Software Engineering

LECTURE 1: Introduction to SE

Ting Su
East China Normal University

自我介绍

- 苏亭(教授/博导)，软件科学与技术系，软件工程学院
- 个人主页

  http://tingsu.github.io (英文)

  https://faculty.ecnu.edu.cn/_s43/st2/main.psp (中文)
- 研究方向

  - 软件系统质量和安全保障

  - 软件分析与验证、软件和系统安全
- 办公室和邮件

  理科楼B1103，tsu@sei.ecnu.edu.cn

  研究生实验室(理科楼**B416)**

自我介绍

个人经历

- 2007-2016：华东师范大学 （本科、博士）
- 2014-2015：美国加州大学戴维斯分校（访问博士生）
- 2016-2019： 新加坡南洋理工大学（博士后研究员）
- 2019-2020:：瑞士苏黎世联邦理工（博士后研究员）
- 2020- ：华东师范大学（教授）

课程介绍

❑课程目标

1. 掌握和熟悉软件工程的理论、概念、方法、和工具
2. 具备分析、设计、开发和管理软件项目的能力
3. “软件工程实践” => “软件工程理论与实践”

❑课程形式

1.课程时间：每周五下午13:00-15:35

2.上课地点：理科楼B226

3.考核形式：出勤：5%；平时作业：45%； 课程项目：50%

❑参考教材

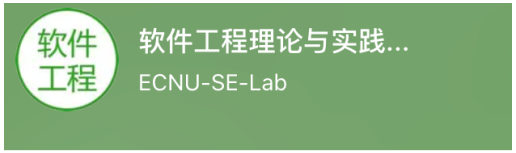
- 《Software Engineering textbook》, by I. Marsic. （电子书）
- 《Software Engineering-A Practitioner’s Approach (Eighth Edition)》, Roger S. Pressman著, 郑人杰等译. 北京: 机械工业出版社, 2015年.
- 软件测试(原书第二版), Patton,R.著, 张小松等译, 北京: 机械工业出版社, 2006.4.

课程介绍

- 课程网站(https://tingsu.github.io/files/courses/se2024.html)

- 课程助教

  陈浩仪（23级硕士研究生）



扫描群二维码，立刻加入该群

该二维码 7 天内 (9/23前)有效

- 课程交流形式

  1.课件资料：课程飞书群

  2.课程项目：GitHub (https://github.com/)

  3.平时交流：课程飞书群

  4.答疑时间：课前/课后（与我/助教交流）；飞书群（Any Time）；
  发邮件约时间/直接来办公室找我（每周三下午4:00-4:30）

What is Software?

What is Software?

- Software is:

(1) ***instructions*** (computer programs) that when executed provide desired features, function, and performance;

(2) ***data structures*** that enable the programs to adequately manipulate information and

(3) ***documentation*** that describes the operation and use of the programs.

What is Software?

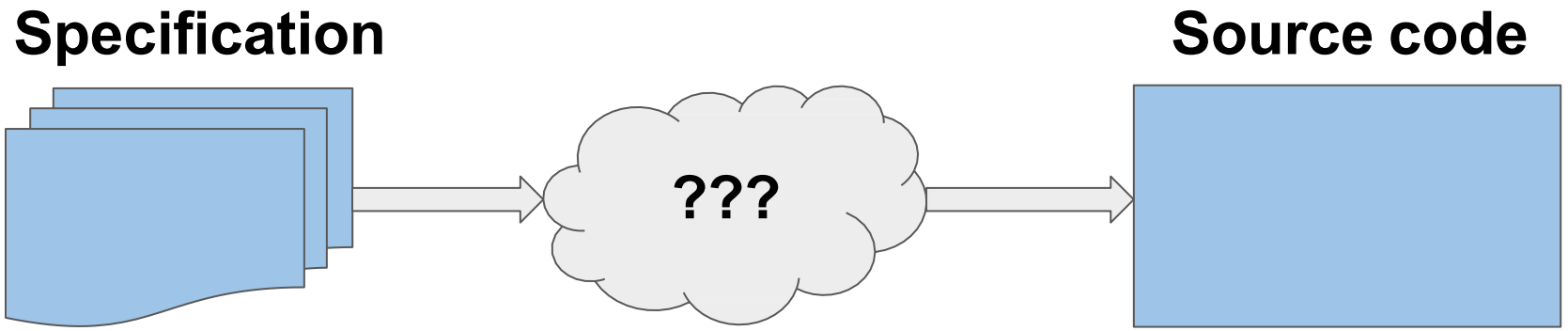
- Software is developed or engineered, it is not manufactured in the classical sense.
- Software doesn't "wear out."
- Although the industry is moving toward component-based construction, most software continues to be custom-built.

Software Applications

Software Applications

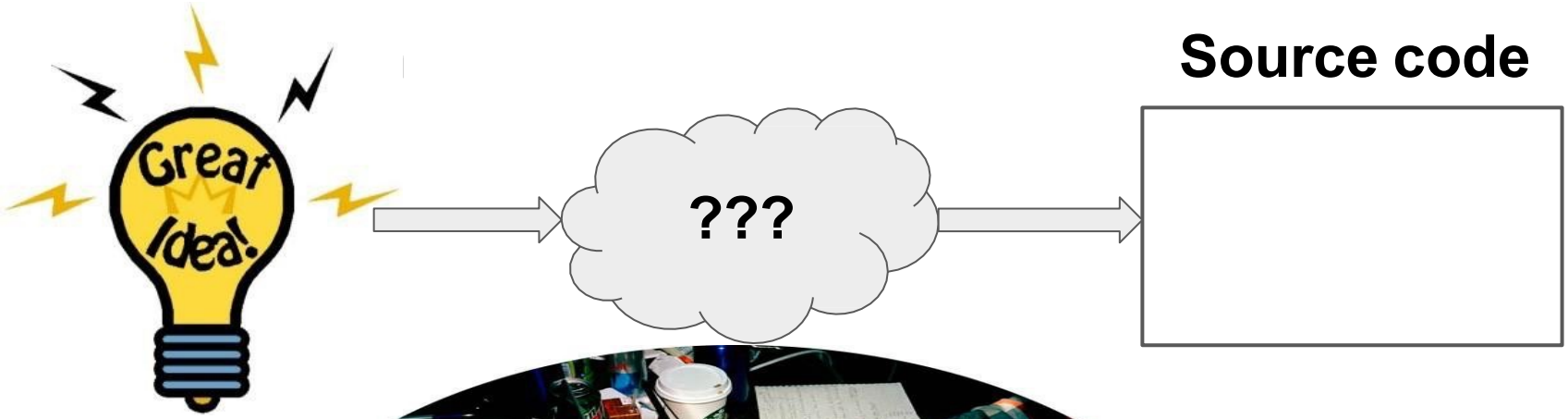
- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- WebApps / MobileApps
- AI software
- ……

Software development: the high-level problem



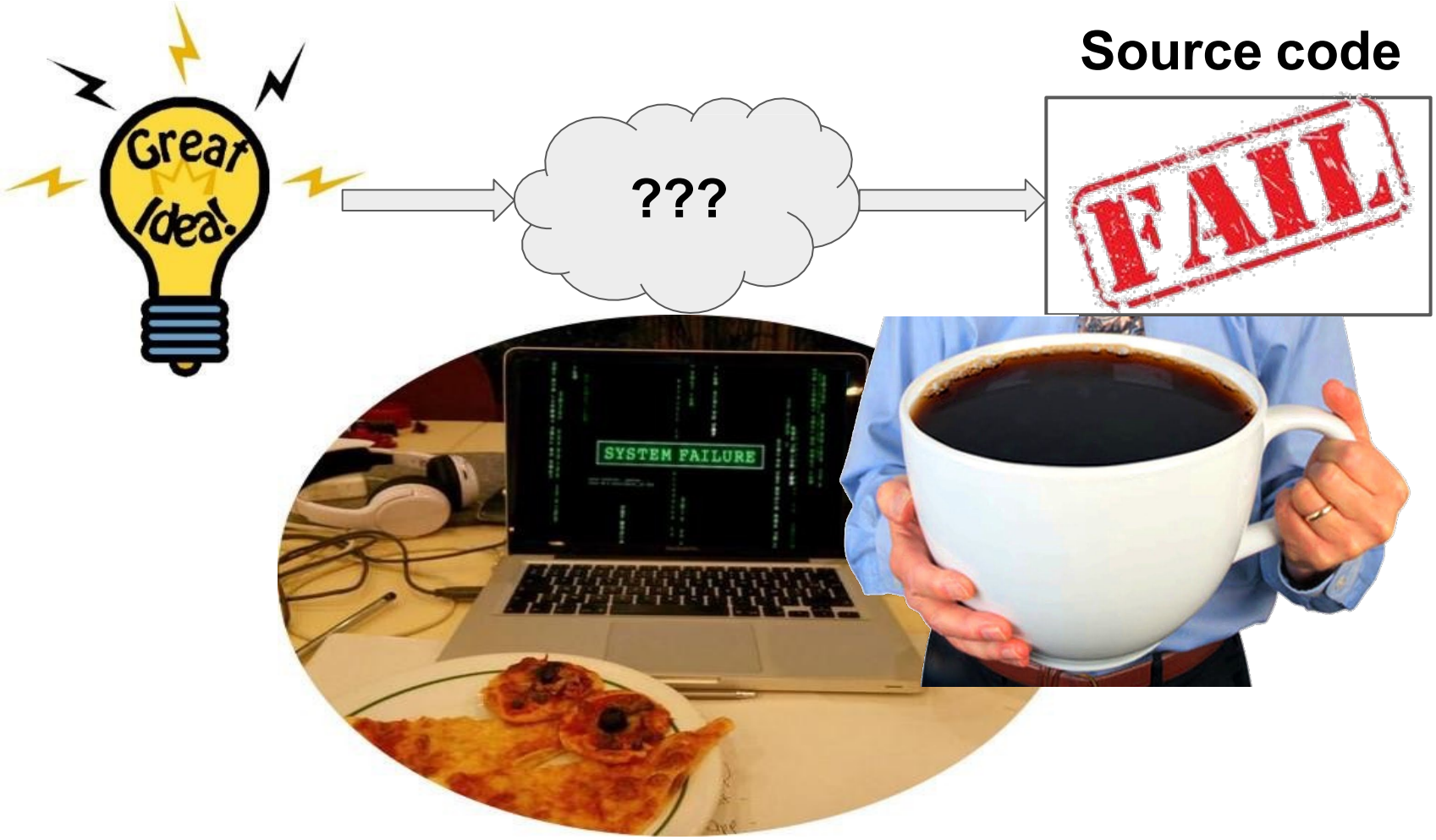
Software development: code and fix

One solution: "*Here happens a miracle*"



Software development: code and fix

One solution: "*Here happens a miracle*"



Software development: ad-hoc or systematic?

Pros

- No formal process.
- Easy, quick, and flexible.

Cons

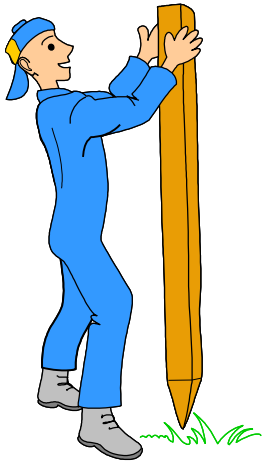
- Might lack important tasks such as design or testing.
- Doesn’t scale to multiple developers.
- How to measure effort and progress?

