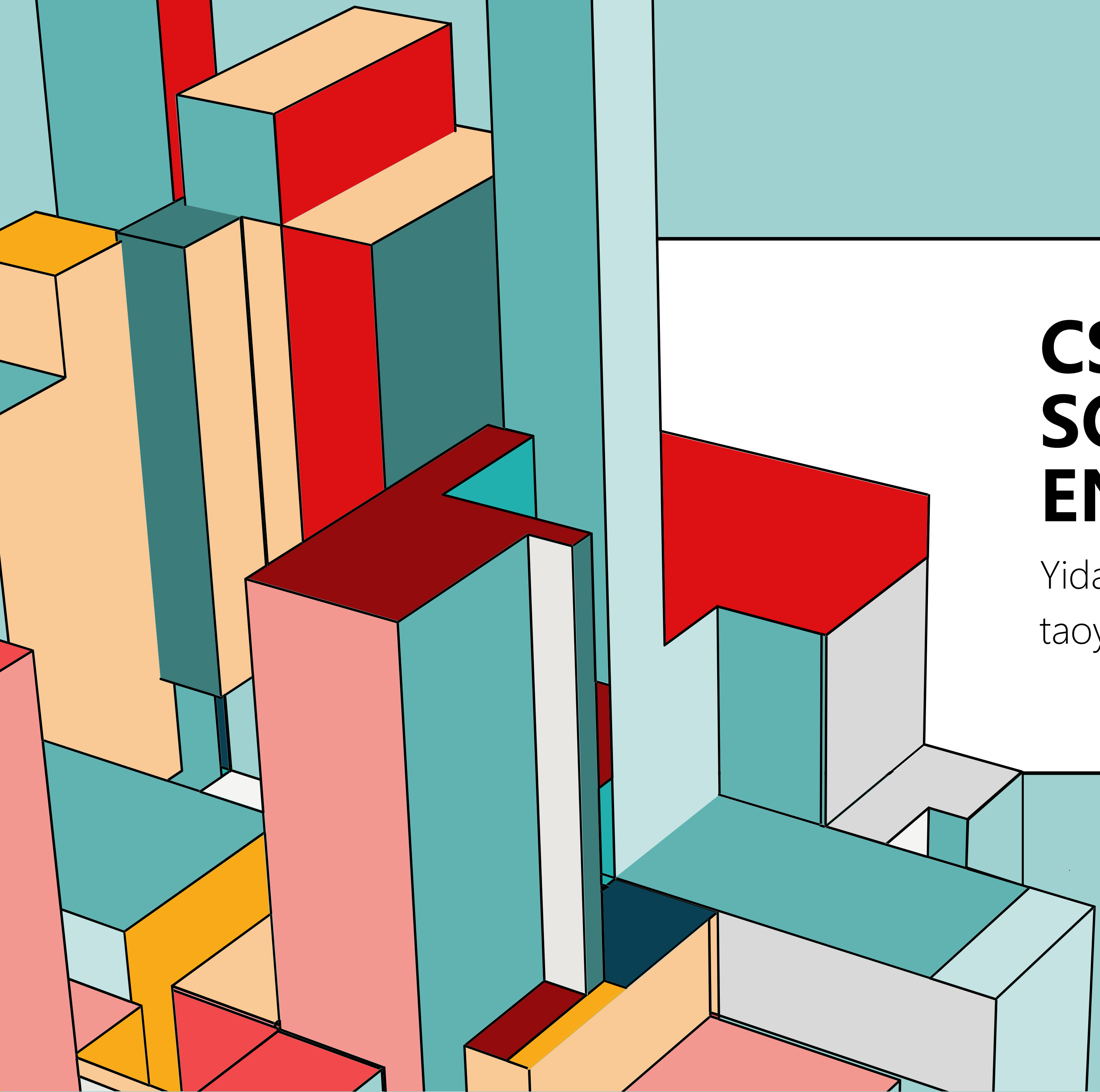


CS304 SOFTWARE ENGINEERING

Yida Tao
taoyd@sustech.edu.cn

COURSE TEAM

Lecturer

Yida Tao (陶伊达) taoyd@sustech.edu.cn

Lab tutor

Yao Zhao (赵耀) zhaoy6@mail.sustech.edu.cn

Daxing Wang (王大兴) wangdx3@mail.sustech.edu.cn

Teaching assistant

潘一诺、王力爽、邓祥波、杨子德、张艺严、张海涵、周子睿

COURSE SCHEDULE

Lecture

01班: Monday 5-6 一教111
02班: Monday 3-4 商学院104

