Technologies & Trends

Legacy Software

legacy system 오래되고 일시 중단 후 교체가 어려운 시스템 (e.g. 은행, 증권사 등등)

- developed decades ago, supportive to core business functions, and indispensable to business
  - → characterized by longevity & business criticality
- continuously modified to meet changes in business requirements and computing platform
- needs to evolve when it must :
  - be adapted to meet the needs of new computing environments or technology
  - be enhanced to implement new business requirements
  - be enhanced to make it interoperable with other (more modern) systems or databases
  - be re-architectured to make it viable within a evolving computing environment

lab(se);

Software Engineering

- Definition:
  고품질 저비용 효율 짱짱을 원한다!
  그것을 위한 방법론!
  - establishment and use of sound engineering principles in order to economically develop, operate, maintain software that is reliable and works efficiently on real machines.

- Force behind the emergence of Software Engineering
  - Management's incompetence in predicting the development cost, time, and effort for software
  - Low quality software
  - Increased importance of maintenance
  - Developments of hardware and software technologies
  - Increased demands on software

lab(se);

Software Engineering Layers

- **A Quality Focus**
  - any engineering approach must rest on an organisational commitment to quality
- **Process** - forms basis for :
  - management control of software project
  - establish context where technology methods are applied
  - work products (models, docs, data, reports, . . .) are produced
  - milestones are established
  - quality ensured
  - change managed properly

lab(se);

Software Engineering Layers

- **Methods** - provides technical how-to's for building software :
  (rely on set of basic principles that govern each area of technology)
  - communication
  - requirement analysis
  - design modeling
  - program construction
  - testing and suport
- **Tools** - provides :
  - automated & semi-automated support for process and methods
  - tool integration → Computer-Aided Software Engineering

lab(se);

Software Process

- Generic Process Framework
  1. Communication — Customer(갑)/ stakeholder 와 대화 -> spec 도출
  2. Planning — Task Definition && Division
     Work Product, Risk Management
     (Progress Destination)
     기간 스케쥴링/ 리소스(인원투입/비용) 플랜
  3. Modeling — BluePrint
     아키텍쳐링
  4. Construction — 아키텍쳐에 맞춰 세부적으로 실질적인 디자인
     Testing/Validation/Verification
  5. Deployment — Customer에게 전달
     피드백을 받음
     evaluation

lab(se);

Software Process

- Umbrella Activities
  - complements Software Engineering framework activities
  - applied throughout project to help software development team manage and control progress, quality, change, and risk

1. Software project tracking and control
2. Risk management
   resource 대비 spec이 과하게 높을경우 해당 스펙에 못미치는 부분에 대해 대처
3. Software quality assurance
   standard에 맞게 평가하여 레벨 체크
4. Technical reviews
   코드를 보는 등의 리뷰하고 마이너한 에러의 경우 넥스트
   메이저한 에러의 경우 수정후 넥스트
5. Measurement
6. Software configuration management
   형상관리
7. Reusability management
8. Work product preparation and production
   coding / modeling / documentation / logging

lab(se);

Software Engineering Practice

- Practice:
  - set of concepts, principles, methodologies, and tools to consider when software developments are planned and implemented
  - represents specific details

- The Essence of Software Engineering Practice
  - by George Polya (1945)

  1. **Understand the problem** (communication and analysis)
  2. **Plan a solution** (modeling and software design)
  3. **Carry out the plan** (code generation)
  4. **Examine the result for accuracy** (testing and quality assurance)

lab(se);

1. Understand the Problem

- Who are the stakeholders?
- What are the unknowns?
  - What data, functions, and features are required to properly solve the problem?
- Can the problem be compartmentalized?
  - Is it possible to represent smaller problems that may be easier to understand? (Divide & Conquer)
- Can the problem be represented graphically?
  - can an analysis model be created?

2. Plan the Solution

- Have you seen similar problems before?
  - are there patterns that are recognizable in a potential solution?
  - is there existing software that implements the data, functions, and features that are required?
- Has a similar problem been solved?
  - if so, are elements of the solution reusable?
- Can subproblems be defined?
  - if so, are solutions readily apparent for the subproblems?
- Can you represent a solution in a manner that leads to effective implementation?
  - can a design model be created?

lab(se);

3. Carry Out the Plan

- Does the solution conform to the plan?
- Is source code traceable to the design model?
- Is each component part of the solution provably correct?
  - have the design and code been reviewed?
  - have the correctness proofs been applied to the algorithm?

4. Examine the Result

- Is it possible to test each component part of the solution?
- Has a reasonable testing strategy been implemented?
- Does the solution produce results that conform to the data, functions, and features that are required?
- Has the software been validated against all stakeholder requirements?

General Principles of Software Engineering

1. The Reason It All Exists
   맞는 platform , feature를 선택
   쇼핑몰에 7천만원 서버는 ㄴ에러
   - software system exists to provide value to its users?
2. KISS (Keep It Simple, Stupid!)
3. Maintain the Vision (Project & Product)
4. What you Produce, Others Will Consume
   개발자가 계속 maintaining 할 수 있진 않음
5. Be Open to the Future (changeability & extendability)
6. Plan Ahead for Reuse
7. Think!

lab(se);

Software Development Myths

- **Management** myths
  - we have all the standard and procedures for software developments
  - if we get behind schedule, we can hire more programmer to catch up
  - if we decide to outsource the software project to a third party, we can just relax and let that firm build it.

Software Development Myths

- **Customer** myths
  - A general statements of objectives is sufficient to begin writing programs- we can fill in the details later.
  - Software requirements continually change, but change can be easily accommodated because software is flexible.

Software Development Myths

- **Practitioner's** myths
  - Once we write the program and get it to work, our job is done
  - Until I get the program running, I have no way of assessing its quality
  - The only deliverable work product for a successful project is the working program.
  - Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.