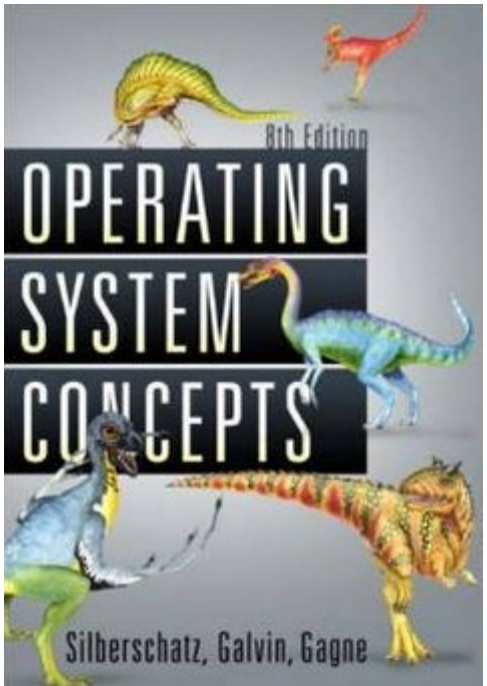
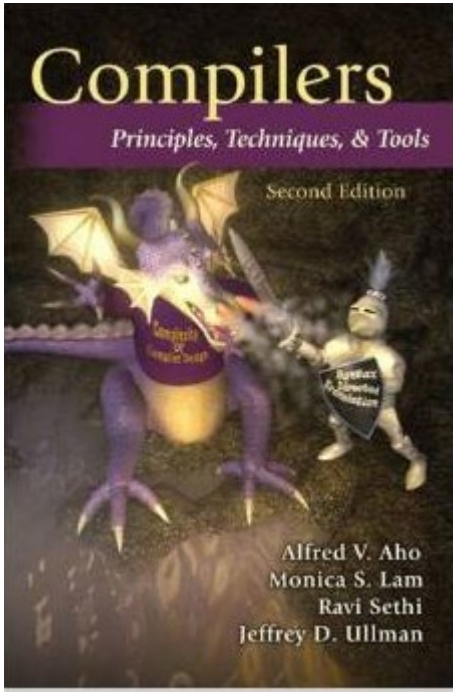
Intro. To Quality Software
- Fight the Complexity of SW

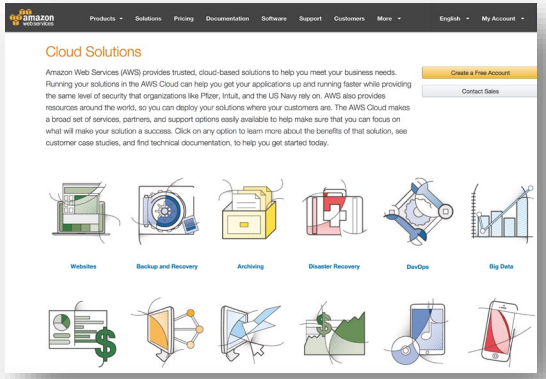
Moonzoo Kim

School of Computing. KAIST





What is Software?



https://mashable.com/article/how-to-hide-apps-iphone

- Computer *programs (instructions)* that provide desired features and functionality
- Associated *documentation* that describes the operations and use of the programs

[PrMa20]

Where is Software?