Software is Complex

- Complex $\neq$ complicated
- Complex = composed of many simple parts

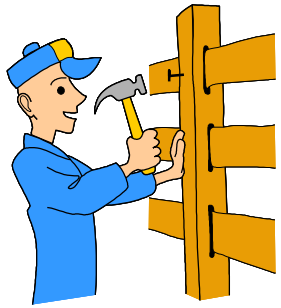
  related to one another
- Complicated = not well understood, or explained

Complexity Example: Scheduling Fence Construction Tasks

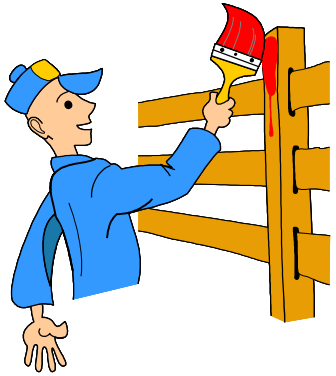


Setting posts
[3 time units]

Cutting wood
[2 time units]



Nailing
[2 time units for unpainted;
3 time units otherwise]



Painting
[5 time units for uncut wood;
4 time units otherwise]

Setting posts $<$ Nailing, Painting

Cutting $<$ Nailing

…shortest possible completion time = ?

[$\Rightarrow$ “simple” problem, but hard to solve without a pen and paper]

Legacy Software（遗留软件）

- Legacy software: ***Why must it change?***
  - software must be adapted to meet the needs of new computing environments or technology.
  - software must be enhanced to implement new business requirements.
  - software must be extended to make it interoperable with other more modern systems or databases.
  - software must be re-architected to make it viable within a network environment.

What is Software Engineering?

❑ More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming **Just one out of many important tasks!**
  - Software testing and debugging
  - Maintenance and refactoring

What is Software Engineering?

- Some realities:
  - a concerted effort should be made to understand the problem before a software solution is developed
  - design becomes a pivotal activity
  - software should exhibit high quality
  - software should be maintainable
- The seminal definition:
  - *[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.*

What is Software Engineering?

- The IEEE definition: Software Engineering:

(1) The application of a **systematic, disciplined, quantifiable approach** to the **development, operation, and maintenance** of software; that is, the application of engineering to software.

(2) The study of approaches as in (1).

This is what our SE researchers are doing!

History of Software Engineering

- Its origin: 1945 to 1965
  - Margaret H. Hamilton
- The software crisis: 1965 to 1985
  - Cost and Budget Overruns、 Property Damage、 Life and Death
  - The NATO Software Engineering Conferences (1968 Fall, 1969 Fall)

    (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/)

https://en.wikipedia.org/wiki/History_of_software_engineering

History of Software Engineering

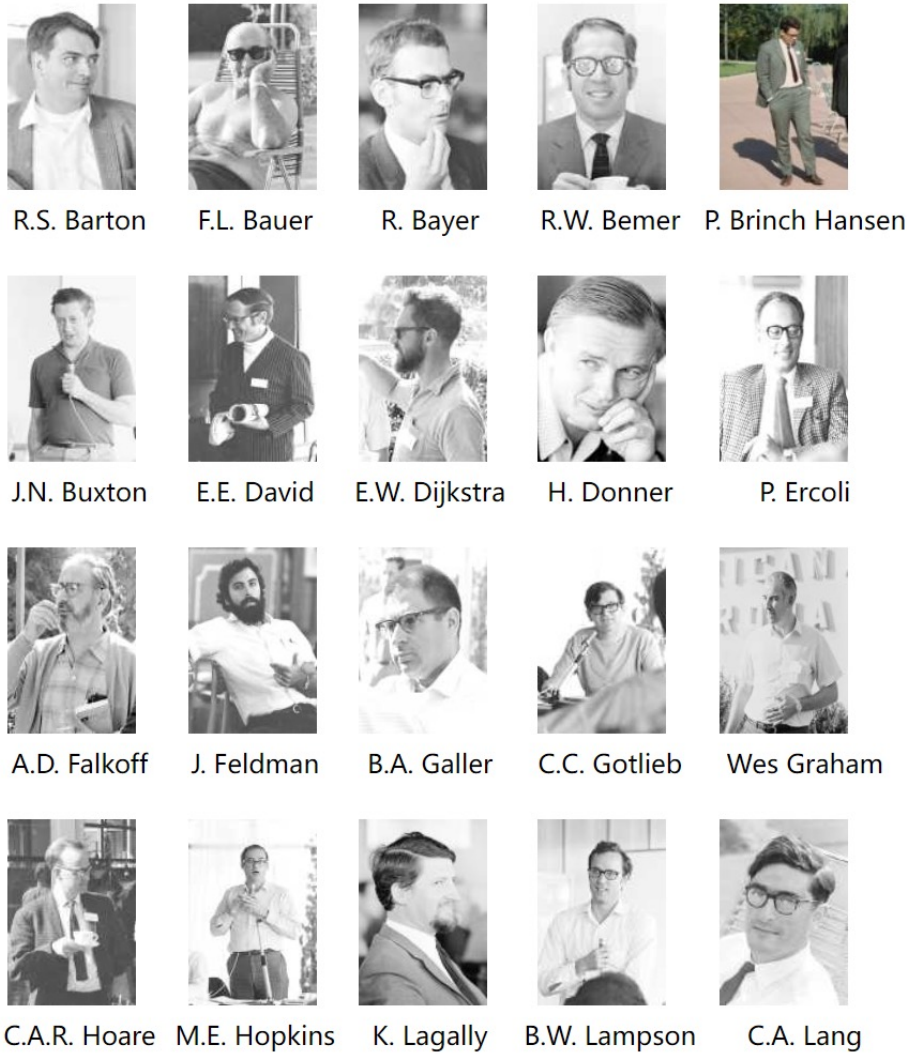
https://en.wikipedia.org/wiki/History_of_software_engineering

Hi ing

NATO Software Engineering Techniques Conference

Rome, Italy, 27-31 Oct 1969

Photographs provided by Robert McClure and Brian Randell.



R.S. Barton F.L. Bauer R. Bayer R.W. Bemer P. Brinch Hansen

J.N. Buxton E.E. David E.W. Dijkstra H. Donner P. Ercoli

A.D. Falkoff J. Feldman B.A. Galler C.C. Gotlieb Wes Graham

C.A.R. Hoare M.E. Hopkins K. Lagally B.W. Lampson C.A. Lang

https://en.wikipedia.org/wiki/History_of_software_engineering

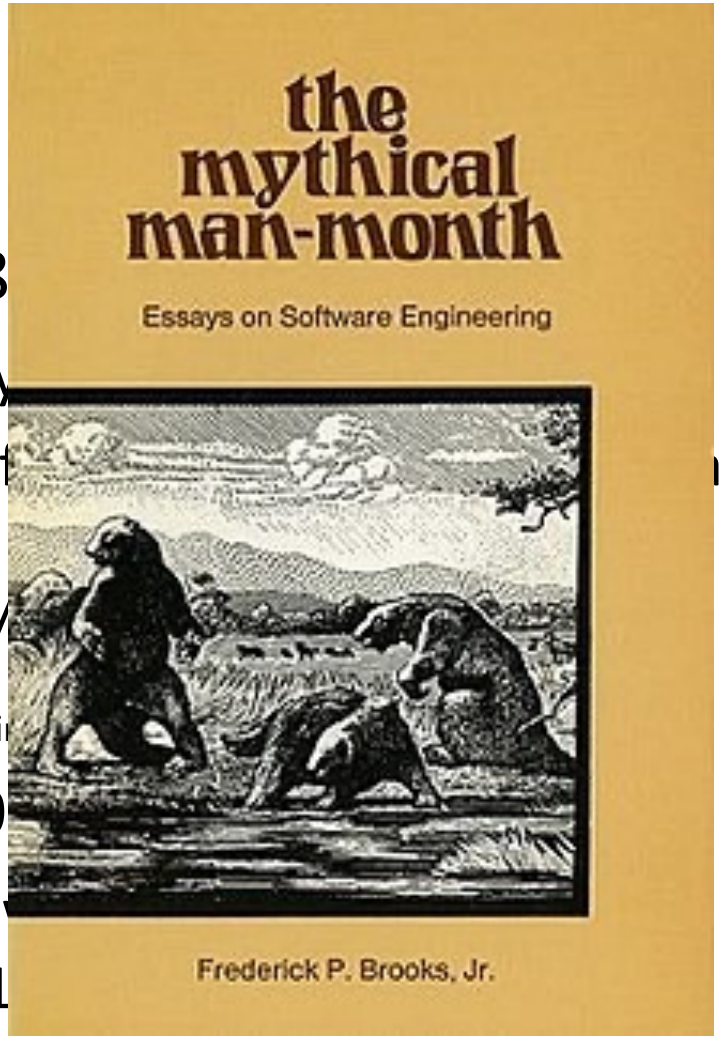
History of Software Engineering

- Its origin: 1945 to 1965
  - Margaret H. Hamilton
- The software crisis: 1965 to 1985
  - Cost and Budget Overruns、Property Damage、Life and Death
  - The NATO Software Engineering Conferences (1968 Fall, 1969 Fall)
    (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/)
  - Fred Brooks's 《The Mythical Man-Month》(1975、1995)
    (https://en.wikipedia.org/wiki/The_Mythical_Man-Month)
    "For landmark contributions to computer architecture, operating systems, and software engineering"
- "No Silver Bullet": 1985 to 1989
  - No individual technology or practice would ever make a 10-fold improvement in productivity within 10 years --- Fred Brooks

https://en.wikipedia.org/wiki/History_of_software_engineering

History of Software Engineering

- Its origin: 1945 to 1965
  - Margaret H. Hamilton
- The software crisis: 1965 to 198
  - Cost and Budget Overruns、 Property
  - The NATO Software Engineering Cont ll)
    (http://homepages.cs.ncl.ac.uk/brian.randell/NATO/)
  - Fred Brooks's 《The Mythical Man-M
    (https://en.wikipedia.org/wiki/The_Mythical_Man-Month)
    "For landmark contributions to computer architecture, operati
- "No Silver Bullet": 1985 to 1989
  - No individual technology or practice
    improvement in productivity within 1



https://en.wikipedia.org/wiki/History_of_software_engineering

History of Software Engineering

- Its origin: 1945 to 1965
- The software crisis: 1965 to 1985
- “No Silver Bullet”: 1985 to 1989
- Prominence of the Internet: 1990 to 1999
  - World Wide Web (WWW)、E-mail、Search Engines、Natural Language Translation、Computer Viruses
- Lightweight methodologies: 2000 to 2015
  - Extreme Programming (XP)
- 2015 – now
  - Low/no-code development, microservices, AI-enabled software,…

https://en.wikipedia.org/wiki/History_of_software_engineering

History of Software Engineering

50 years of Software engineering – 50 years of tremendously successful promotion of research, education and practices in software engineering.