Lab

01班
1组 Tuesday 3-4 三教503 赵耀
2组 Wednesday 3-4 三教503 赵耀
3组 Wednesday 5-6 三教503 赵耀

02班
1组 Tuesday 3-4 三教506 陶伊达
2组 Monday 5-6 三教504 王大兴
3组 Thursday 3-4 三教504 王大兴

COURSE WEBSITE

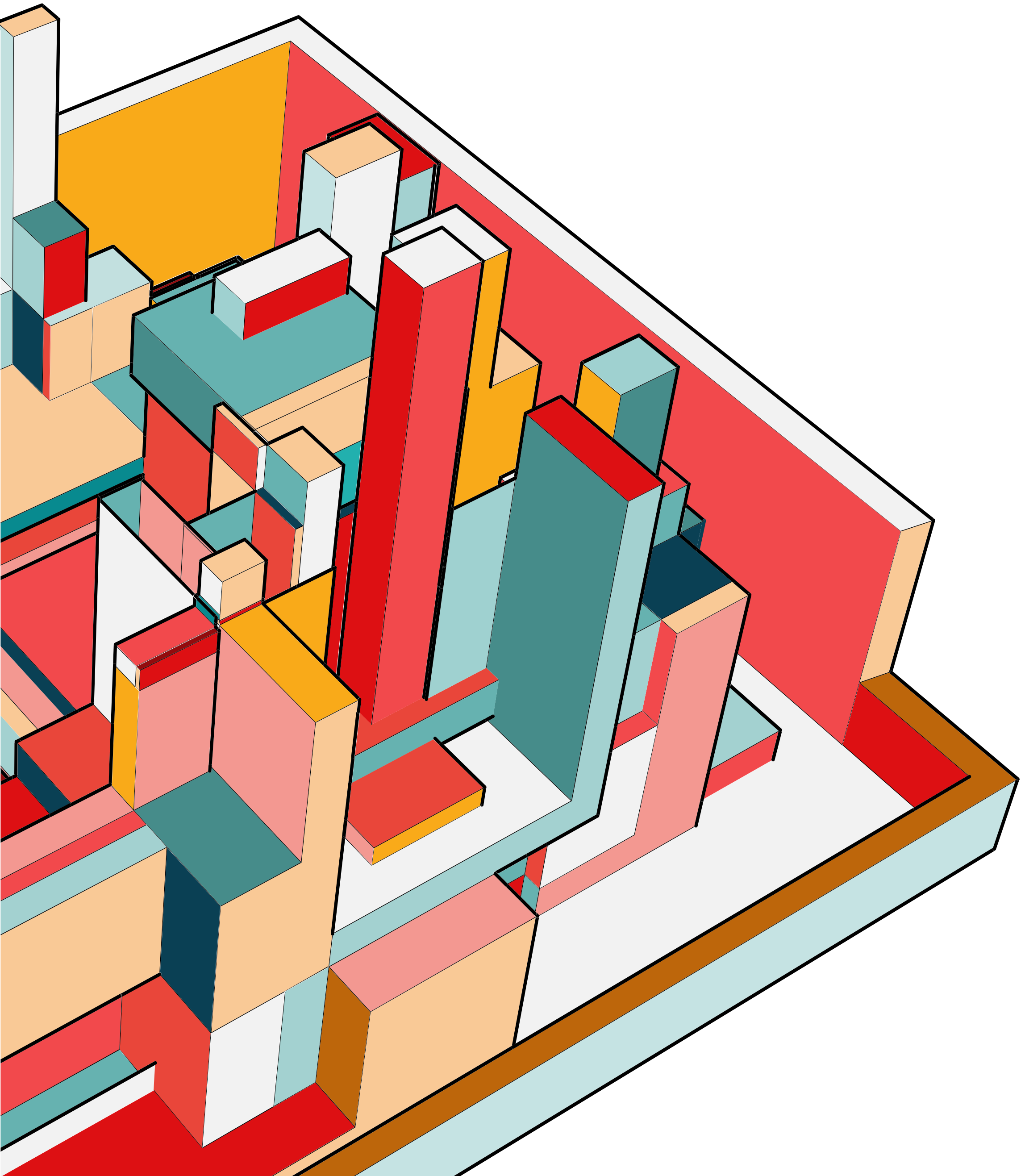
Blackboard

You’ll be automatically added to the enrolled class.
All course resources (slides, notifications, etc.) will be uploaded here.