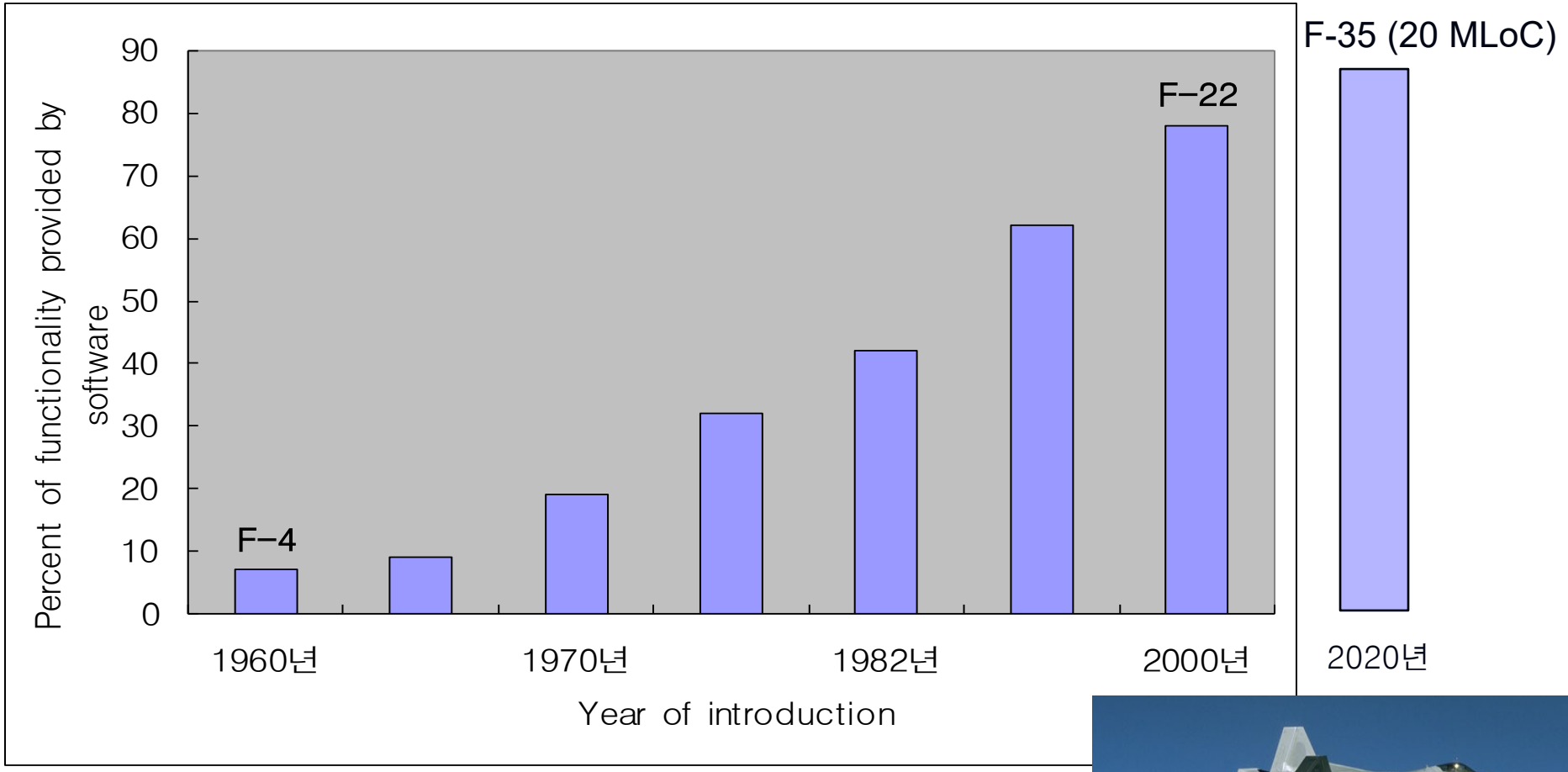
Everywhere!

Types of Software

- System software
- Embedded software
- Application software
- Engineering/scientific software
- Product-line software
- Cloud/Web/Mobile software
- AI software (robotics, neural nets, game playing)

[PrMa20]

Role of S/W: Increased in Everywhere



자료출처: Watts Humphrey 2002

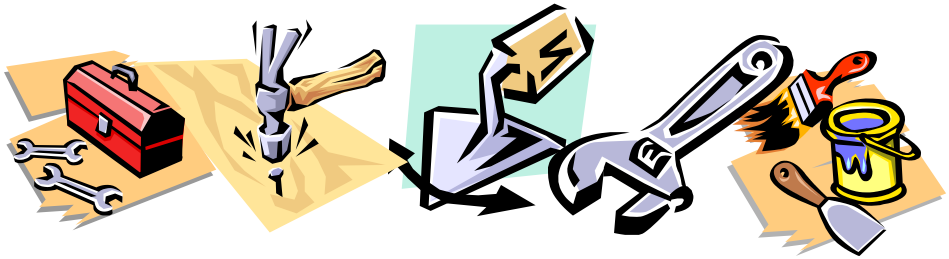
KAIST

Analogy of SE with Civil Engineering

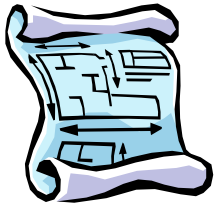
There are various kinds of construction from a house to a building complex

Analogy with Civil Engineering(Cont.)

