Introduction

Spring 2016

Sungwon Kang

Table of Contents

1. Software Engineering

[Simon 96]p.111

"Historically and traditionally, it has been the task of the science disciplines to teach about natural things:

how they are and how they work.

It has been the task of engineering schools to teach about artificial things:

how to make artifacts that have desired properties and how to do design."

Herbert Simon, *The Sciences of the Artificial*, 3rd Ed., MIT Press, 1996.

Engineering (1/2)

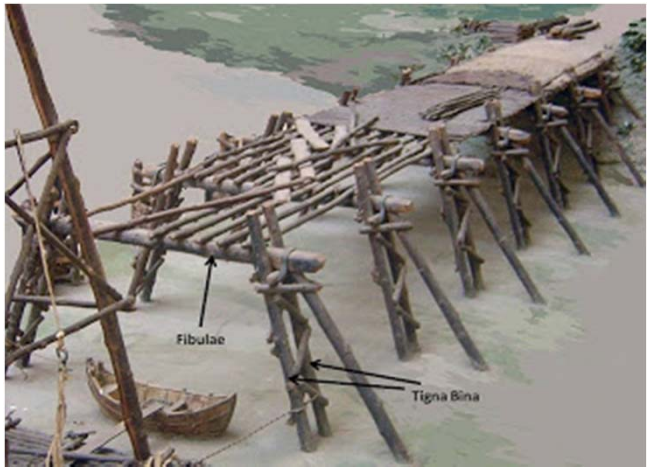
- **Civil Engineering** "is a ... engineering discipline that deals with the design, construction, and maintenance of the ... works like roads, bridges, canals, dams, and buildings." [Wikipedia]
- The 2nd oldest engineering discipline

Past

Present

Engineering(2/2)

- **Military engineering** is "the art and practice of designing and building military works and maintaining lines of military transport and communications." [Wikipedia]
- The oldest engineering discipline



Caesar's Rhine Bridge

Past

Present

Software Engineering

- More and more systems are software controlled.
- Expenditure on software represents a significant fraction of GNP in all developed countries.
- The economies of ALL developed nations are dependent on software.

☛ We are surrounded by software and software controlled systems.

☛ Those who make and utilize them well will succeed.

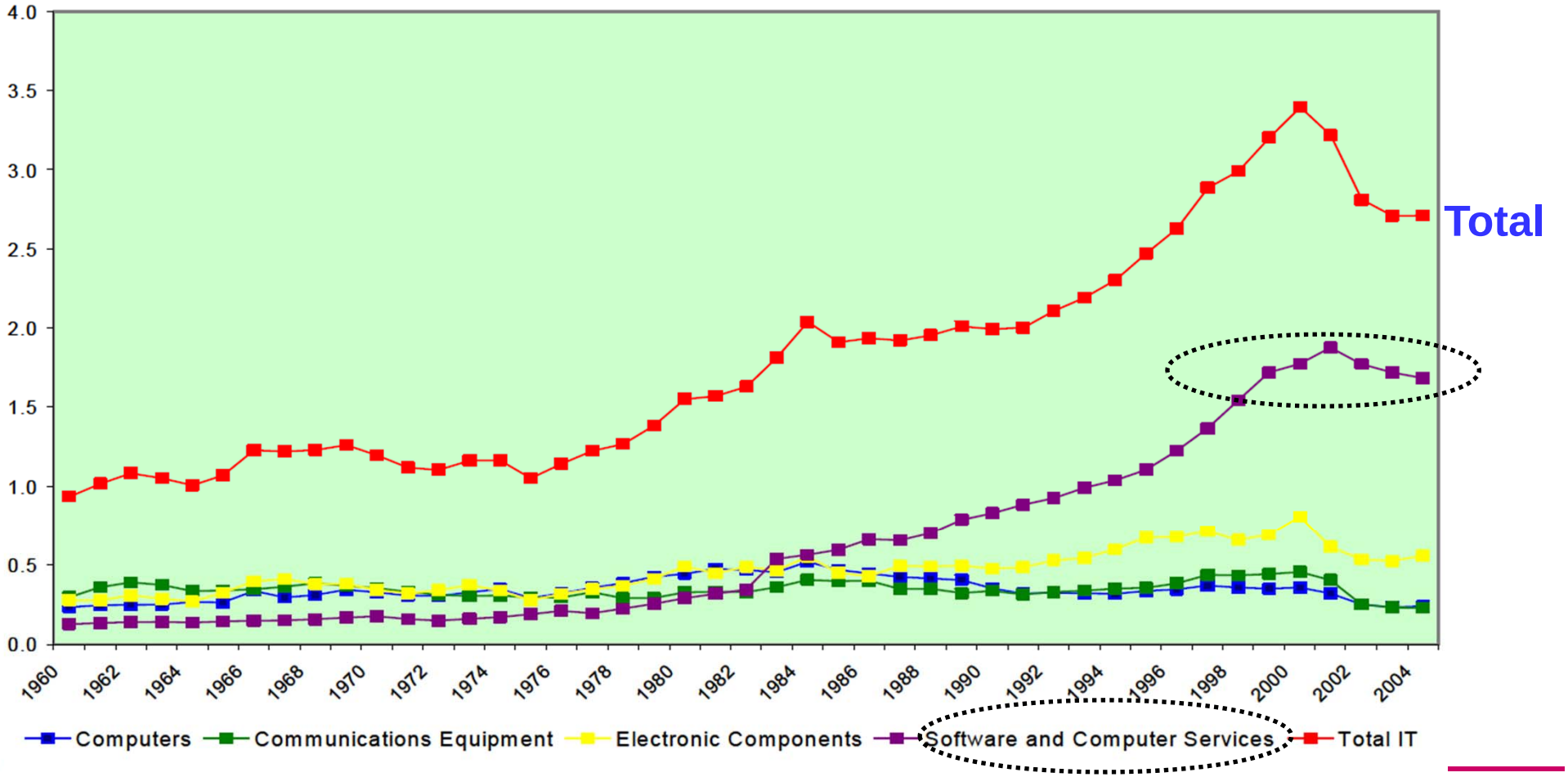
- ***Software engineering*** is concerned with theories, methods and tools for professional software development. (By teams !)

- **Definition (IEEE Standard 93):**

  "The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software."

2. Growing Importance of Software

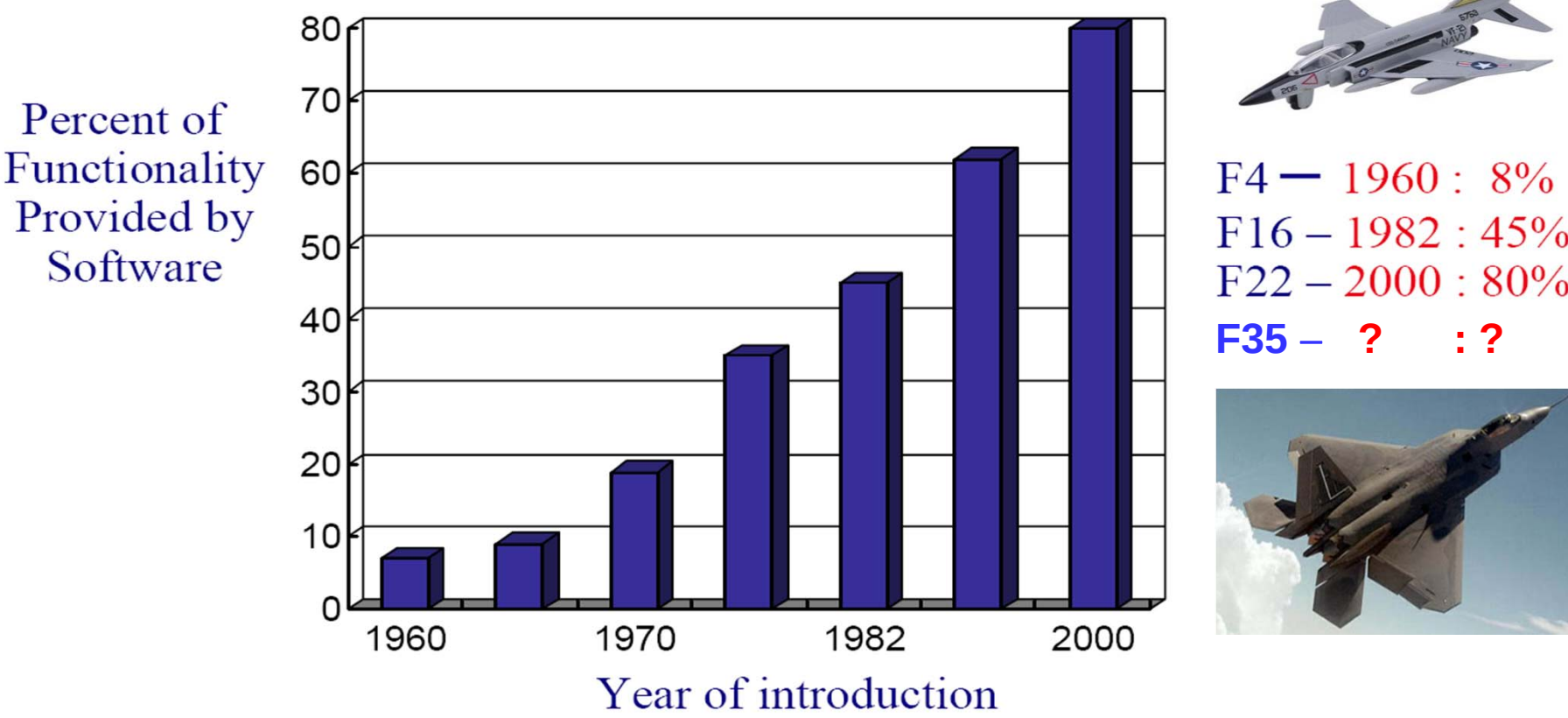
Value added Shares of Information Technology by Type
1960–2004