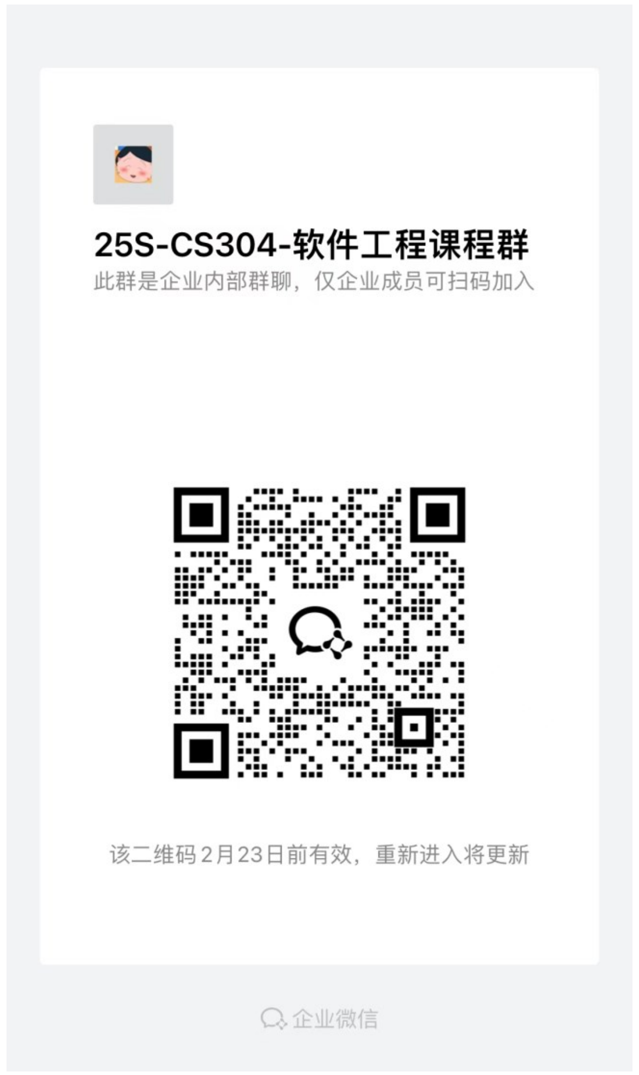
GitHub Classroom:

Team projects, milestones, and assignments will be submitted here.
Details later.
https://classroom.github.com/classrooms/