When building a small house,...

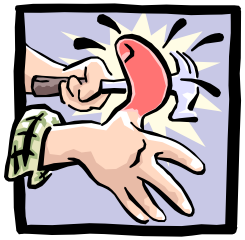


With simple tools



From just a blueprint

By small group of novice workers



Some failures are endurable

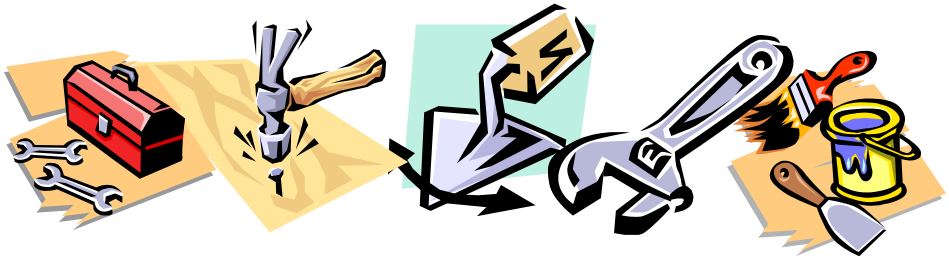
Analogy with Civil Engineering(Cont.)

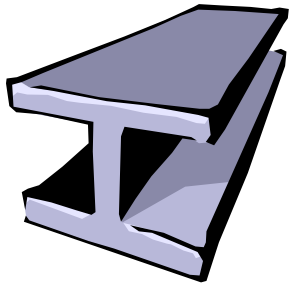
When constructing a building complex,...
we must re-think everything

Analogy with Civil Engineering(Cont.)

With scalable tools

as well as simple ones

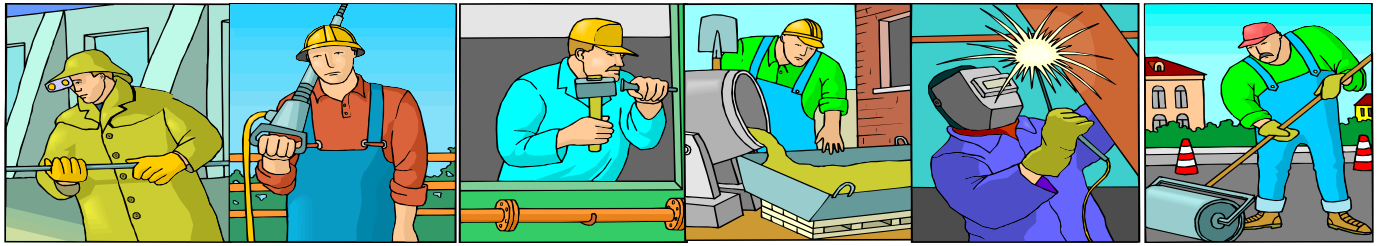




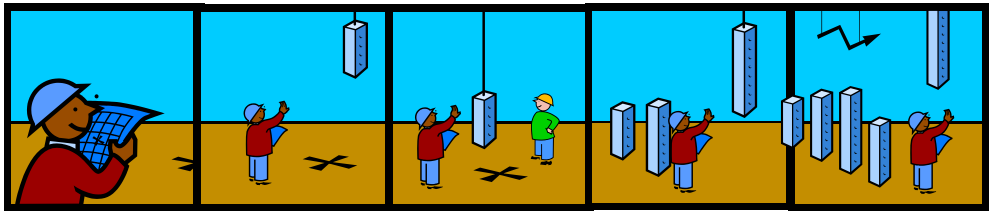
Use different materials

Analogy with Civil Engineering(Cont.)