Total

Note: Share of current dollar GDP. Source: Jorgenson, Ho, Samuels, and Stiroh (2006b)

Fighter Airplane



F4 — 1960 : 8%
F16 – 1982 : 45%
F22 – 2000 : 80%
F35 – ? : ?

(Source: Humphrey, 2003)

Automobile

- Increase of dependency on software of functionality of automobile
  - **"By 2010, 90% of new innovations in the car will be electric/electronic while 80% are software innovations,"** A.T. Kearny, 2001.

  [Reuse of Software in Distributed Embedded Automotive Systems." Audi. 2004.]

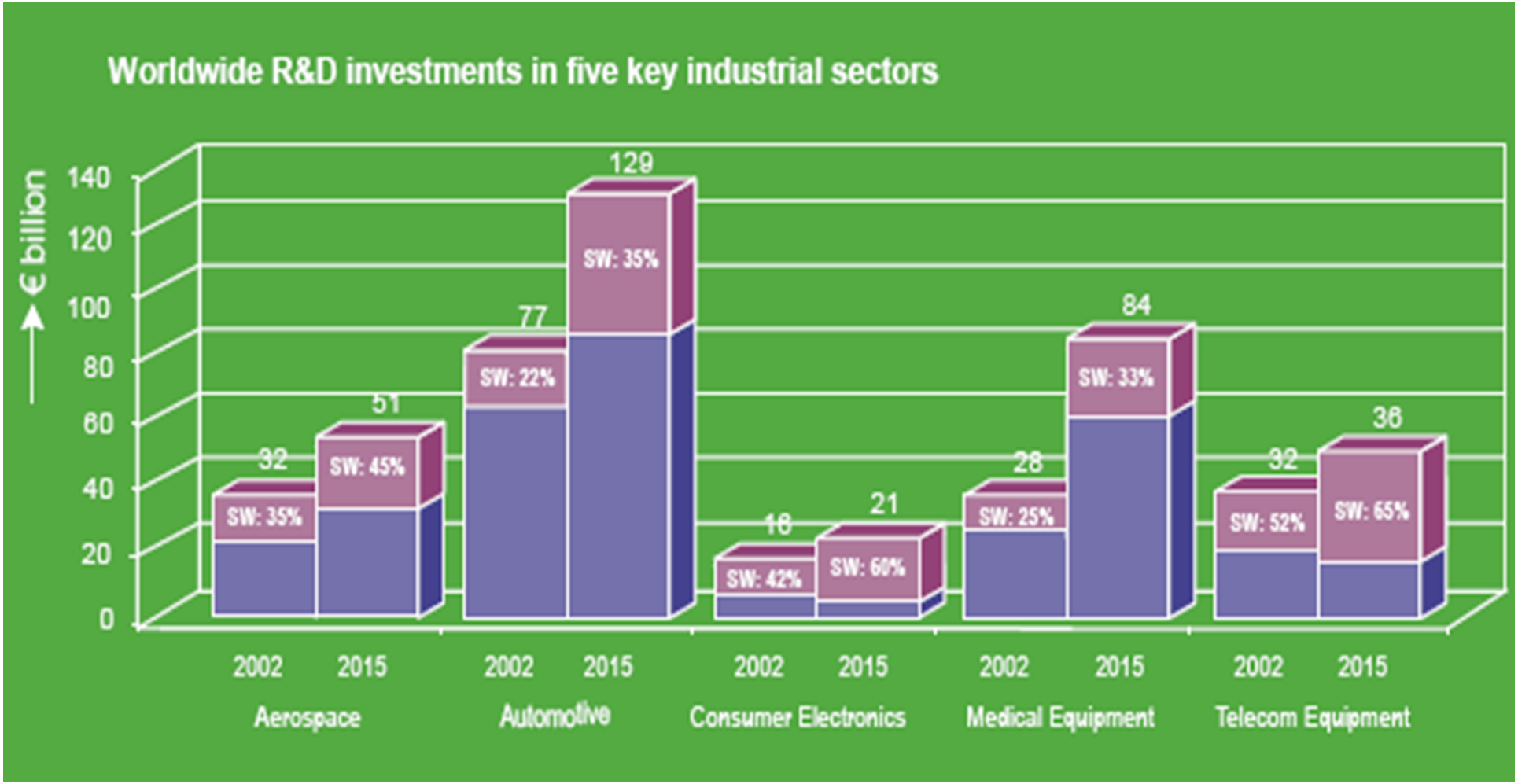
  [5 Embedded Automotive Electronics Symposium. Peugeot. June 23, 2004.]

  Examples
  - Low emission engine control system
  - Hybrid power-train system
  - GPS Navigation
  - Body control, Chassis dynamics control system
  - Multimedia Entertainment System
  - Comfort & Convenience System
  - Etc...

- In 2015, software cost in electronics systems will be 40% of the cost of the car.

  [Nov 2006: Workshop on Automotive Electronics Technology – the Present and the Future of Automotive Electronics System Technology]

Software R&D



(Source: IOA, 2005)

Major Global IT Companies in 2013

	Sale	Profit	ROI	Net Profit
Samsung Electronics	228.69	36.79	16.1%	30.47
Apple	184.3	52.8	28.7%	39.23
Intel	56.8	13.3	23.3%	10.37
IBM	107.6			17.77
HP	121.1	7.7	6.4%	5.51
Microsoft	90.0	30.2	33.6%	24.60
LG Display	27.0	1.16	4.3%	0.42
LG Electronics	58.14	1.29	2.2%	0.22
SK Hynix	14.165	3.38	23.9%	2.87