History of Software Engineering

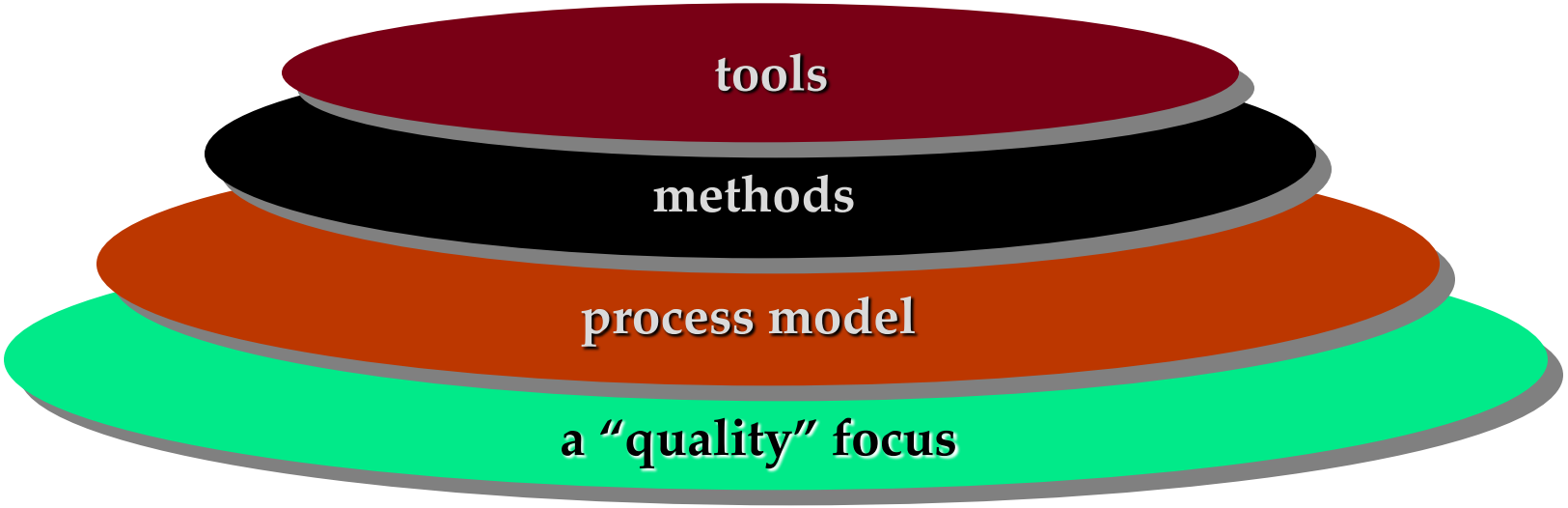
Margaret Heafield Hamilton is an American computer scientist, systems engineer, and business owner. She was director of the Software Engineering Division of the MIT Instrumentation Laboratory, which developed on-board flight software for NASA's Apollo program. She later

ONFERENCE ON
NG MAY 27 - JUNE 3 2018 GOTHENBURG, SWEDEN

* ICSE 2018 * (series) / Margaret Hamilton

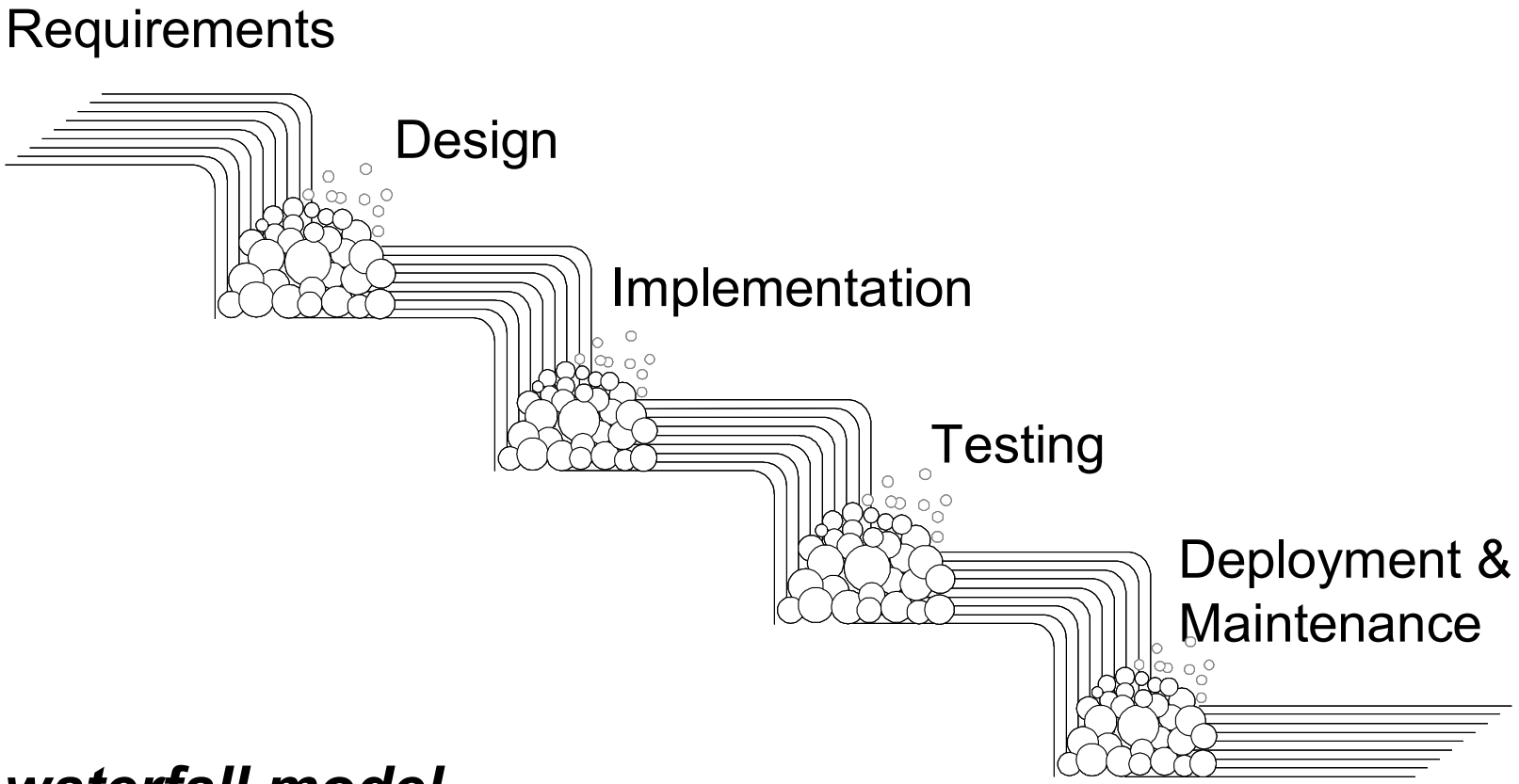
https://www.youtube.com/watch?v=ZbVOF0Uk5lU (ICSE 2018 – Keynotes - Margaret Hamilton)

A Layered Technology



Software Engineering

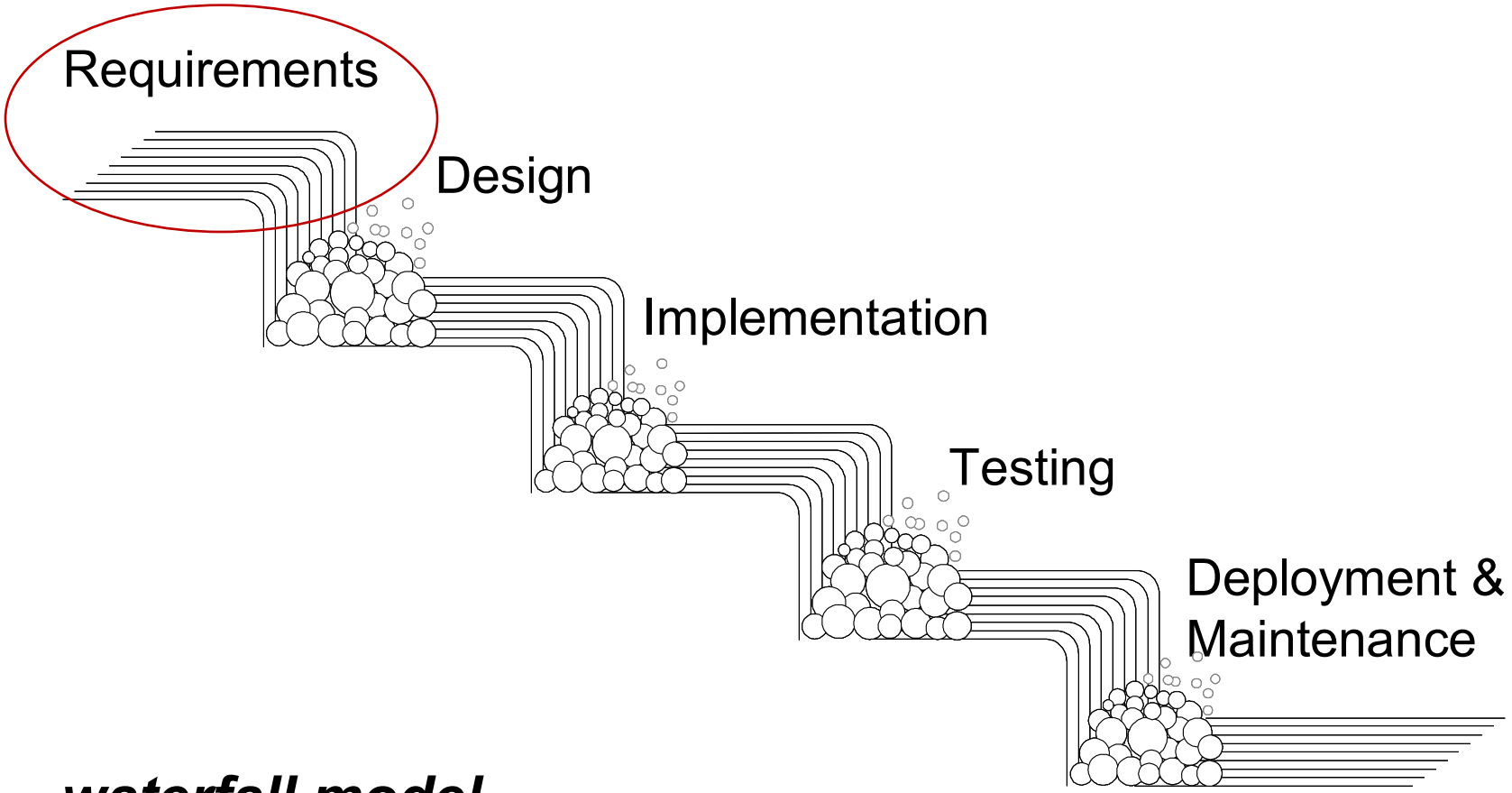
Process Model (SDLC)



waterfall model

https://en.wikipedia.org/wiki/Waterfall_model

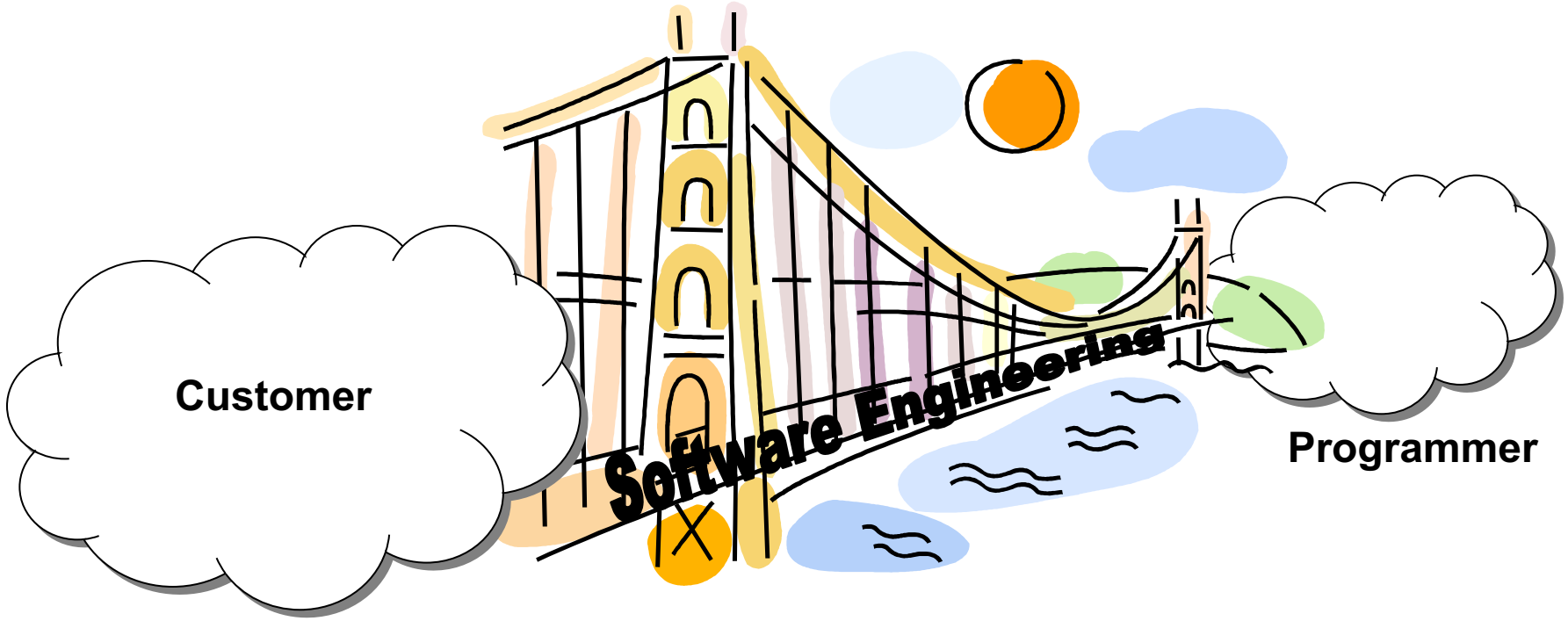
Process Model (SDLC)