By a big group of various technicians



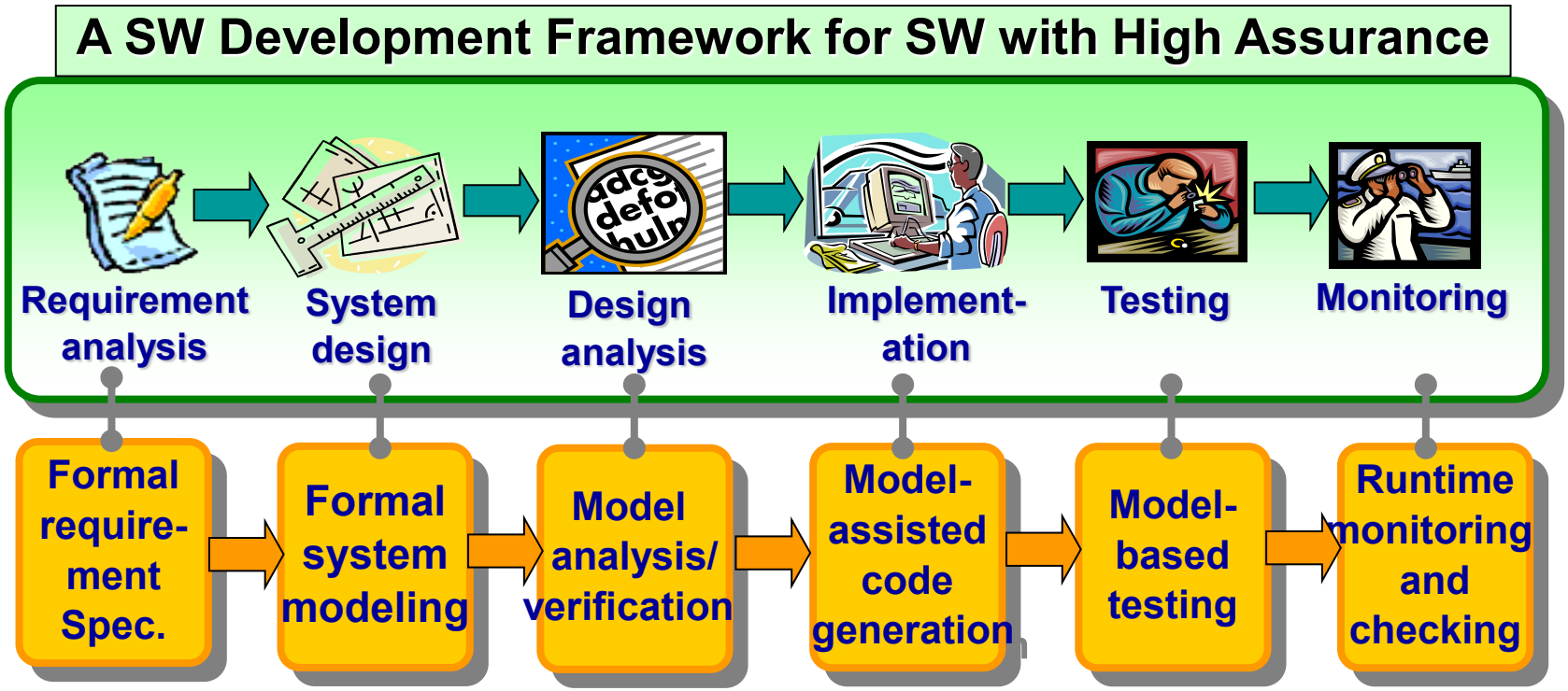
With collaborations and guides



Set of blueprints As well as careful plans