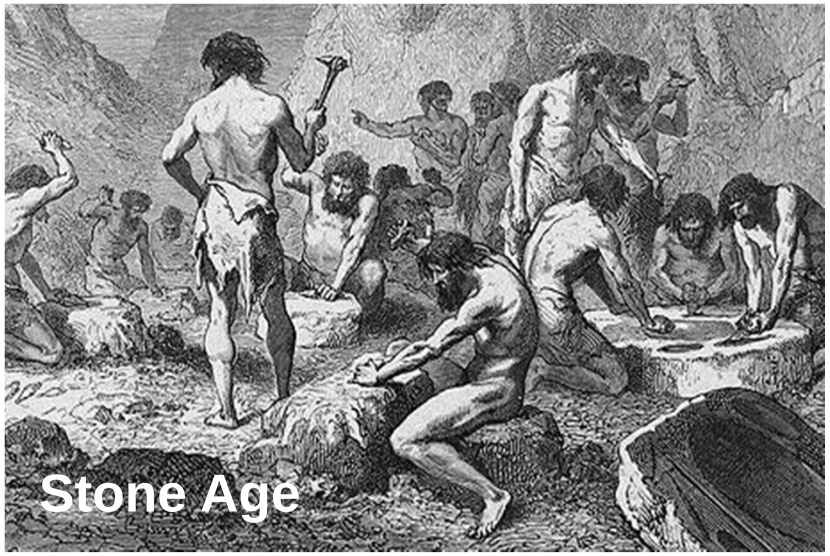
1 Billion Dollars

Change in the World ICT Industry

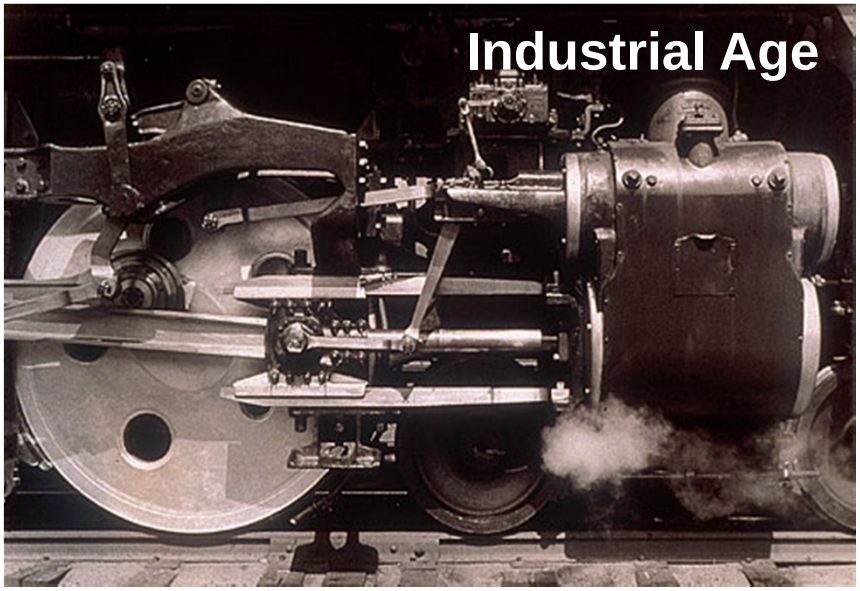
IT 산업 시가총액 세계 10대 기업 순위(1980~2010년)

	1980年		1990年		2000年		2010年	
1	IBM	HW	IBM	HW	씨스코	HW	애플	SW
2	코닥	HW	히타치	HW	마이크로소프트	SW	마이크로소프트	SW
3	휴렛 패커드	HW	파나소닉	HW	노키아	HW	구글	SW
4	파나소닉	HW	루슨트 테크놀로지	HW	인텔	HW	IBM	SW
5	소니	HW	NEC	SW	오라클	SW	오라클	SW
6	산요	HW	소니	HW	IBM	SW	인텔	HW
7	텍사스 인스트루먼트	HW	코닥	HW	EMC	HW	씨스코	HW
8	모토롤라	HW	후지쯔	SW	에릭슨	HW	삼성	HW
9	에머슨	HW	샤프	HW	텍사스 인스트루먼트	HW	휴렛 패커드	HW
10	유니시스	SW	산요	HW	루슨트 테크놀로지	HW	퀄컴	HW

[Source: Prof. Sooyoung Park, KCSE 2014 Keynote Talk]


Stone Age


Iron Age


Industrial Age

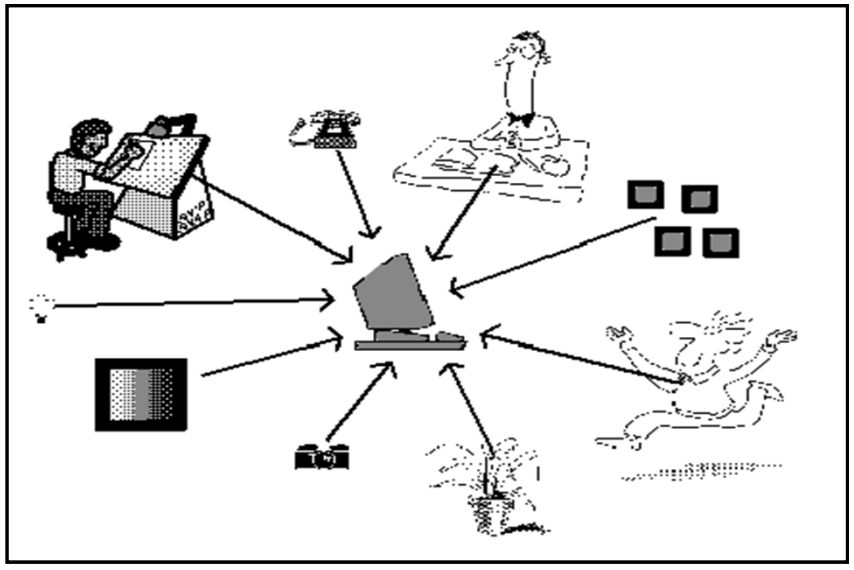

Computer and
Software Age

Our World(1/2)

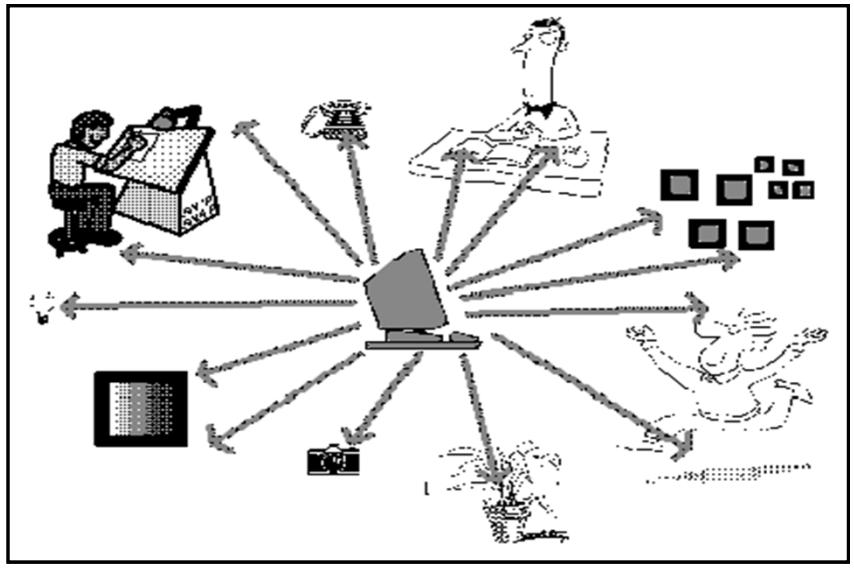
"Now we are living in personal computing era. ... Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives." 1988

Mark Weiser(1952-1999)

Virtual Reality



Embodied Reality(Ubiquitous computing)



14

Our World(2/2)


Internet
"Our civilization runs on software." - Stroustrup
WWW

My Conjecture

As those who were good at making stone and iron artifacts ruled their ages, those who are good at making computer and software artifacts will dominate our age.

- Then why don't we just make a lot of software and embed them in computer systems?

☛ It is not easy !

3. Need for software engineering skills(1/3)

- **Quality of Software**
  - 75,000 Toyota Prius hybrid cars recalled (2004 and 2005 Priuses).
    - Software Errors:
      - Warning signal goes off without turning it on (68 incidents)
      - Gas Engine abruptly stops

      [Oct. 14, 2005, Michael Kanellos, CNET News.com]
  - BMW:
    - When computer terminates, engine stops, air-conditioner stops and windows locked
    - iDrive:
      - Stalling unexpectedly and while driving at high speeds
      - Stalling unexpectedly when the fuel gauge reads < 1/3 full
      - Voice activation system malfunction
      - Cell Phones power switch
      - Etc.