waterfall model

The Role of Software Engineering

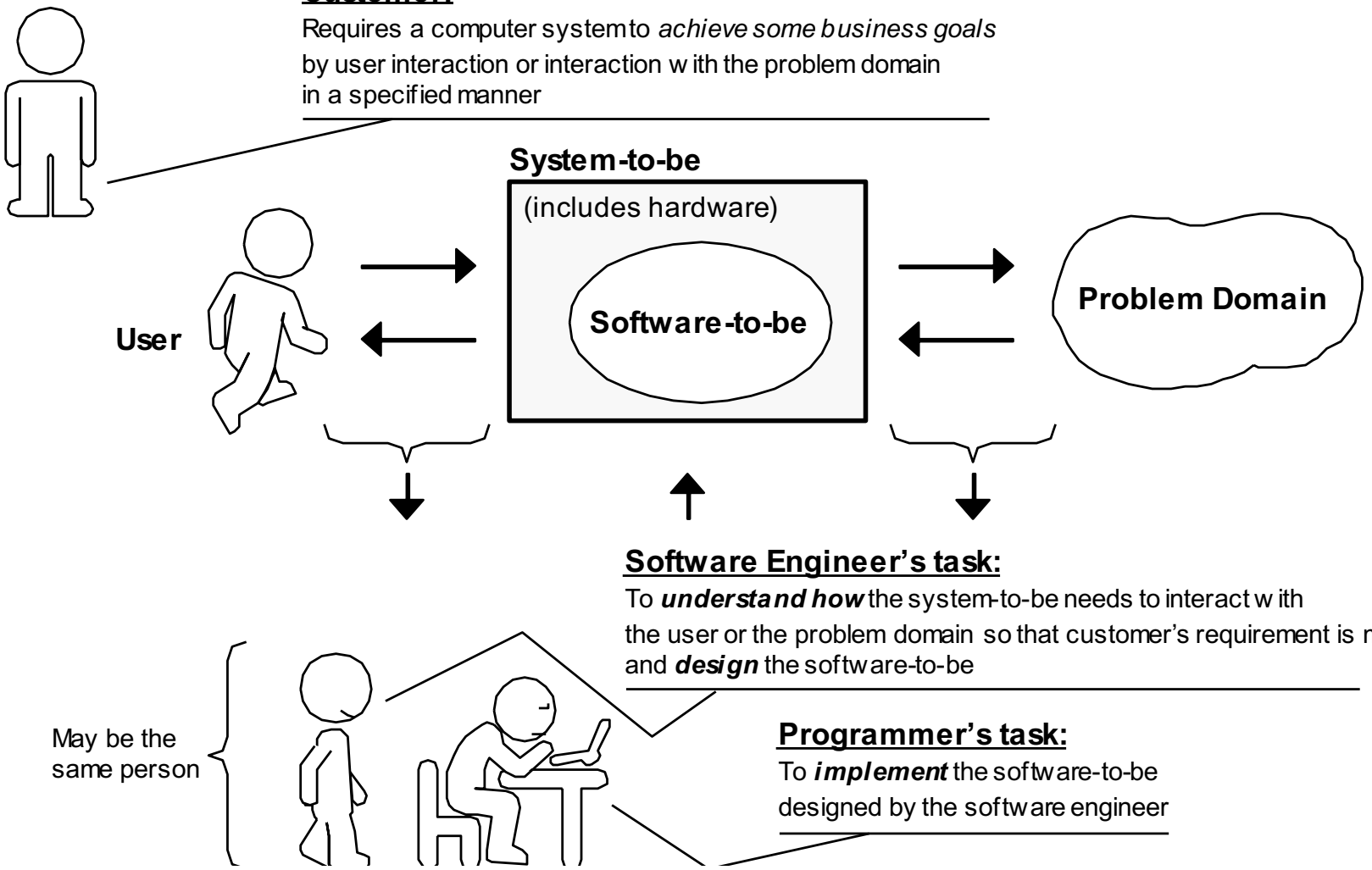
A bridge from customer needs to programming implementation



First law of software engineering

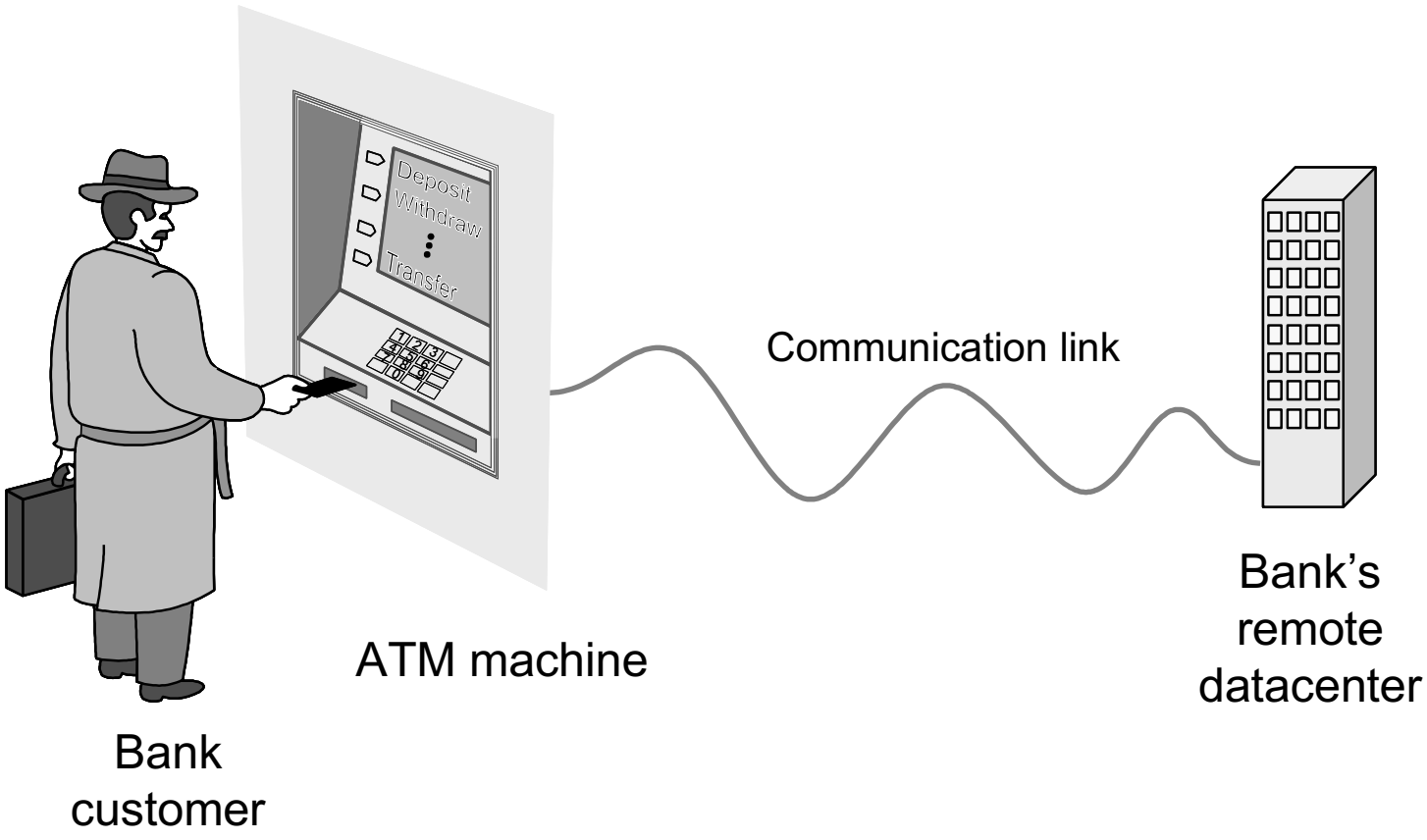
Software engineer is willing to learn the problem domain
(problem cannot be solved without understanding it first)