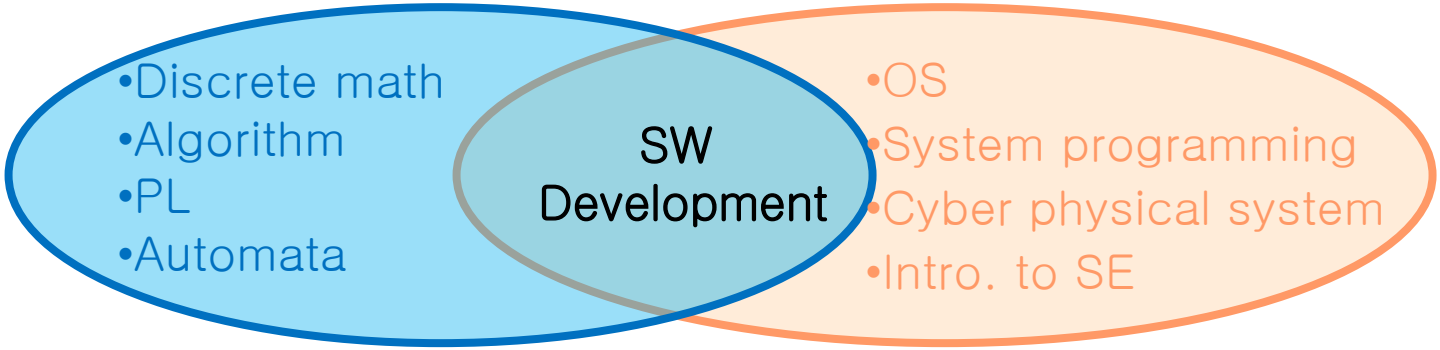
Software Development Cycle

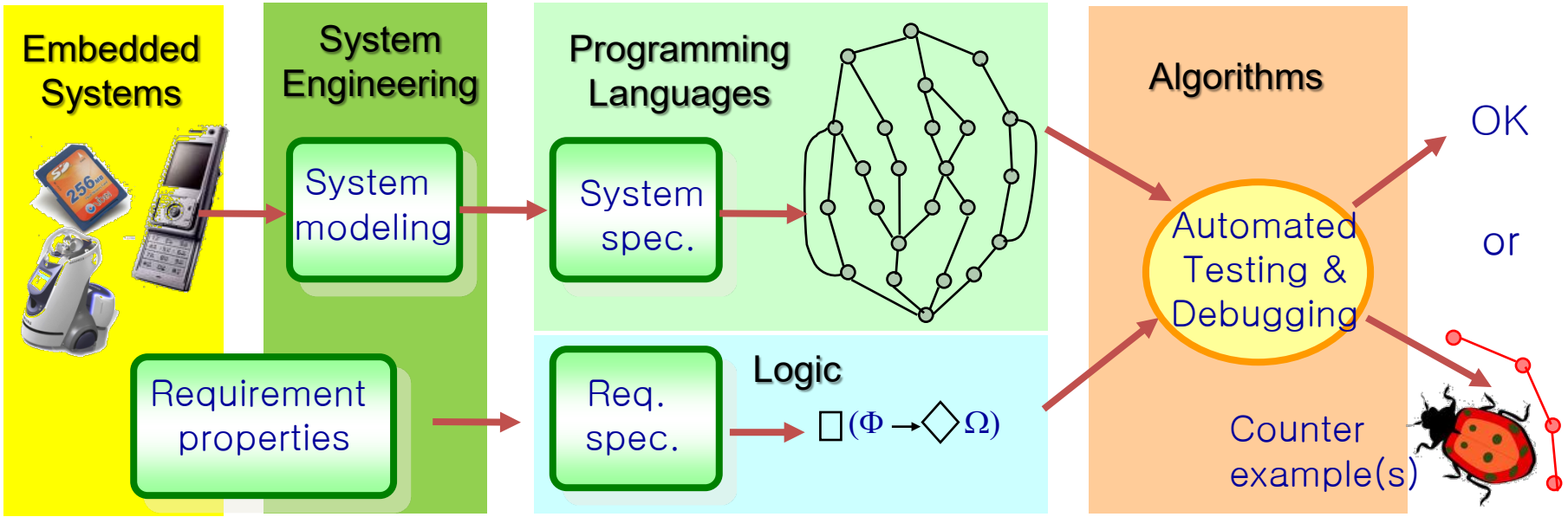
- A practical end-to-end formal framework for software development