3. Need for software engineering skills(2/3)

- Software defects cost
  - US: (0.6% of GDP/ year: about $60 Billion, 2002)
  - Korea:
    - Direct economic effect: $2 Billion(67% of Sale)
    - Indirect economic effect: $2.7 Billion(87% of Sale)

  [Source: “Report on Economic Impact of Reliability Support Technology“, April 2008]

- Software Reliability Cases
  - US : Pathfinder derailed due to software bug ($0.2 Billion Loss)
  - Germany: Extra cost per each car for each recall ($700)
  - Korea: In 2005, bus fare charging system error ($ 0.4 M/day)

3. Need for software engineering skills(3/3)

- June 4th 1996, Ariane 5 ($180M) was launched for its maiden flight

- A European rocket designed to launch commercial payloads (e.g. communications satellites, etc.) into Earth orbit

- Ariane 5 can carry a heavier payload than Ariane 4

Ariane 5

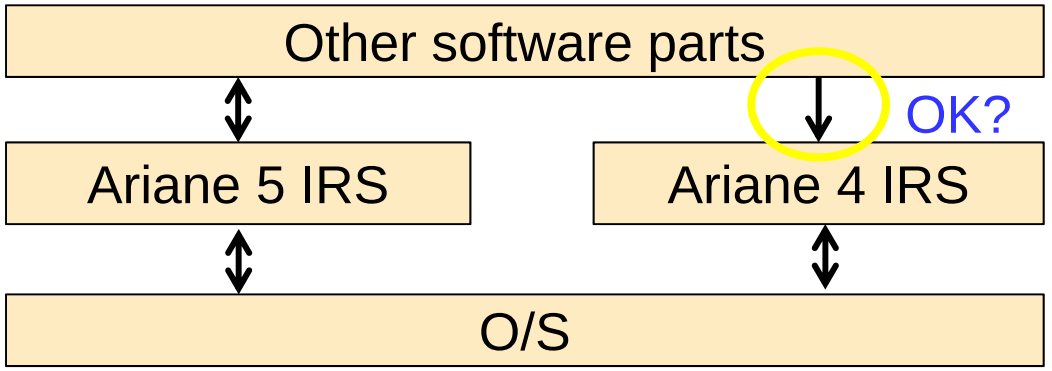
- Approximately 37 seconds after lift-off, Ariane 5 lost control.
- It started to break up and was destroyed by ground controllers.

The Problem

- The altitude and trajectory of the rocket are measured by a computer-based inertial reference system (IRS). This transmits commands to the engines to maintain altitude and direction.
- An attempt to convert a 64-bit floating point number to a signed 16-bit integer caused the number to overflow.
- There was no exception handler associated with the conversion. The software failed and this system and the backup system shut down.
- The backup software was a copy and behaved in exactly the same way.

What went wrong? (1/2)

- The software component that failed was reused from the Ariane 4 launch vehicle.



- Decisions were made
  - Not to remove the facility as this could introduce new faults (A Big Mistake !)
  - Not to test for overflow exceptions because the processor was heavily loaded. (Another Big Mistake !)

What went wrong? (2/2)

- Why not Ariane 4?
  - Ariane 4 was a smaller vehicle such that it has a lower initial acceleration and build up of horizontal velocity than Ariane 5.
  - The value of the variable on Ariane 4 could never reach a level that caused overflow during the launch period.

Design Failure

- The designers of Ariane 5 made a critical and elementary error.

  ☛ They designed a system where a single component failure could cause the entire system to fail.

What should they have done?

Review Failure

- The design and code of all software should be reviewed for problems during the development process

- Either

  the IRS software was not reviewed because it had been used in a previous version

  or the review failed to appreciate the consequences of system shutdown during a launch

Testing Failure

- As the facility that failed was not required for Ariane 5, there was no requirement associated with it.

  => There were no tests of that part.

The story reveals either the *lack of application of software engineering* or *an unsound reasoning in software engineering*.

• For more of software horror stories
http://www.cs.tau.ac.il/~nachumd/horror.html

Other Failure Cases

- Company A requested Company B to build a system. A added new requirements in the middle of development period and also extended deadline. Fair Enough ?

  B could not finish the system within the new deadline. A asked for money back since the project failed.

- Moreover, A asks B to make up the loss caused by failed project.

- Company A was building a software system. In the middle of project, key technology people left the company.

- B passed 90 % of the acceptance test. Has B succeeded or not?

Status of Software Projects

- Standish Group (1994)
  - US IT Application Development cost
    = \$250 Billion = 250 x 10^{12} Korean Won
  - No. of projects = 17500
  - 0.5 Million < Cost per project < 2.5 Million
- Rules of Thumb
  - 1/3 projects cancelled before completion
  - 1/2 projects will cost about twice the original estimate
  - 1/2 projects will take another year to complete

=> Over time, over budget, quality below expectation

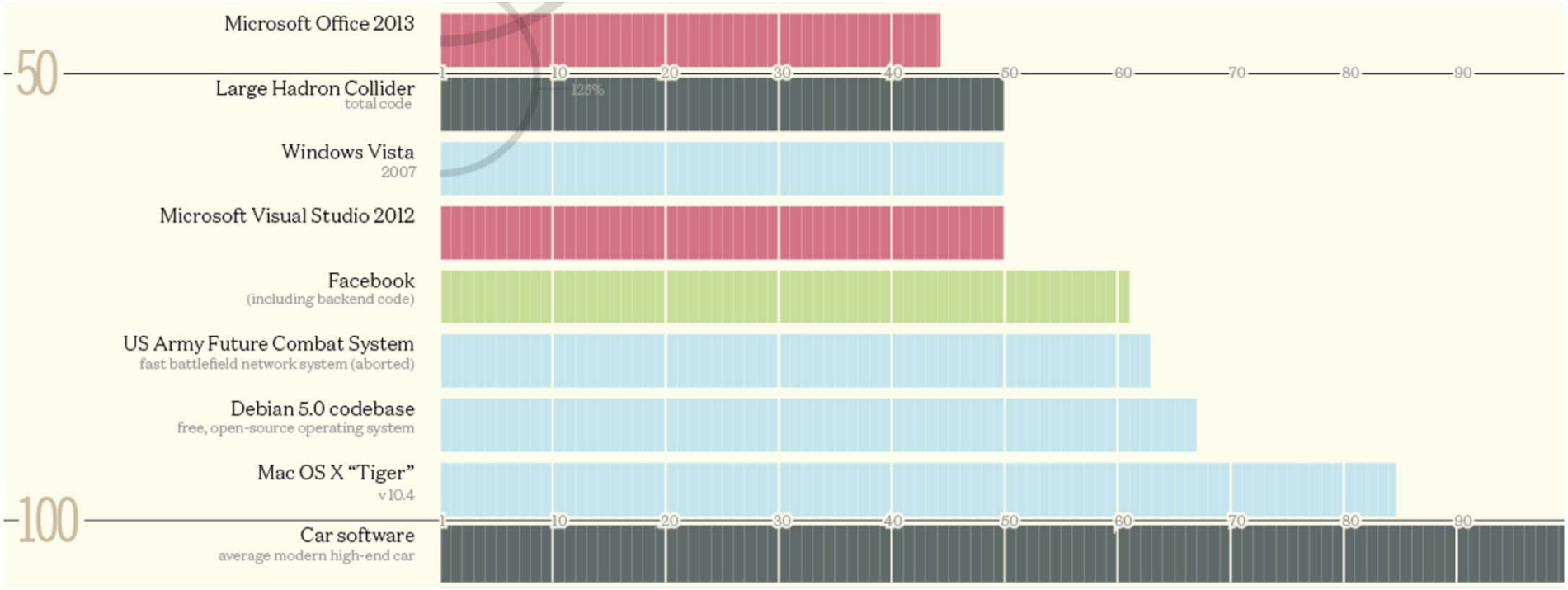
- **Only 28% of all the projects in the world are successful**
  Standish Group = 'Chaos Report 2001'

Size of Today's Software

http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

Codebases

Unit: Million Lines of Code



Conclusion

- Software engineers have great opportunities !
  - To make contribution
  - To be well-paid

4. Value of Software Engineering

Best Jobs in America

2013

CNNMoney/PayScale's top 100 careers with big growth, great pay and satisfying work.