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY



COURSE GROUP



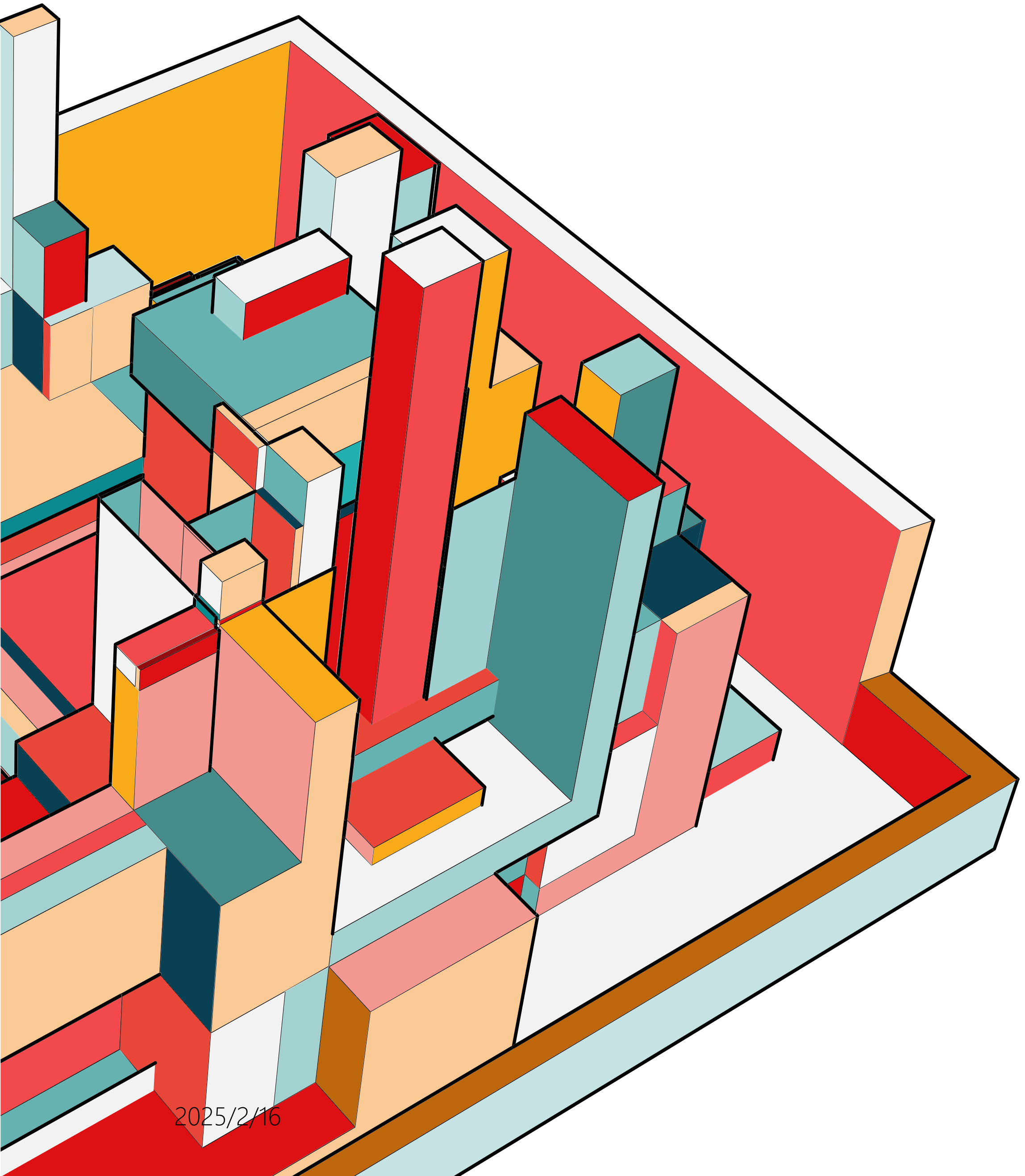
25S-CS304-软件工程课程群

此群是企业内部群聊，仅企业成员可扫码加入

该二维码2月23日前有效，重新进入将更新

企业微信

TAO Yida@SUSTech



LECTURE 1

- What is software?
- How to build a software?
- What is software engineering?
- Objectives of software engineering?
- What will we learn?

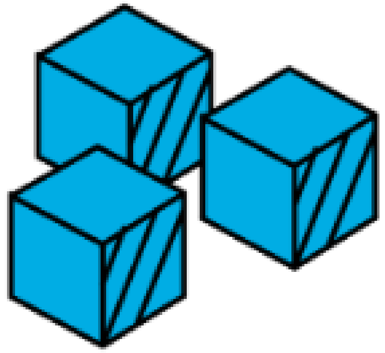
Q1. WHAT IS SOFTWARE?

Is programming assignment a software?

ARTIFACTS

Programming Assignment

- Source code



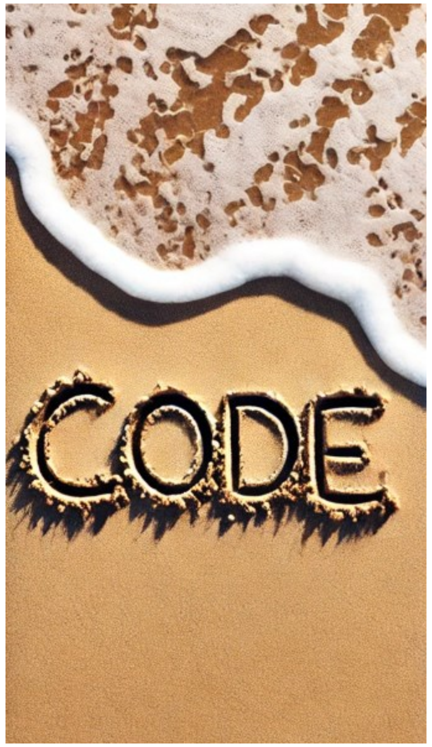
Software

- Source code
- Tests
- Documentation
- Build scripts
- Configurations
- Executables (.exe etc.)
- ……

WHAT'S THE EXPECTED LIFE SPAN OF YOUR CODE?