The Role of Software Engineering

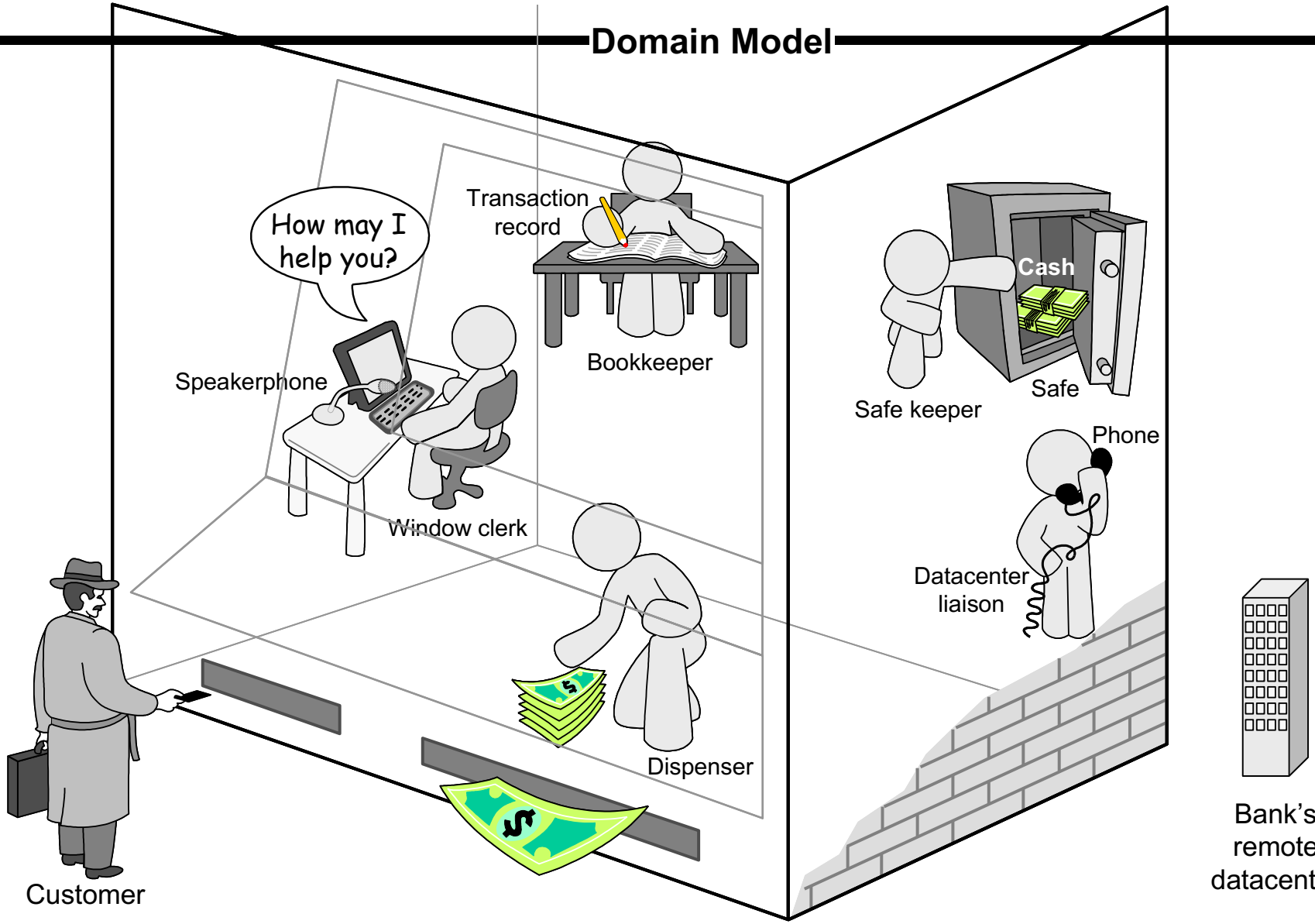


Example: ATM Machine

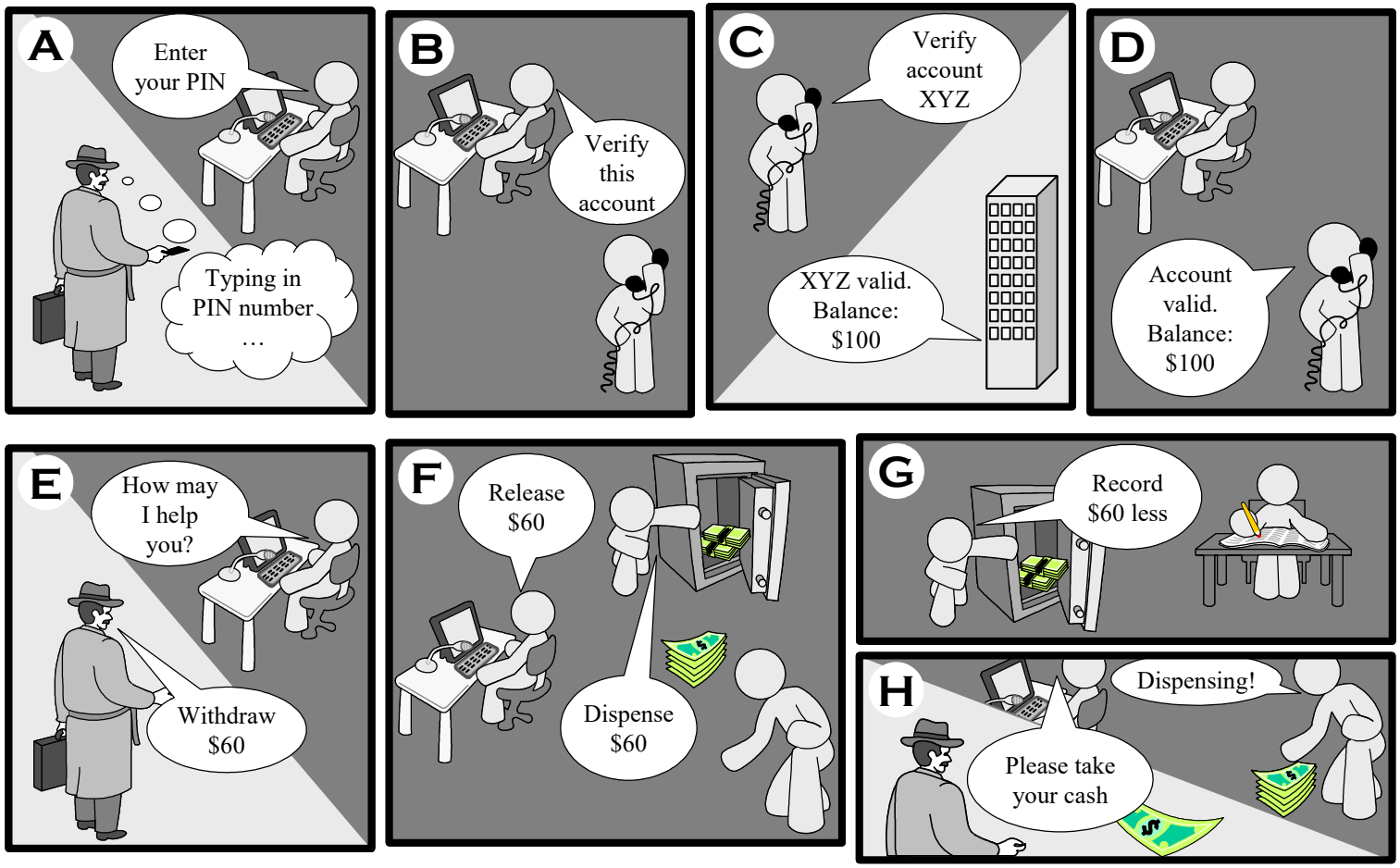
Understanding the money-machine problem:



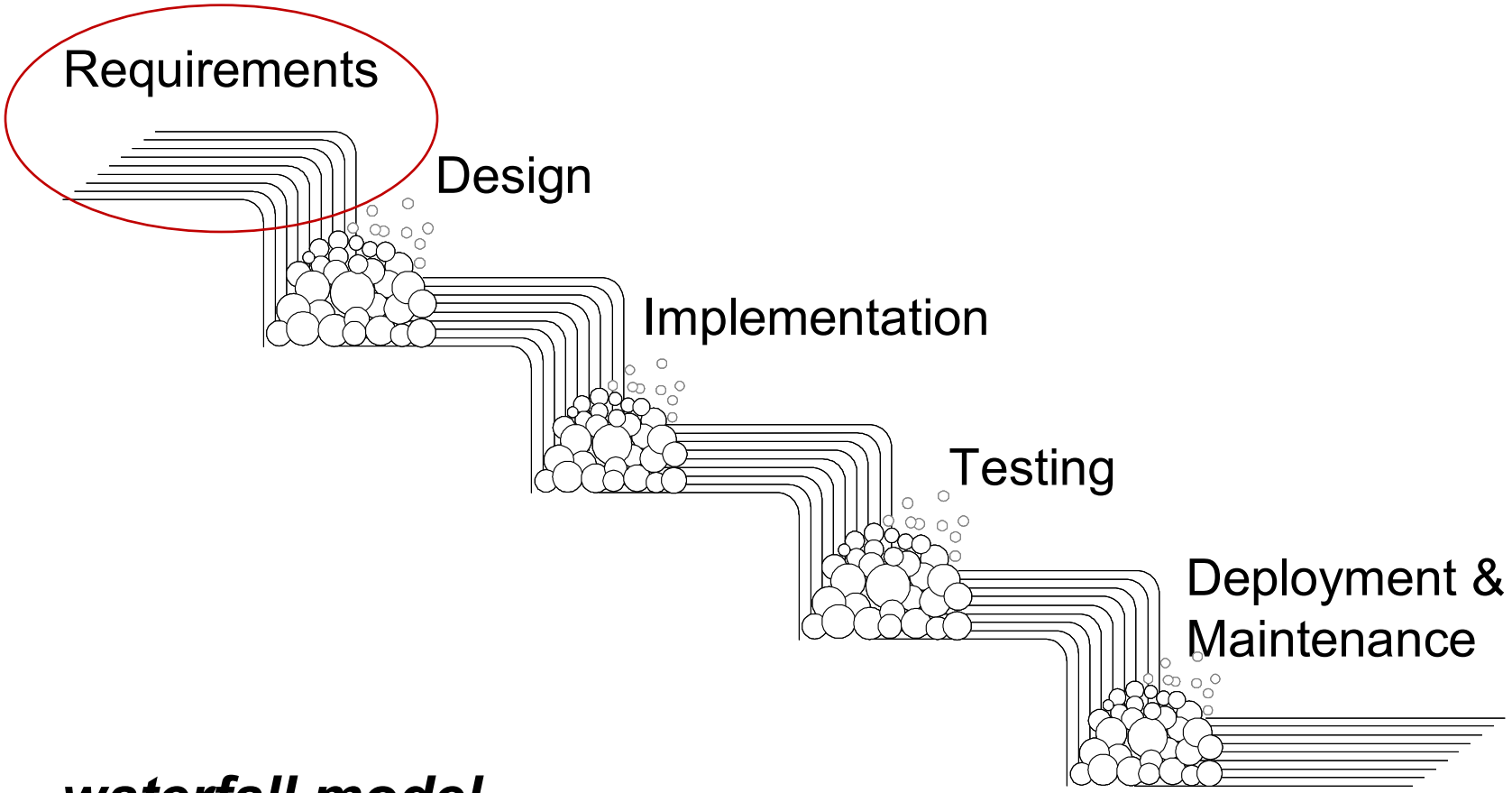
How ATM Machine Might Work



Cartoon Strip: How ATM Machine Works

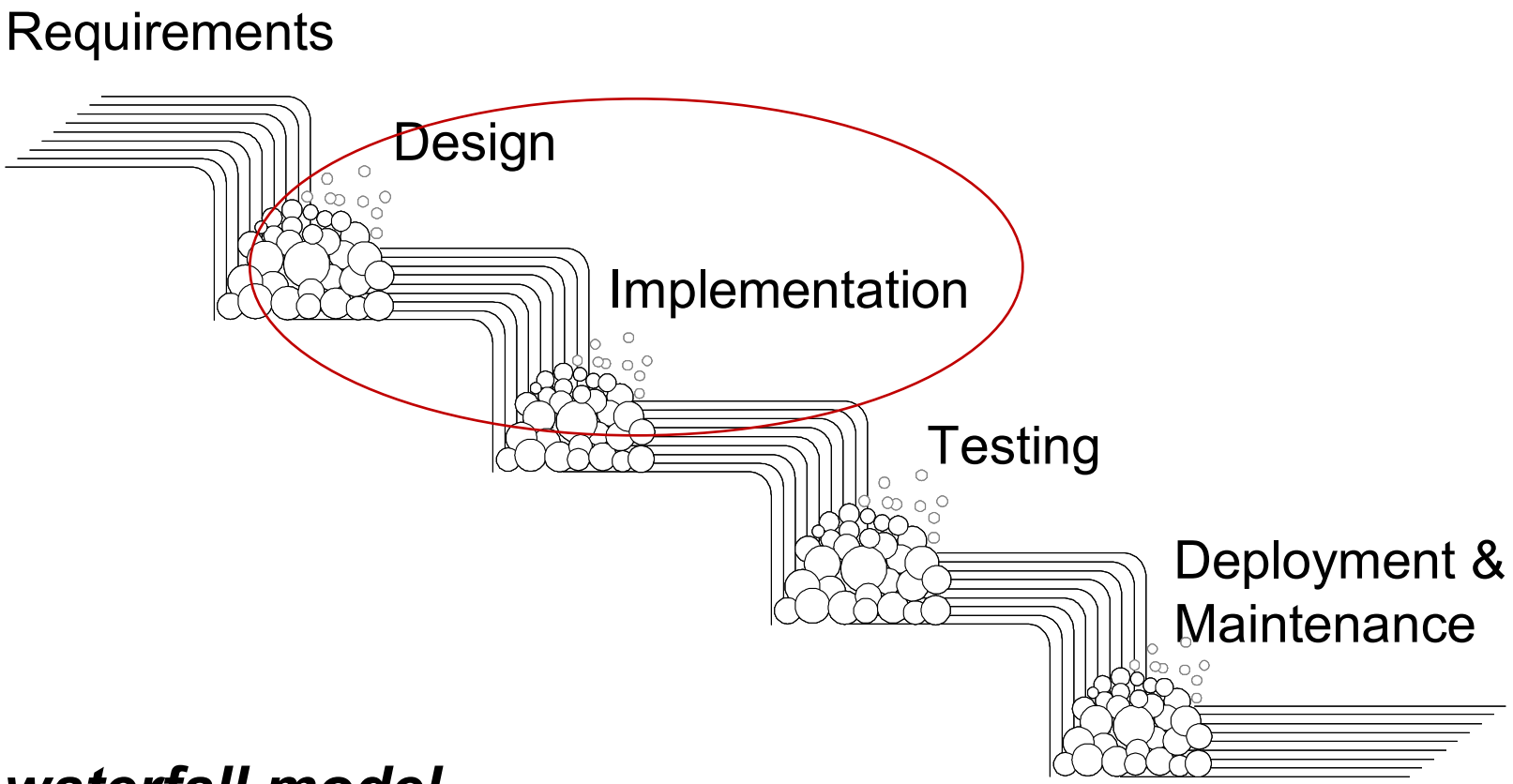


Process Model (SDLC)



waterfall model

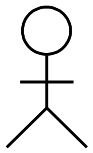
Process Model (SDLC)