1		Biomedical Engineer	$87,000	61.7%
2		Clinical Nurse Specialist	$86,500	26%
3		Software Architect	$121,000	27.6%
4		General Surgeon	$288,000	24.4%
5		Management Consultant	$110,000	29.1%
6		Petroleum Geologist	$183,000	21.2%
7		Software Developer	$88,700	27.6%
8		IT Configuration Manager	$95,800	28.5%

12. User Interface Engineer
$92,100 27.6%

14. Database Administrator
$88,800 30.6%

15. Video Game Designer
$72,000 27.6%

22. Applications Engineer
$87,000 27.6%

Source:
http://money.cnn.com/pf/best-jobs/

Case 1: Software Product Line

Aspect	Improvement
Quality	52%
Cost	45%
Productivity	39%
Customer Satisfaction	39%
Time to market Reduction	30%

[Cohen 02] Sholom Cohen, *Product Line State of the Practice Report*, Technical Note CMU/SEI-2002-TN-017, September 2002

Case 1: Software Product Line

- DOD: Cost: 50% ↓

  Labor: 75% ↓

- CelsiusTech Systems AB (Sweden)
  - Software cost out of system cost reduced from 65% to 20%
  - Code reuse 80%

Cummins SSPL ROI Analysis

Table 5: Five-Year Development and Maintenance Costs of Monitoring Products

Program Name	Lines of Code	Development Cost (in millions)	Maintenance Cost per Year (in millions)	Number of Years in Maintenance	Total Cost (including Development and Maintenance) (in millions)
Small vehicle	120000	12.0	3.0	4.00	24.00
Large vehicle	140000	14.0	3.5	4.00	28.00
Small truck	126000	12.6	3.15	3.00	22.05
Large truck	150000	15.0	3.75	3.00	26.25
SUV	150000	15.0	3.75	3.00	26.25
New 1	130000	13.0	3.25	2.00	19.50
New 2	130000	13.0	3.25	2.00	19.50
New 3	130000	13.0	3.25	2.00	19.50
Assets	100000	12.0	3.00	4.00	24.00

Before

[Cohen, S.,"Product Line State of the Practice Report," CMU/SEI-2002-TN-017, September, 2002]

Cummins SSPL ROI Analysis

After

Program Name	Cumulated Cost Without Reuse (in millions)	Product Line Savings (50% from assets, 25% cost)	Cumulated Savings with Product Lines	Cumulated Product Line Costs	Cumulated Savings over 5 years (in Millions)
Small vehicle	24.00	9.00	9.00	12.00	-3.00
Large vehicle	52.00	10.50	19.50	12.00	7.50
Small truck	74.05	8.27	27.77	15.00	12.77
Large truck	100.30	9.84	37.61	15.00	22.61
SUV	126.55	9.84	47.46	15.00	32.46
New 1	146.05	7.31	54.77	18.00	36.77
New 2	165.55	7.31	62.08	18.00	44.08
New 3	185.05	7.31	69.39	18.00	51.39

[Cohen, S.,"Product Line State of the Practice Report," CMU/SEI-2002-TN-017, September, 2002]

Case 2: Model Driven Development(MDD)

- **Company:** CGI

  (Big 4 IT services Co. in North America)

- **Business:** End-to-end IT services and Business Solutions

- **Employees:** 13,700

- **Sale:** $130 M

Cost when MDD not used:	$ 8.7 M
Cost when MDD used:	$ 5.47 M
Reduced Cost:	$ 2.59 M
Reduction Rate:	32 %
MDD Investment:	$ 0.37 M
ROI:	700%

Case 3: Return on Investment(ROI)

- Personal Software Process: 260 ~ 1300 Times

  Rico, David F., "Personal Software Process (PSP): An Executive Overview," http:// davidfrico. com (2000)

- Software Inspection Process : 133 Times

  Rico, David F., "Software Inspection Process: An Executive Overview," http:// davidfrico. com (2000)

- CMM Level 5: 83 Times

  McGarry, Frank, "Experiences in Attaining Level 5 Process Maturity at CSC," Washington D. C. Society for Software Quality (SSQ) (June 2000)

5. Course Mechanics

- How the course proceeds:

- **See the course syllabus for details:**

- The course material is based on Pressman's textbook.
- The course is project-based.
- We have two TAs who will be your project mentors.
- You will work as a team of 3 members to do the project.
- Each week team has to meet with the team's mentor for progress report and have Q/A session.
- At the end of the semester, each team will deliver required deliverables as required in the course syllabus.
- There will be two presentations: MOSP and EOSP.

"Software engineering cannot be taught but can be learned."
- UC Berkeley Software Engineering Course Material
(글로벌 소프트웨어를 꿈꾸다 - 김익환)

☛ "Learning by Doing"

6. FAQs about Software Engineering

A) What is software?

B) What is software engineering?

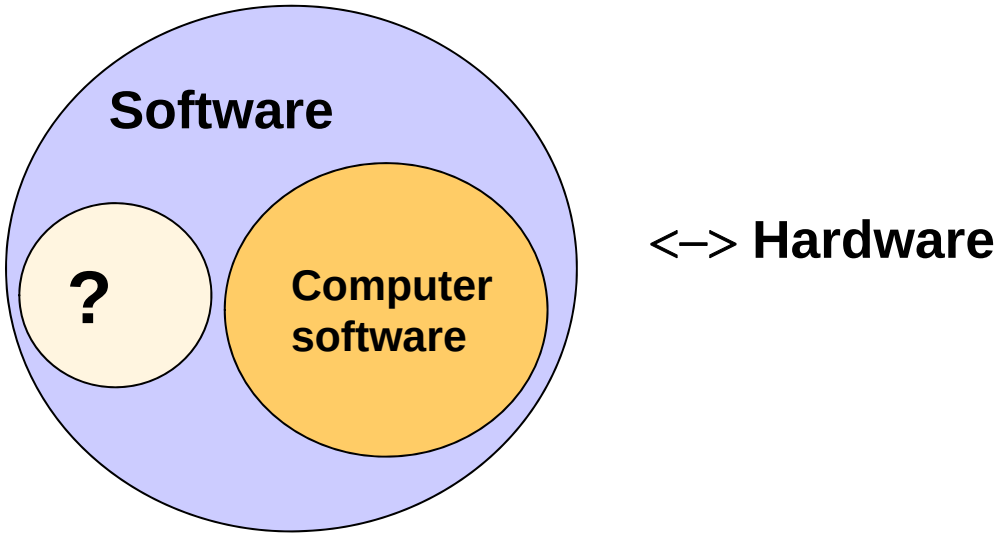
C) What is the difference between software engineering and computer science?

D) What is the difference between software engineering and system engineering?

A) What is Software?

- Computer programs and associated documentation such as requirements, design models and user manuals.

- Software products may be
  - **Generic** (= **Package**) - developed to be sold to a range of different customers: E.g. PC software such as Excel or Word.
  - **Custom** - developed for a single customer according to their specification.



B) What is Software Engineering?

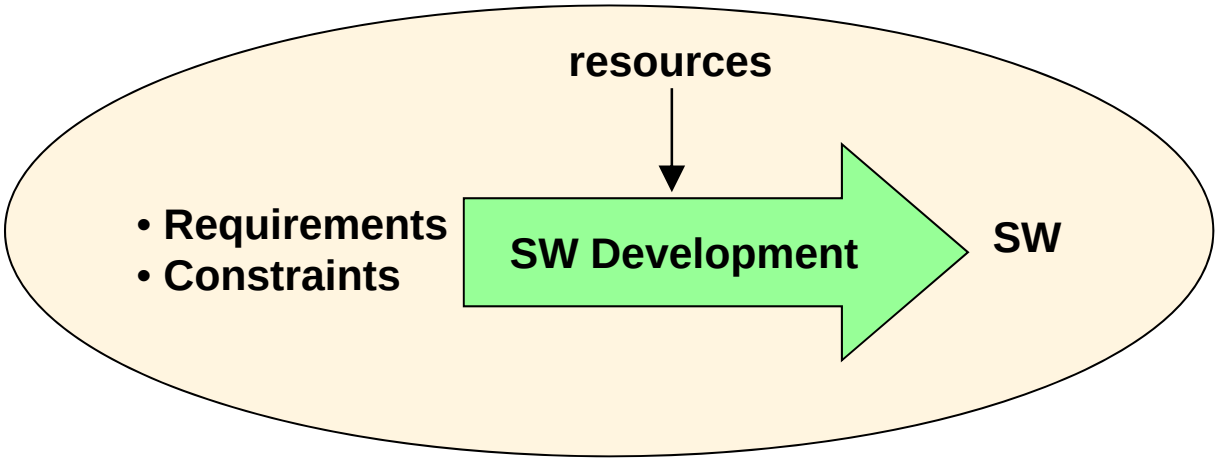
- Software engineering is an engineering discipline that is concerned with all aspects of software production.