Programming Assignment

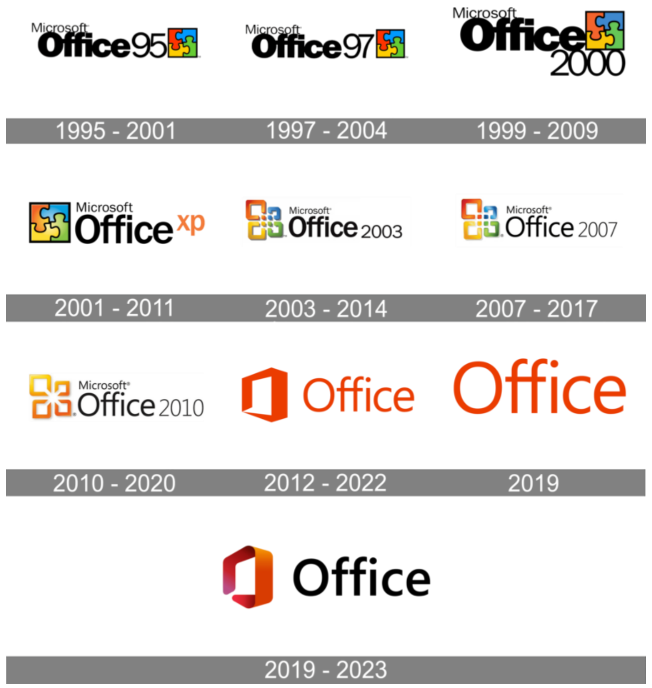
- **Short-term**: your programming code is likely to last for only hours, days, or weeks, not any longer (i.e., decades)
- **No-change**: You probably won't upgrade and maintain your programming code after the assignment deadline ☺



WHAT'S THE EXPECTED LIFE SPAN OF YOUR CODE?

Software

- **Long-term**: large software (e.g., Microsoft Office, Google Search) tend to live for decades
- **Adapt-to-change**: to allow for longer life spans, software needs to adapt to new versions of underlying dependencies, OS, hardware, programming language versions, etc.



HOW MANY RESOURCES ARE INVOLVED?

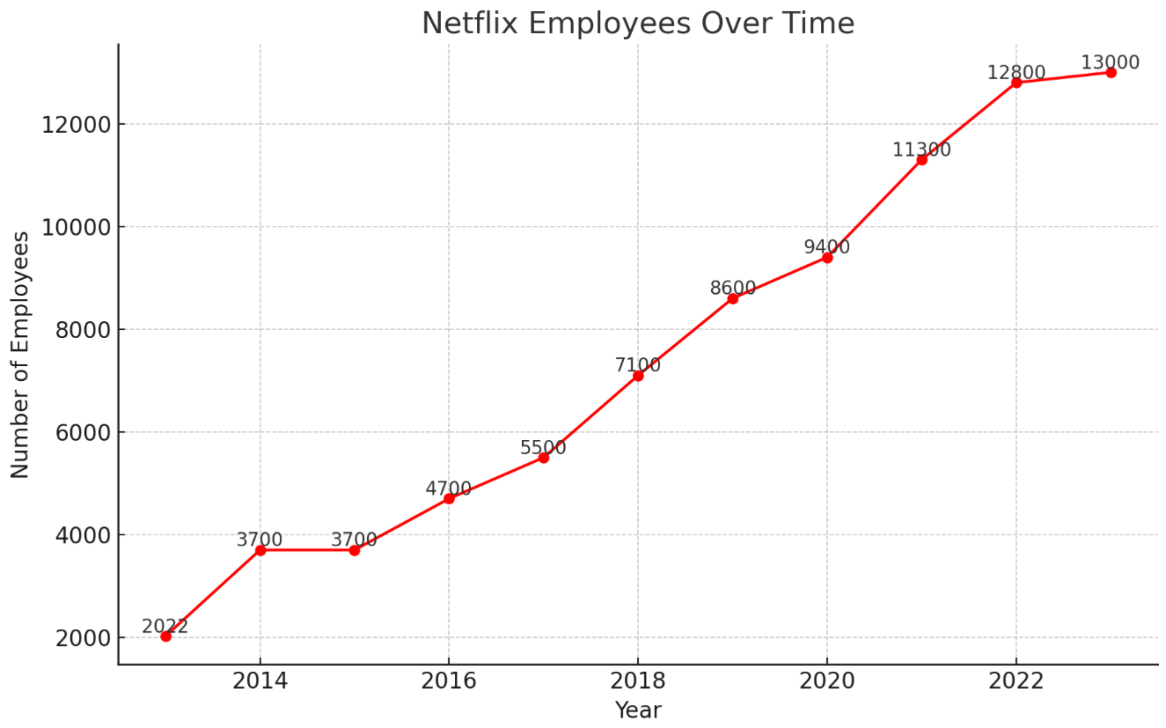
Programming Assignment

- **Human resources**: solo or 2-3 sized small groups
- **Computing resources**: a single laptop is typically sufficient

HOW MANY RESOURCES ARE INVOLVED?