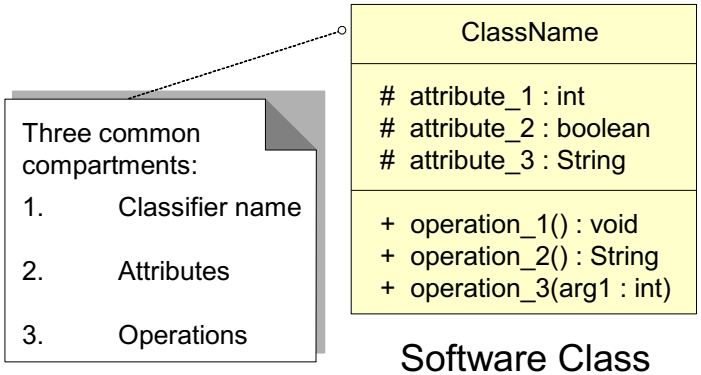
waterfall model

UML – Language of Symbols

UML = Unified Modeling Language

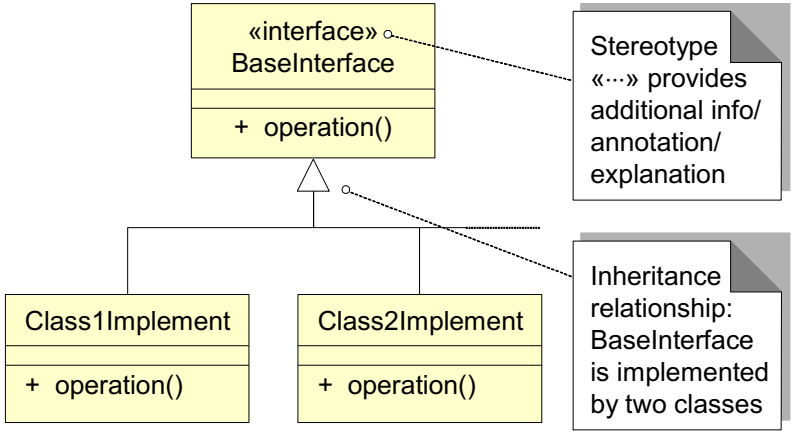


Actor

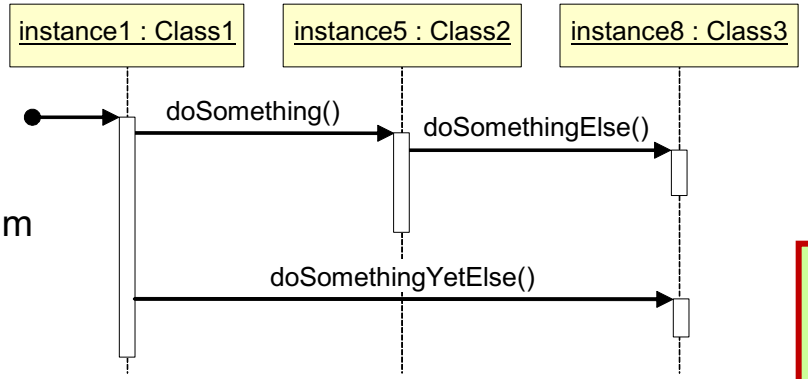


Comment

Software Class



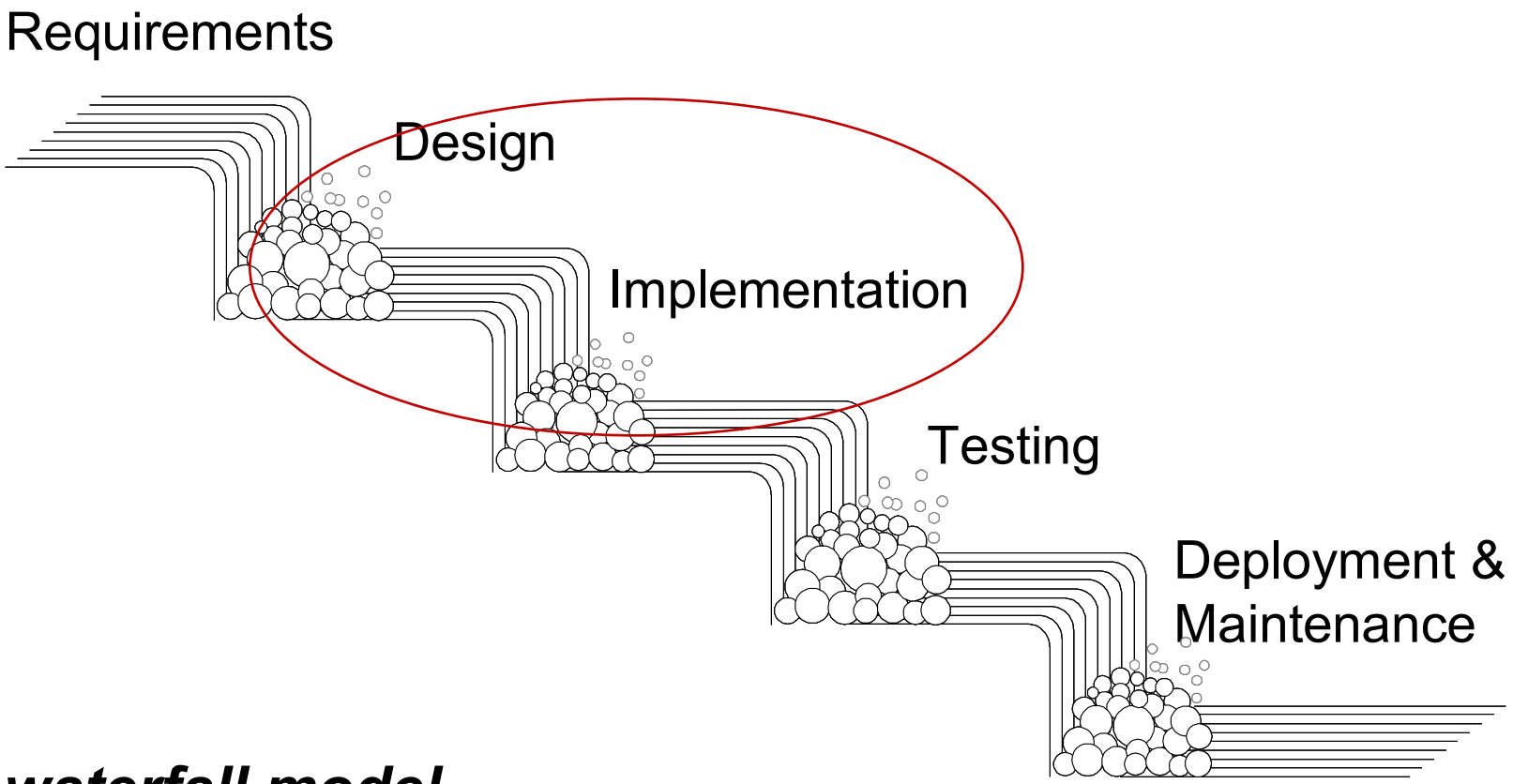
Software Interface Implementation



Interaction Diagram

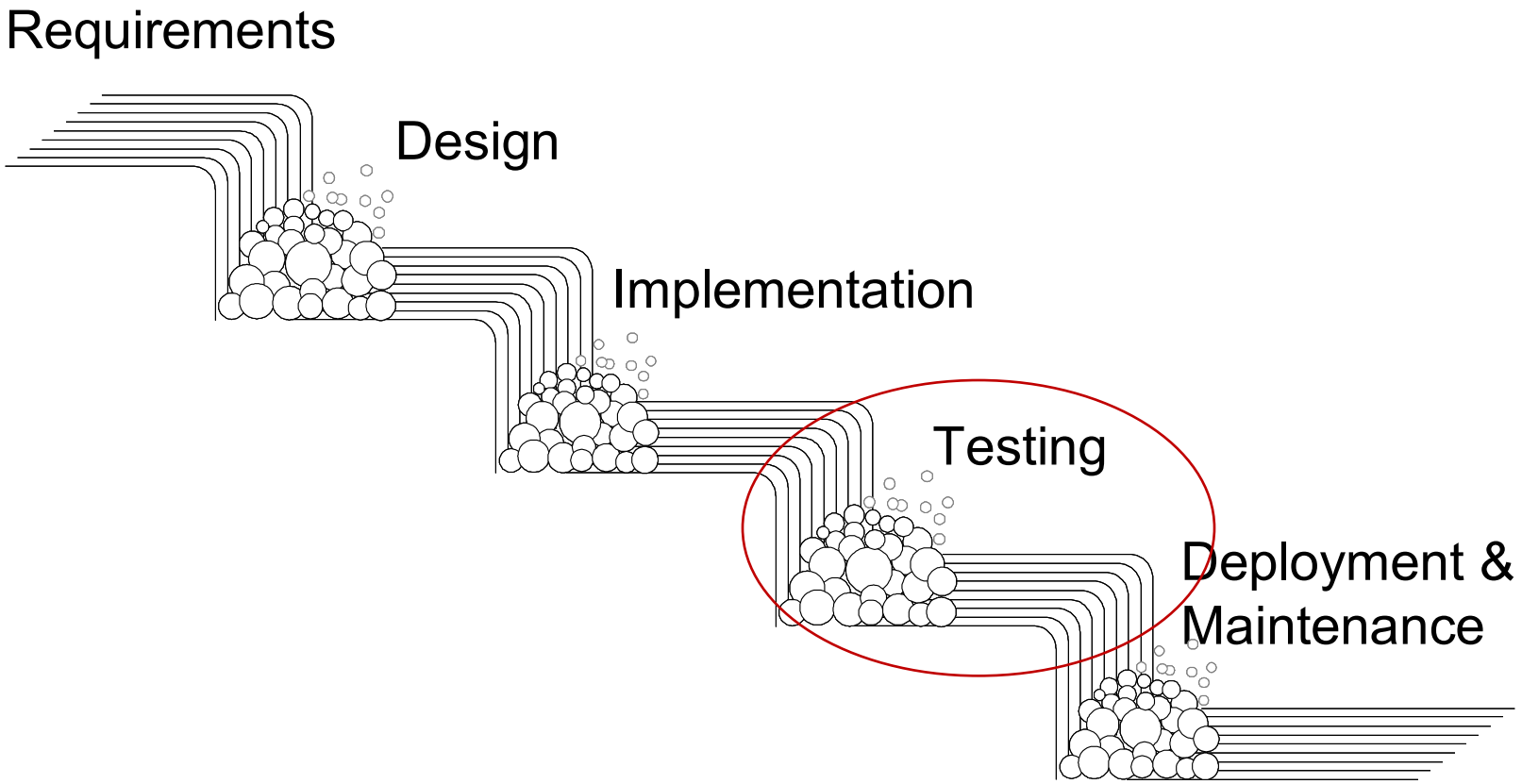
Online information:
http://www.uml.org

Process Model (SDLC)