- Software engineers *should*

  - adopt a systematic and organised approach

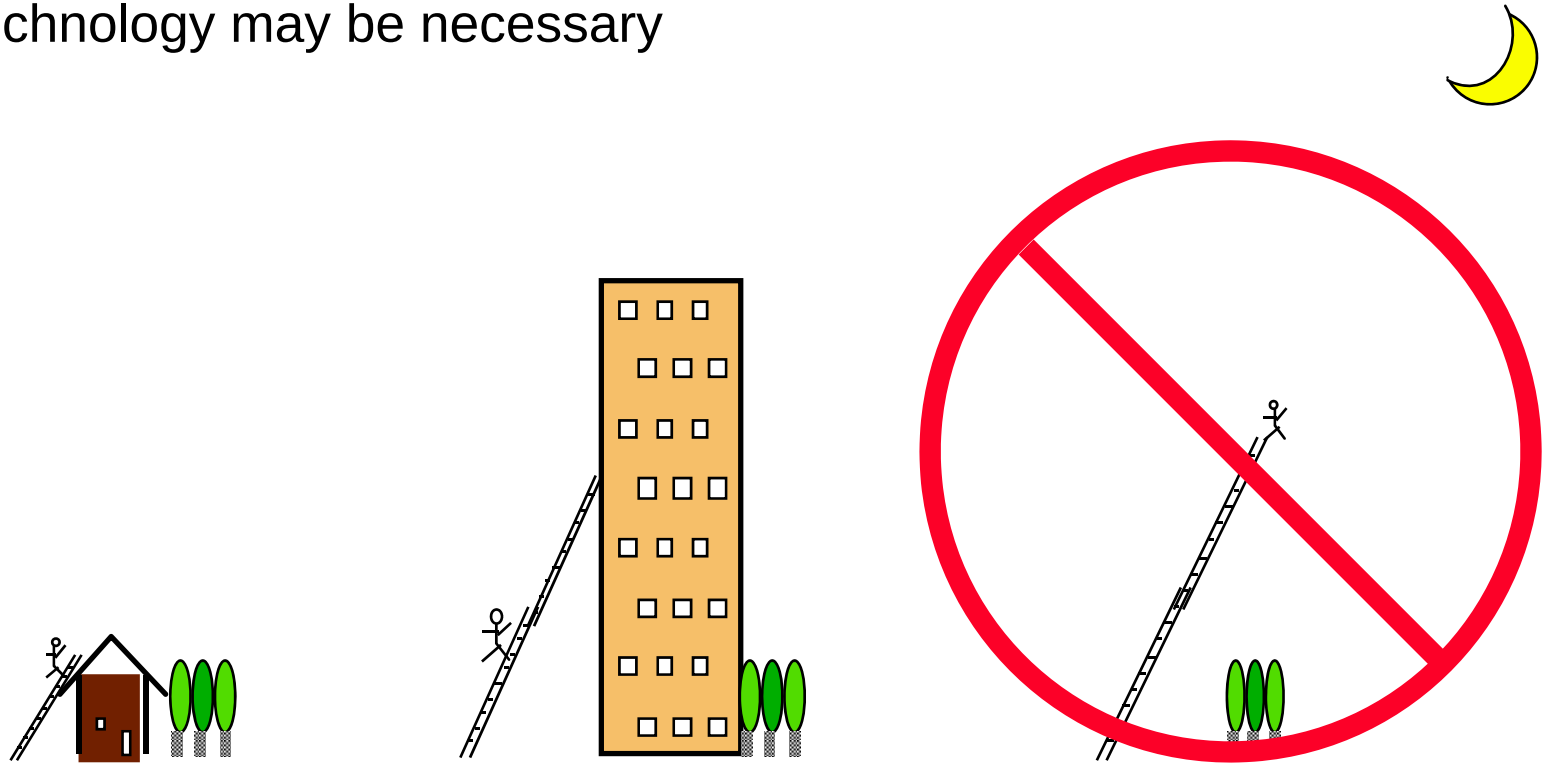
  - use appropriate tools and techniques depending on the problem to solve, the development constraints and the resources available.

Engineering



Aren't Programming Languages Good Enough?

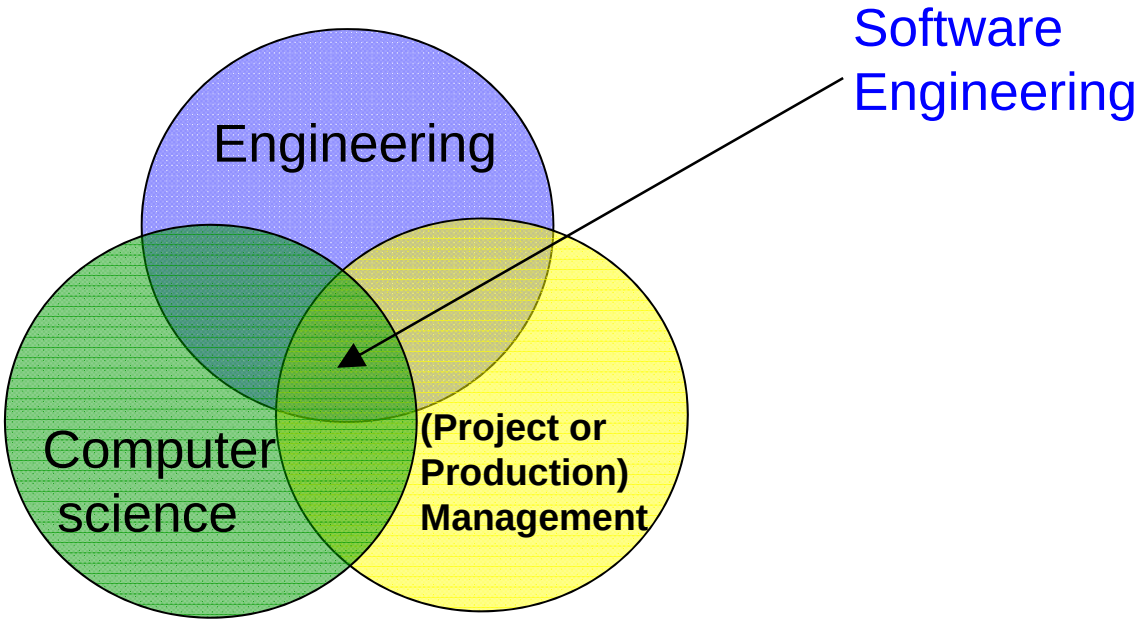
When orders-of-magnitude improvement are required, new technology may be necessary



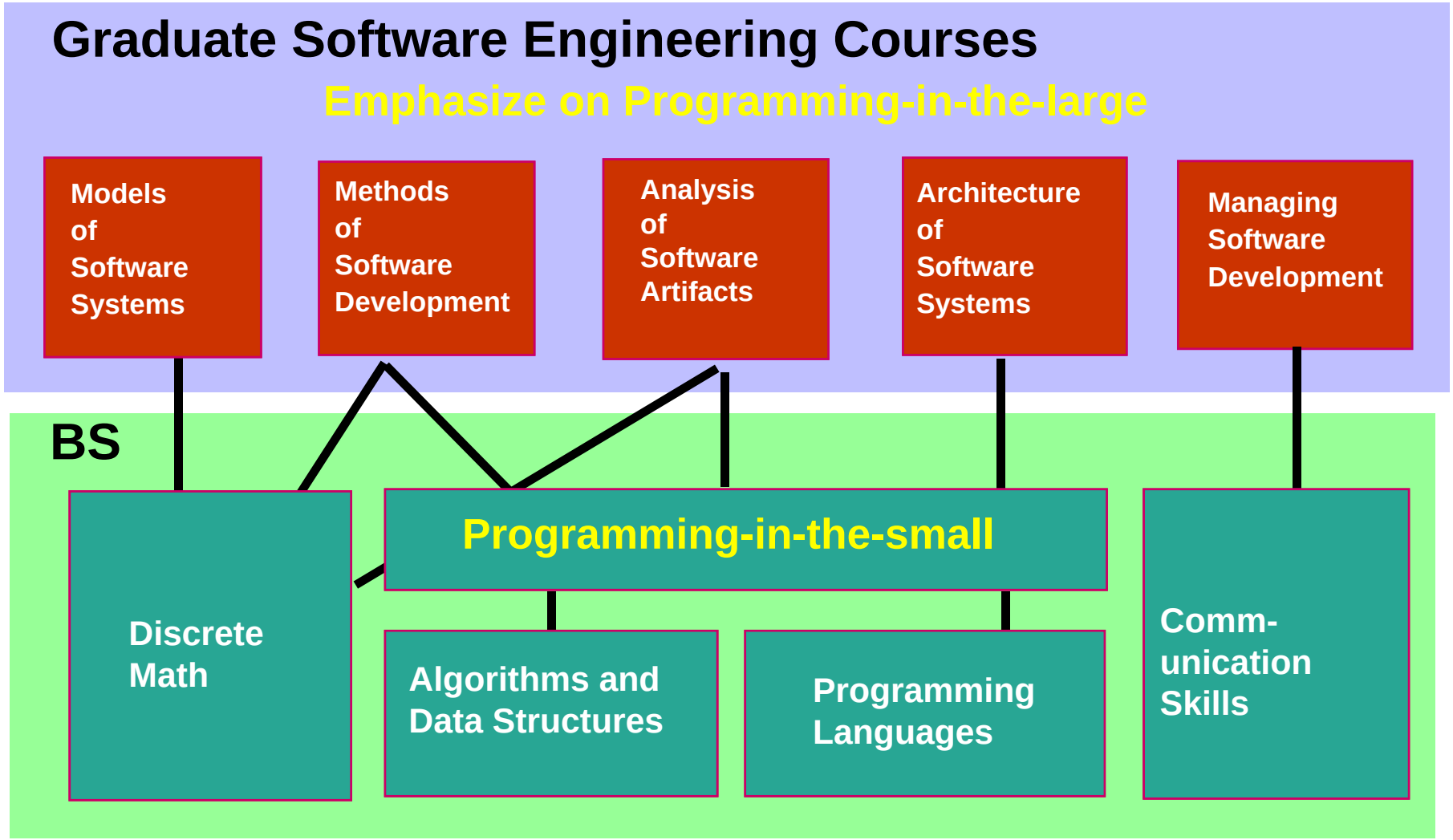
- Slide from Dr. David Garlan's Architecture Course