Related Disciplines to SW Development

- Undergraduate CS classes contributing to SW development



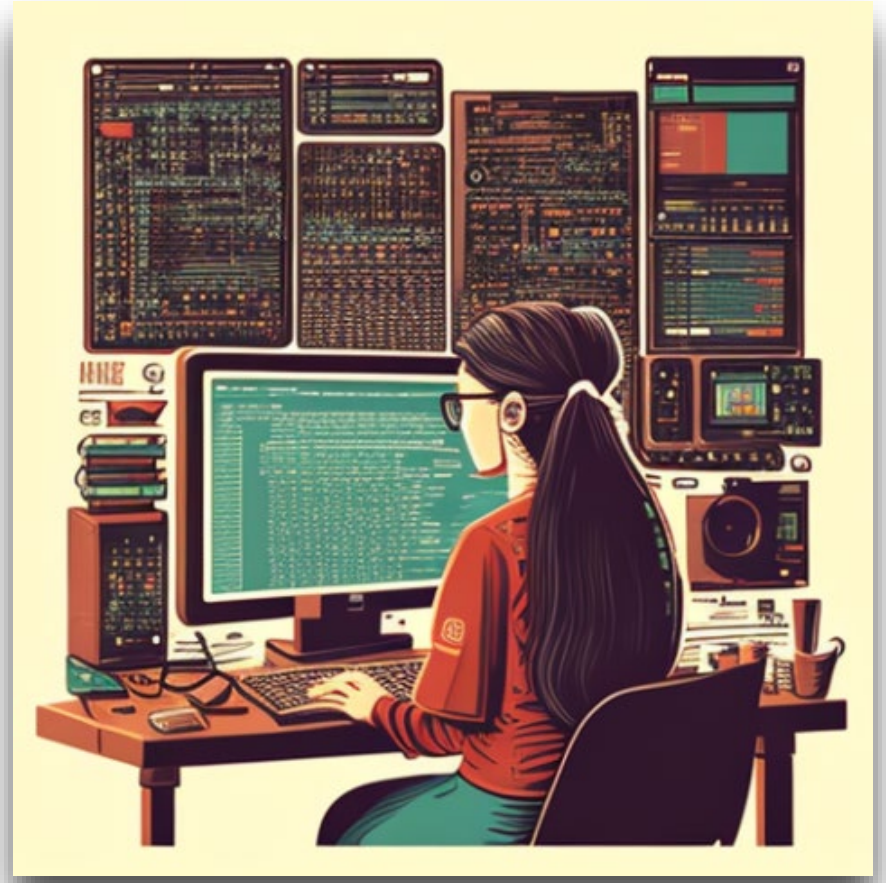


Hardware v.s. Software

- Flexibility leads to low development cost
  - Minimal costs for HW board manufacturing > 20K$
  - Minimal costs for SW development = 0$
- Growing popularity leads to large and complex software systems
  - Intel Meteor Lake CPU : 25 billion transistors
  - Windows 11: 70 million LoC (= multi-billions machine instructions)
- **Much harder** to validate/verify (V&V) software
  - HW design exploits symmetry, structure, and components
    - Formal design and V&V tools (e.g. Verilog, VHDL, etc) are popular
      - Standard property spec. language: OVL, PSL, SVA, etc
  - SW design allows maximal degree of freedom in programs
    - Formal principles and techniques have been rarely applied to SW due to
      - Failure to manage (entire) SW complexity
      - Lack of commercial tools and supports
      - (relatively) High learning curve
    - Products of all engineering fields provide warranty *except SW*
      - Ex> Intel CPU provides 3 years warranty
      - Ex> Microsoft Windows® provides no warranty - *Use it at your own risk!!!*

Software Engineering?

- “The application of a systematic, disciplined, quantifiable approach to development, operation and maintenance of software” (IEEE Systems and Software Engineering Vocabulary, 2010)
- “a systematic engineering approach to software development” (Wikipedia, https://en.wikipedia.org/wiki/Software_engineering)



From Prof. Shin Yoo’s CS350 lectures

Software Crisis

- As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.
  — Edsger Dijkstra, The Humble Programmer, Communications of the ACM, 1972

From Prof. Shin Yoo's CS350 lectures

NATO Conference 1968

(Dijkstra was also there)

- *The phrase 'software engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering.*
  - Editors of the proceedings of Software Engineering (1968)

From Prof. Shin Yoo's CS350 lectures

NATO Conference 1968

(Dijkstra was also there)

- The conference proceedings is a fascinating read: you can actually read the whole thing in a nicely typeset PDF: https://www.scrummanager.com/files/nato1968e.pdf
- There is also a very helpful blog post by Logan Mortimer, a software engineer in Melbourne: https://isthisit.nz/posts/2022/1968-nato-software-engineering-conference/
- Many interesting ideas and questions. Are software to be “manufactured” like other engineering products? How do we measure progress? How do we scale up to hundreds and thousands of people building the same system together?

How is SW like other engineering product?

How is it different from other engineering products?

From Prof. Shin Yoo's CS350 lectures

Essential Properties of SW

as pointed out by Brooks Jr. in 1987

- Complexity
  - More complex than any other human constructs for their size (no two parts are identical, because we would factor them out)
  - Scaling up does not mean making the same thing larger
- Conformity
  - Physicists firmly believe that there is a unified theory of things
  - Most of complexity in SW is arbitrary - SW has to conform to countless many things (institutions, systems, users, regulations…)

From Prof. Shin Yoo's CS350 lectures

Essential Properties of SW

as pointed out by Brooks Jr. in 1987

- Changeability
  - Compared to other engineering products, SW is more frequently pressured to change
  - If a software system is useful, people will try new edge cases at the borderline of the original domain
  - A successful software can outlive the underlying hardware, and therefore has to adapt
- Invisibility
  - SW cannot be easily embedded in space and therefore has no useful geometric representation
  - Without visual representation, communication and design becomes much more difficult

From Prof. Shin Yoo's CS350 lectures

“No Silver Bullet”

- “Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. (…) (software) is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products.” - Brooks Jr.