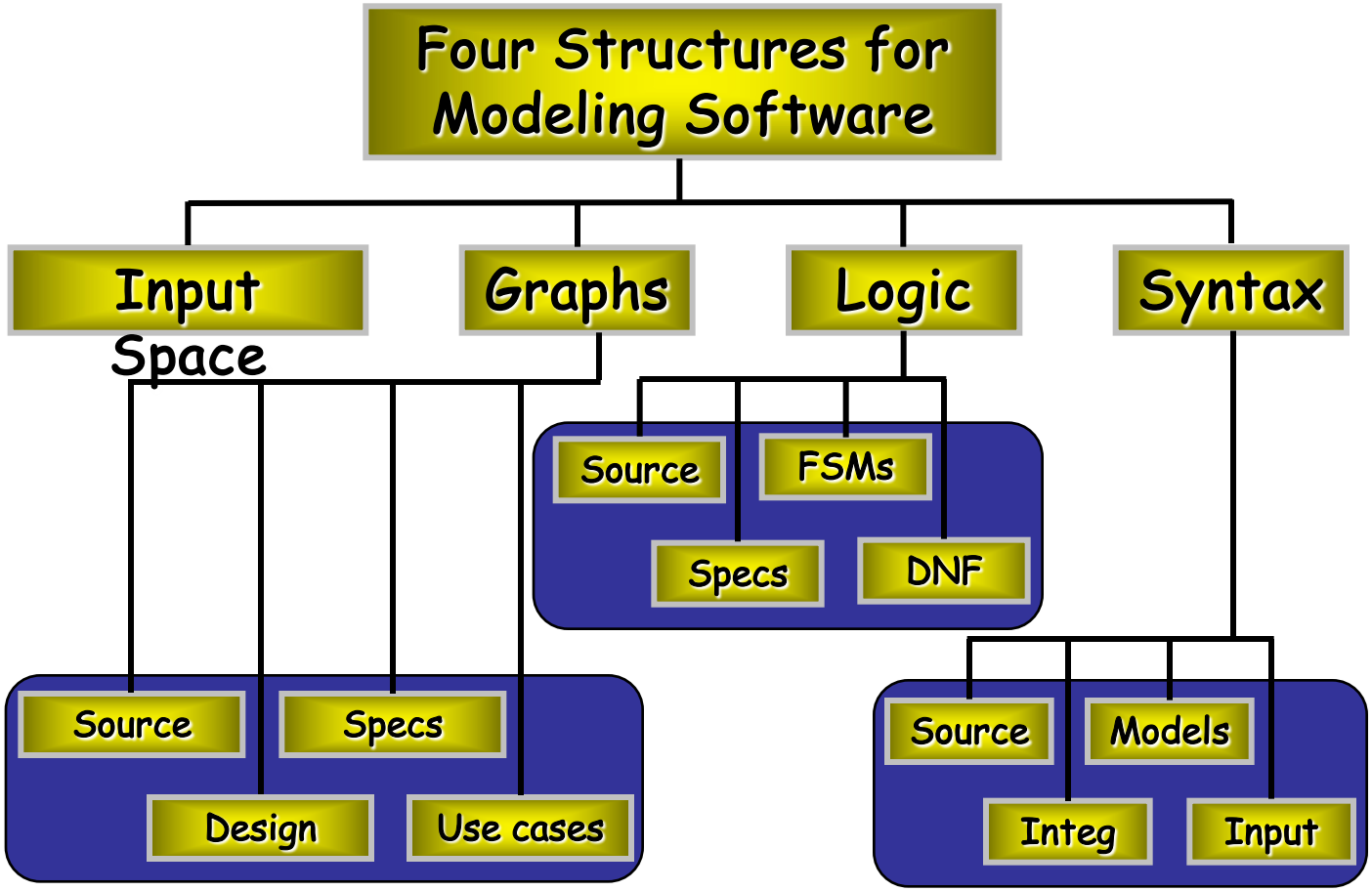


waterfall model

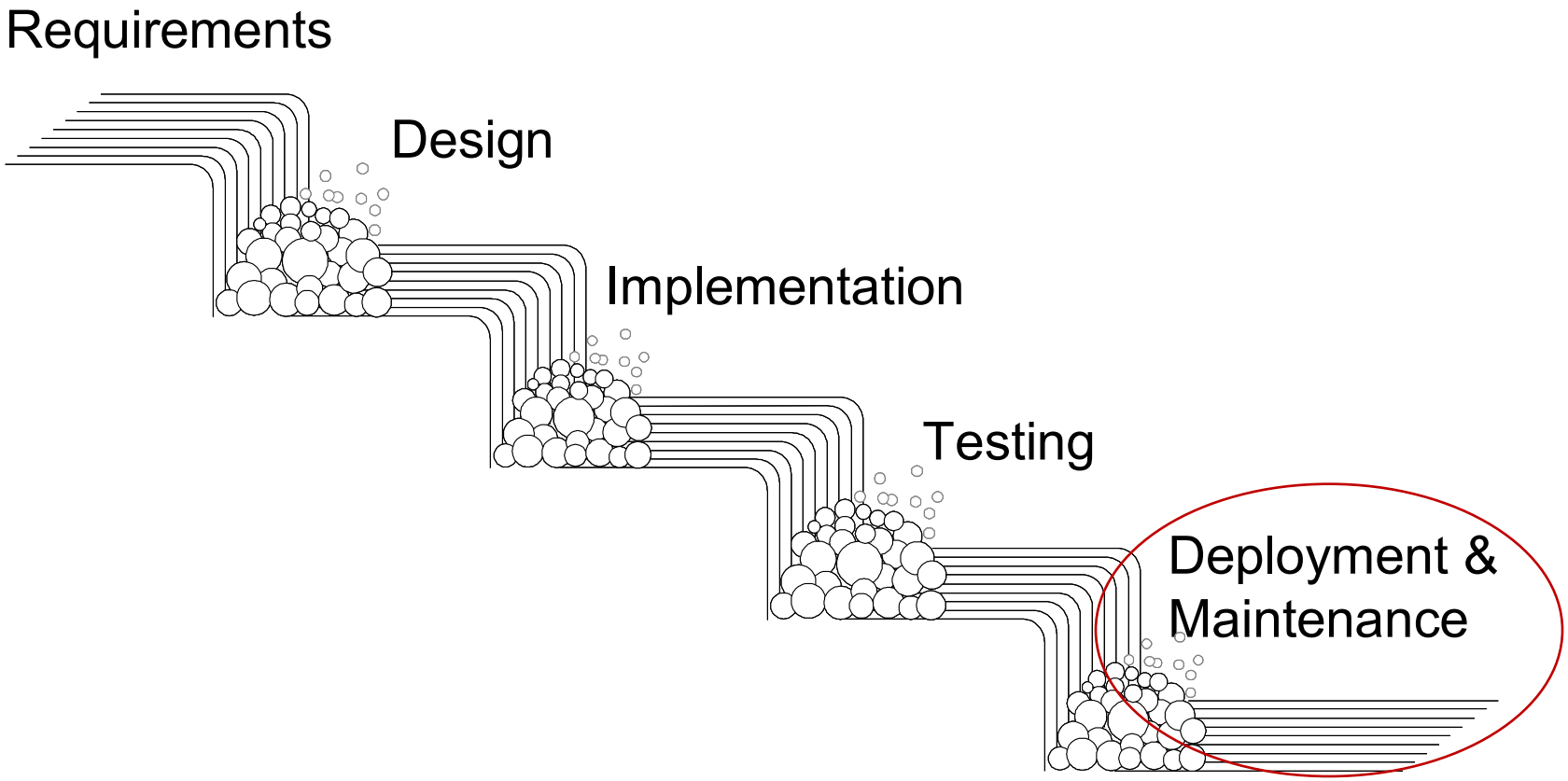
Process Model (SDLC)



Coverage-based Testing



Process Model (SDLC)



Course Overview

- **Software processes and requirements**
  - Different software development processes.
  - Problem understanding and requirement analysis
  - Precise documentation (requirements and specifications).
- **Software design and development**
  - Decompose a complex problem and build abstractions.
  - Learn and apply software design and architecture
  - Effectively use version control, build systems, and code review.
- **Software testing and debugging**
  - Know software testing theory
  - Hands-on experience on using testing and debugging techniques.
  - Continuous integration (CI).

Lectures (Tentative Schedule)

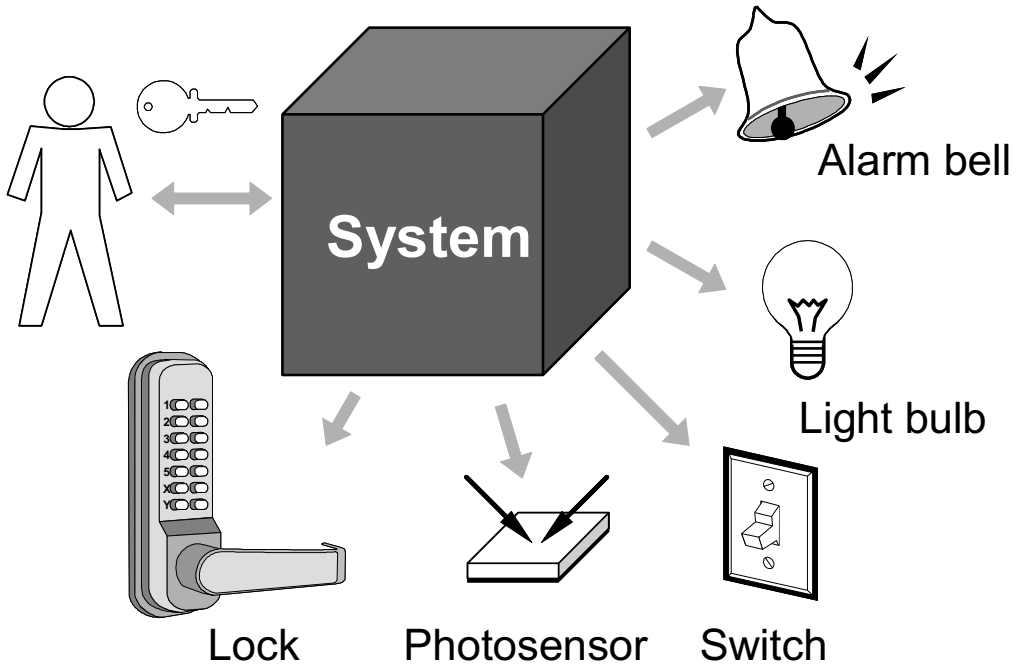
Date	Topic	Material
9 Sep.	Introduction: Software Engineering	Margaret Hamilton's talk at ICSE'18
20 Sep.	Software Development Lifecycle	The Joe Test
27 Sep.	Lab: Version Control and Git (Project II)	
4 Oct.	Introduction of Web/iOS/Android/小程序/HarmonyOS Apps Development	飞书群
	Lab: Build a demo app (preparing for Project II)	
7 Oct.	Requirment Analysis & Use cases	Writing Effective Use Cases
	Lab: Use case diagrams & schema (Project I)	
12 Oct.	Domain Analysis and Domain Model	Domain Model v.s. Data Model, General Guide For Exploring Large Open Source Codebases
	Lab: System sequence diargram, Activity diagram and Domain model (Project I)	
2 Nov.	Design Basics & Principles	Good Programming Principles
	Lab: Project Presentation --- New Features (Project II)	
9 Nov.	Design Patterns	Design patterns implemented in Java
	Lab: design sequence digram, class diagram and design patterns	
16 Nov.	Software Architecture	Client/Server, Central Repository, Layered, Model-View-Controller, Pipe-and-Filter, Microservices. Better Design: EasyChair v.s. HotCrap (Hot Crap!)
23 Nov.	Foundations on Software Testing	
	Lab: JUnit Testing	
30 Nov.	Coverage-based Software Testing	Code Coverage at Google, Does mutation testing improve testing practices?