C) Software Engineering and Computer Science

- Computer science is concerned with theory and fundamentals, domain specific algorithms and data structures
- Software engineering is concerned with the practicalities of developing and delivering useful software regardless of domain.



Software Engineering and Computer Science

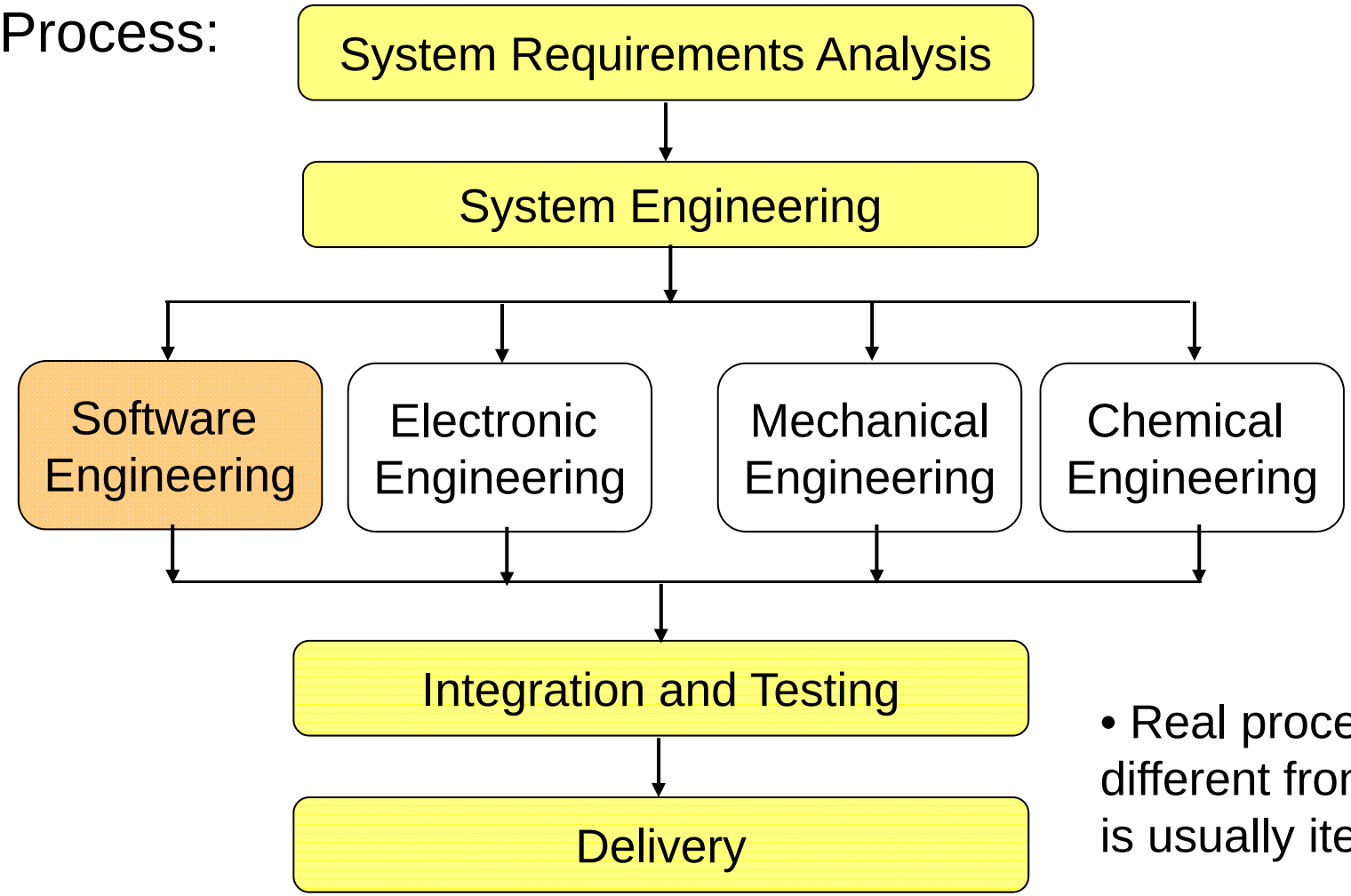


D) Software Engineering and System Engineering

- Webster's Dictionary: "System is a set of arrangement of things so related as to form a unity or organic whole."
- System engineering is concerned with all aspects of computer-based systems development including
  - Hardware aspect
  - Software aspect
  - process aspect
- System engineers are involved in system specification, architectural design, integration and deployment.
- "System" is a term that hardware people used first before software people used for "software system", which is synonymous with "software".