From Prof. Shin Yoo’s CS350 lectures

“No Silver Bullet”

- Brooks Jr. argues that there is no silver bullet that will kill the SW werewolf: the difficulties are inherent in SW, and no single technique will solve all of these.
- In the same titled essay, Brooks Jr. examines some candidates for the silver bullet (back in 1987)

From Prof. Shin Yoo’s CS350 lectures

Summary

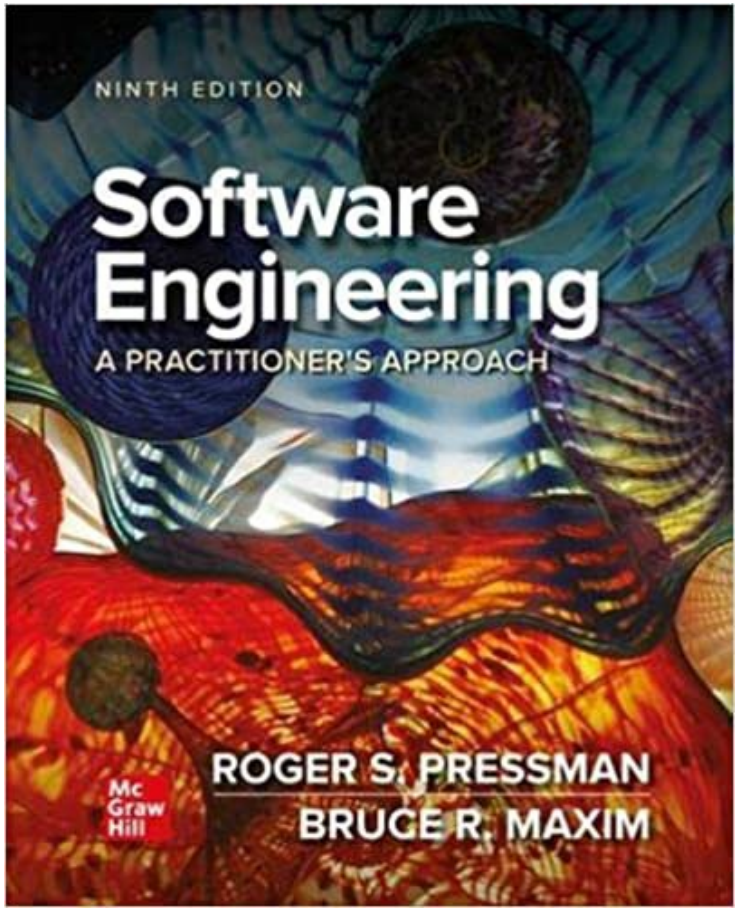
- SW is the most complex engineering artifact
- Building SW well requires more than excellent programming skills (although it helps).
- We will think about known problems, and get ourselves familiarized with widely accepted responses to those problems through lectures and **team projects**

Textbook

[PrMa20] Roger S. Pressman, and Bruce R. Maxim,

Software Engineering – A Practitioner's Approach,

9th Ed., Mc Graw Hill, 2020 (ISBN-13: 978-1259872976, Available Online)



Course Logistics (1/2)

- Pre-requisite:
  - CS.20300 System Programming, CS.20200 Programming Principles
  - I do not recommend students in non-SoC majors to take this class for its heavy class workload (HW, team project, **a large amount of documentation**, etc.).
  - Also, freshmen students (whose student IDs are 2025xxxx) are not eligible to take this 3rd year class
    - team projects requires technical communication skills

Course Logistics (2/2)

- Team projects:

  You should make a team consisting of 4 students by yourselves and report your team members by Oct 1st.

  - No team change allowed w/o legitimate reasons after Oct 1st (e.g., all teammates except you apply leave of absence, etc.)

- All teams will work on the same project topic (TBD) to learn and discuss with other teams
- You should present your team's project progress regularly at the class to promote lively discussion and feedback
- At the end of semester, you will evaluate your teammates and that evaluation contributes to grades of your teammates (e.g., to respect those who work hard and to reduce "free-rider")