Three Mini Projects

- Mini Project 1: Safe Home Access
- Mini Project 2: Build Your Own App
- Mini Project 3: Themis benchmark

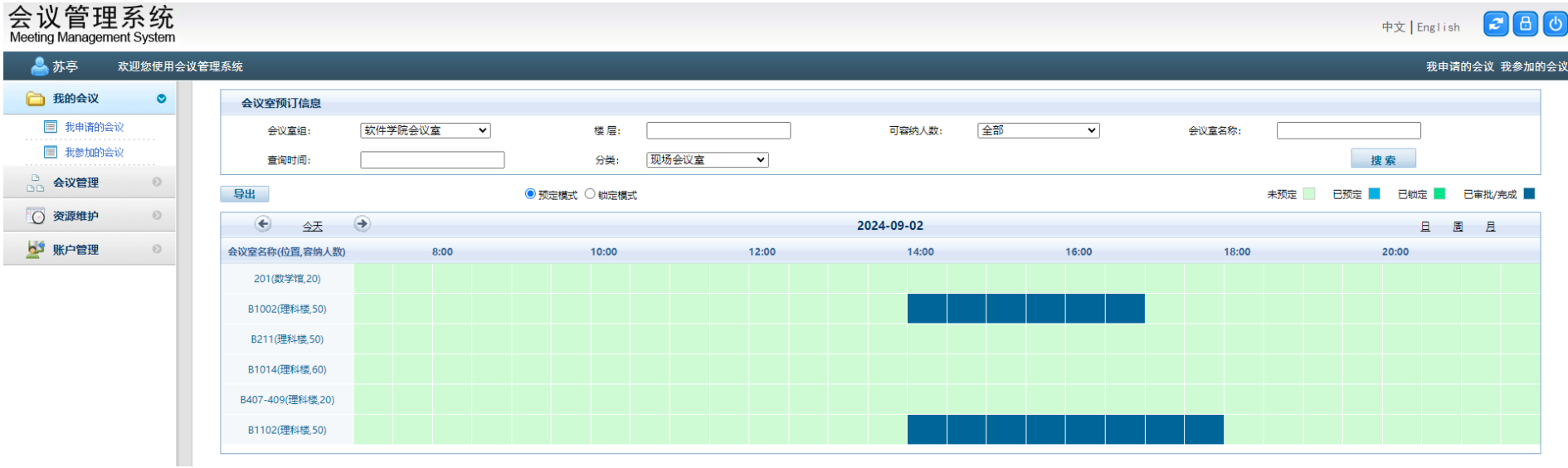
Project 1: Home Access Control

- Functionality: An electronic system for:
  - Home access control
    - Locks and lighting operation
  - Intrusion detection and warning



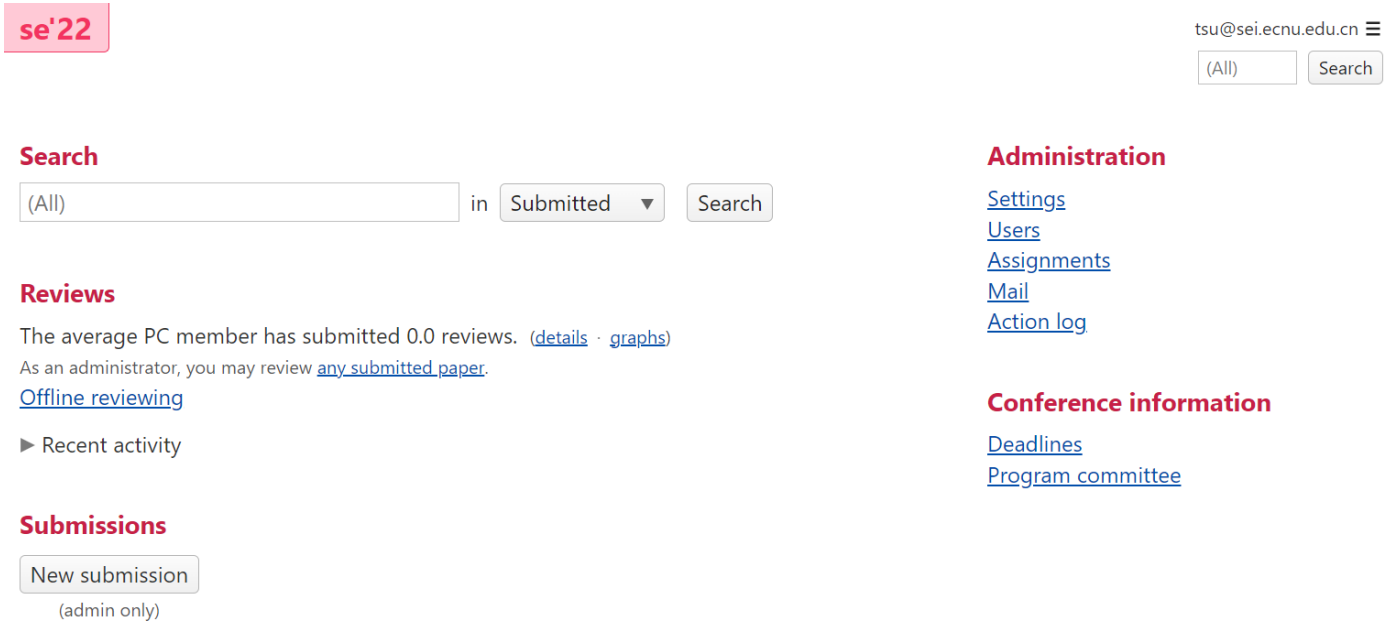
Project 2: Build Your Own App

- 开发你们自己的应用
  - 会议室预约管理App
  - 论文评审管理App
  - 形式：Web/iOS/Android/小程序/HarmonyOS Apps



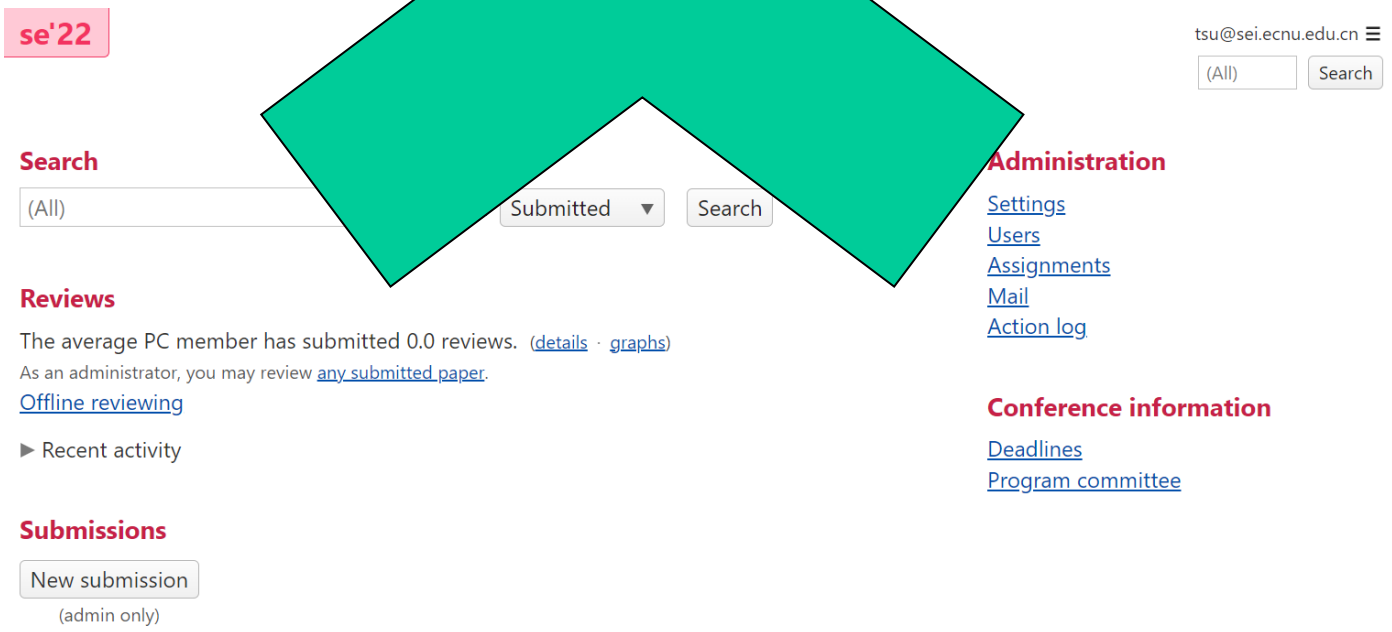
Project 2: Build Your Own App

- 开发你们自己的应用
  - 会议室预约管理App
  - 论文评审管理App
  - 形式：Web/iOS/Android/小程序/HarmonyOS Apps



Project 2: Build Your Own App

- 开发你们自己的应用
  - 会议室预约管
  - 论文评审管理
  - 形式：Web/iOS/A /HarmonyOS Apps



Mini Project 3: Themis Benchmark

- A dataset of software bugs for Android apps
  - GitHub: https://github.com/the-themis-benchmarks/home

Themis: python themis.py --avd *avd_name* -n *dev_cnt* --apk *apk_name* -o *output_dir* --time *testing_time* --repeat *run_cnt* --tool tool_name [--login *login_script*] [--gui] [--check_crash] [--coverage]

Tool Name	Venue / Source	Open-source?	Main technique
Monkey	-	✓	Random testing
Sapienz	ISSTA'16	✗	Search-based
Stoat	ESEC/FSE'17	✓	Model-based
DroidBot	ICSE'17	✓	Model-based
Ape	ICSE'19	✓	Model-based
Humanoid	ASE'19	✓	Deep learning-based
ComboDroid	ICSE'20	✓	Model-based
TimeMachine	ICSE'20	✓	State-based
Q-testing	ISSTA'20	✗	Reinforcement learning-based
FastBot	ByteDance	✗	Model-based
NewMonkey / WeTest	Tencent	✗	Heuristics-based / Reinforcement learning-based

- ✓ Tutorials
- ✓ Bug Dataset
  (52 critical crash bugs)
- ✓ Extensible Infrastructure
- ✓ Other research purposes
  (fault localization, program repair)

对同学们的期望

- 积极参与课堂讨论、随时提问、按时完成作业和课程项目
- 这门课不是编程教学课，所以编程语言和开发知识请利用课余时间自学
- 学会团队合作

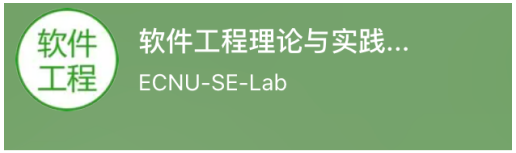
No Pain, No Gain!

课程信息

- 课程网站(https://tingsu.github.io/files/courses/se2024.html)

- 课程助教

  陈浩仪（23级硕士研究生）



扫描群二维码，立刻加入该群

该二维码 7 天内 (9/23前)有效

- 课程交流形式

  1.课件资料：课程飞书群

  2.课程项目：GitHub (https://github.com/)

  3.平时交流：课程飞书群

  4.答疑时间：课前/课后（与我/助教交流）；飞书群（Any Time）；
  发邮件约时间/直接来办公室找我（每周三下午4:00-4:30）

课后作业

课后阅读

- 《Software Engineering-A Practitioner's Approach (Eighth Edition)》,
  Chapter 1 (The Nature of Software);
  Chapter 2 (Software Engineering)

课后准备

- 熟悉VScode、GitHub、Git、大模型(Chatgtp, https://poe.com/)
- 学习Web/iOS/Android/小程序/HarmonyOS Apps开发过程和相关编程语言知识
- 带好自己笔记本、搭建好科学上网工具

小广告

- **欢迎学有余力的同学来我们研究小组做科研实习**
- 良好的科研环境、有挑战的课题、有研究生指导、有科研津贴、推荐保研/企业实习、出国参加国际会议和短期访问…… https://tingsu.github.io/files/application.html

- 黄杉（19级本科、共同一作发表CCF-A类论文1篇、优秀本科毕业论文、本院直博、访问美国）
- 文贺（19级本科、参与发表CCF-A类论文1篇、ACM杰出论文奖、荷兰阿姆斯特丹大学硕士）
- 王可以（20级本科、参与软件安全方向课题研究、本院直研）
- 李丽坤（21级本科、本院直研）、史晓磊（21级本科）、李恩兆（21级本科）……
- 我们实验室所有同学信息：https://tingsu.github.io/files//people.html

- **其他相关课程**
- “程序分析与验证前沿”（本学期的本科选修课，周五上午9:50-11:25）：欢迎有科研兴趣的同学选修！
  https://tingsu.github.io/files/courses/pa2024.html

小广告

- 移动应用软件的功能正确性测试与验证，即如何自动化地发现软件中的功能缺陷，这里涉及到了大模型技术、程序分析技术、基于性质测试等
- 网络协议的二进制逆向分析、模糊测试、安全漏洞挖掘等，涉及到了程序分析技术、污点分析技术等
- 基于符号执行和模糊测试的约束求解研究
- 静态分析工具漏报误报问题的研究，涉及到程序约简技术等
- 编译器优化问题的研究
- 操作系统的安全测试问题研究