System Engineering

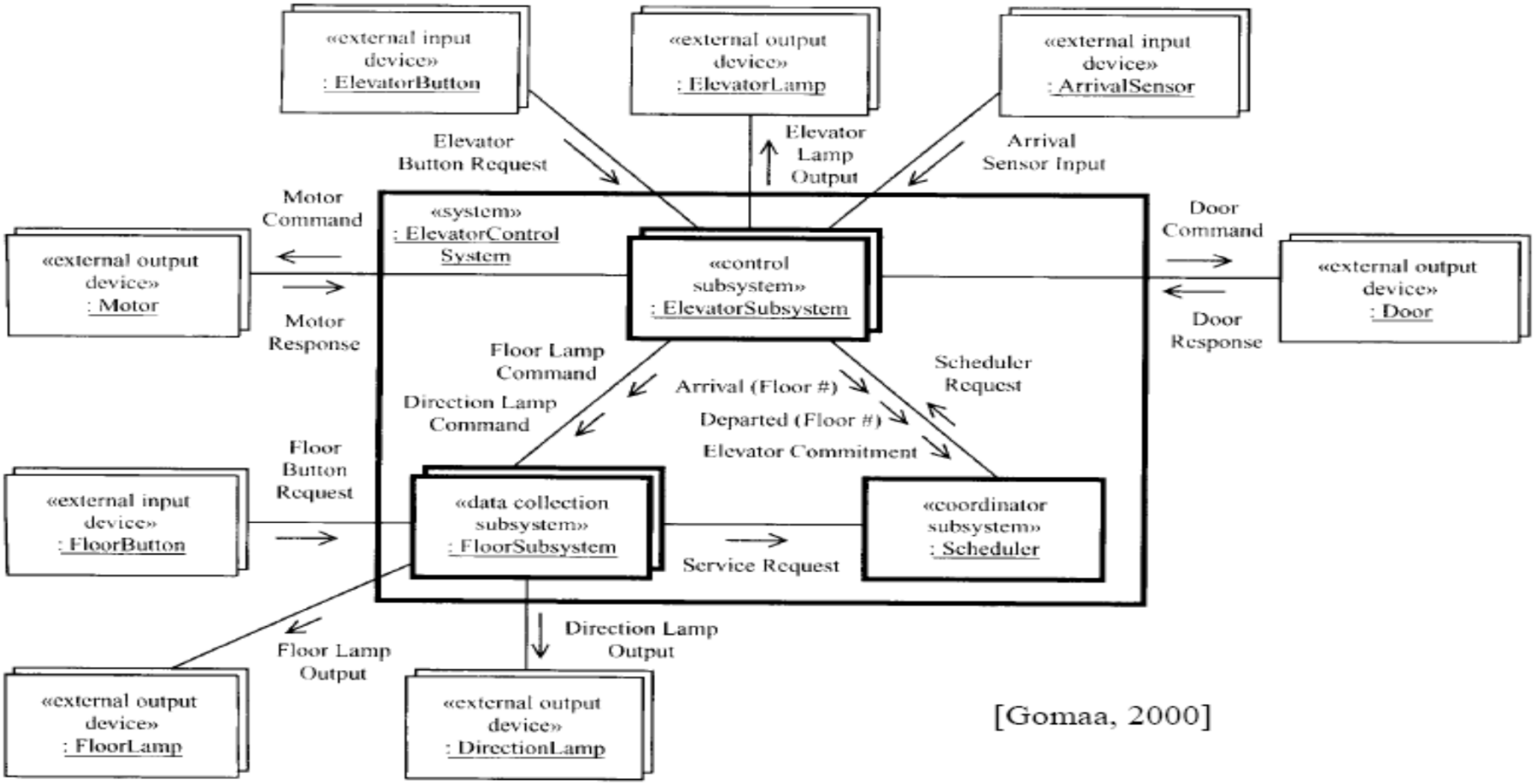
Process:



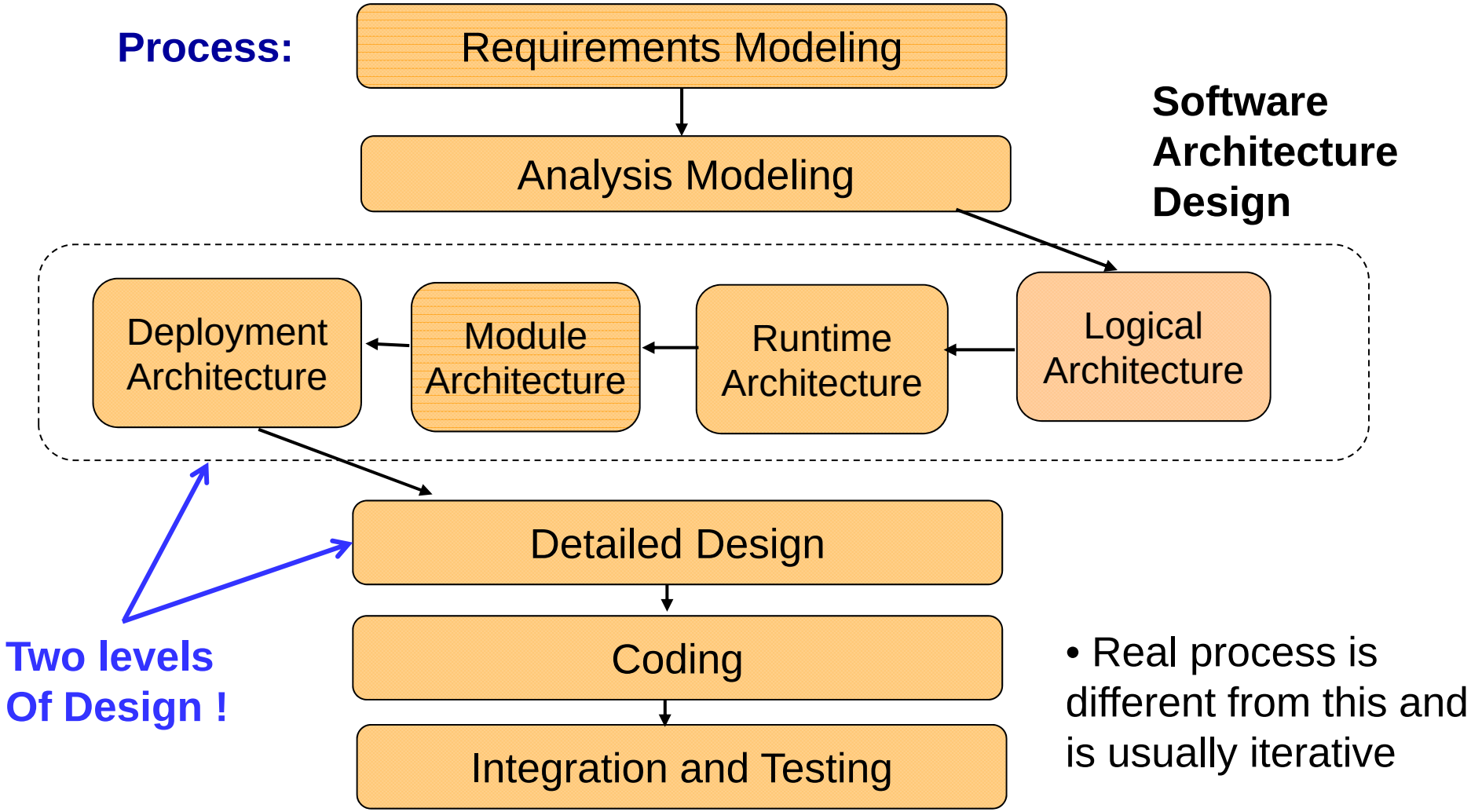
- Real process is different from this and is usually iterative

System Engineering

- **System Architecture Design (Elevator System Example)**



Software Engineering



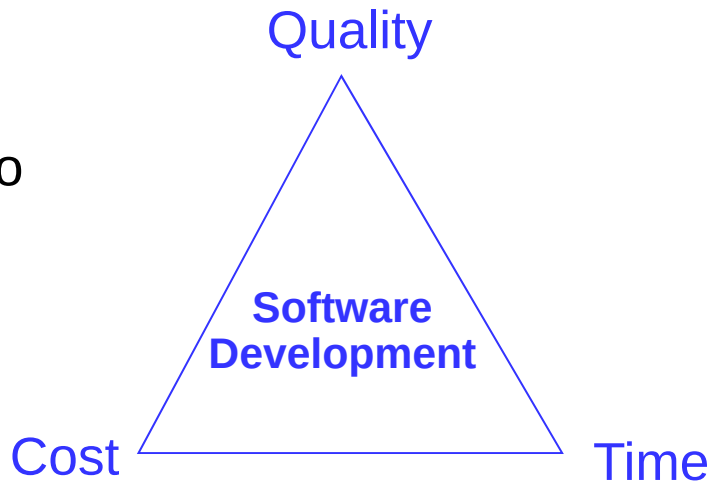
Software Engineering

- Software engineering can be part of system engineering

- Software engineering is concerned with developing
  - Software infrastructure
  - Control
  - Applications
  - Databases

Key Challenges Facing (Software) Production

- Quality
  - Developing techniques that demonstrate that software can be trusted by its users
- Time-to-Market
  - Developing techniques that lead to faster delivery of software
- Cost
  - Develop techniques that minimize development cost
- Interoperability
  - Developing techniques that can cope with heterogeneous platforms and execution environments



7. Professional and Ethical Responsibility

- 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.

Issues of Professional Responsibility

- Confidentiality
  - Engineers should respect confidentiality of their employers or clients
- 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. 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, ranging from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains 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.

Code of Ethics - Principles

1. Public interest
   - Software engineers shall act consistently with the public interest.
2. Client and employer interest
   - 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 quality
   - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. Conscientious Judgment
   - Software engineers shall maintain integrity and independence in their professional judgment.

Code of Ethics - Principles

5. Management ethics
   - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. Advance the profession
   - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. Teamwork with colleagues
   - Software engineers shall be fair to and supportive of their colleagues.
8. Continuous self improvement
   - 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.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of illegal military weapons systems or nuclear systems.

Questions ?