Software

- **Human resources**: large software is developed and maintained by (large) teams
- Computing resources: as organization and users grow, large software needs to scale well with compute, memory, storage, bandwidth resources



HOW MANY RESOURCES ARE INVOLVED?

Software

- **Human resources**: large software is developed and maintained by (large) teams
- **Computing resources**: as organization and users grow, large software needs to **scale** well with compute, memory, storage, bandwidth resources

ChatGPT

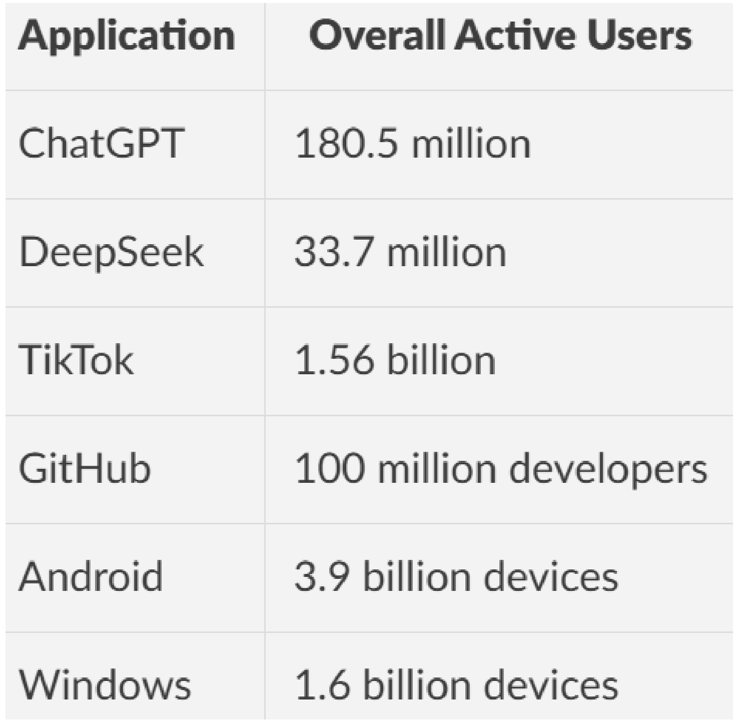
ChatGPT: 1-year running cost of up to 475 million dollars

HOW MANY USERS?

Programming Assignment

- Yourself
- Your team members
- Teachers and TAs

Software

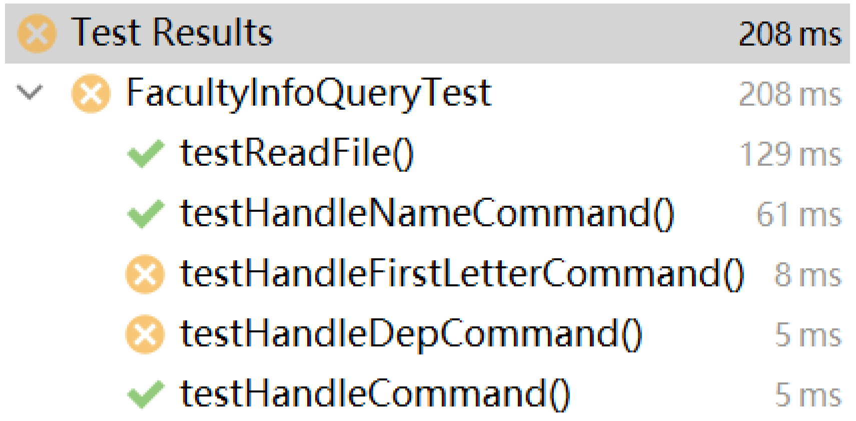


Application	Overall Active Users
ChatGPT	180.5 million
DeepSeek	33.7 million
TikTok	1.56 billion
GitHub	100 million developers
Android	3.9 billion devices
Windows	1.6 billion devices

COMPLEXITY OF DECISIONS

Programming Assignment

- Correctness
- Time (e.g., deadline)



```
Test Results                          208 ms
  FacultyInfoQueryTest                208 ms
    testReadFile()                    129 ms
    testHandleNameCommand()            61 ms
    testHandleFirstLetterCommand()      8 ms
    testHandleDepCommand()              5 ms
    testHandleCommand()                 5 ms
```

Software

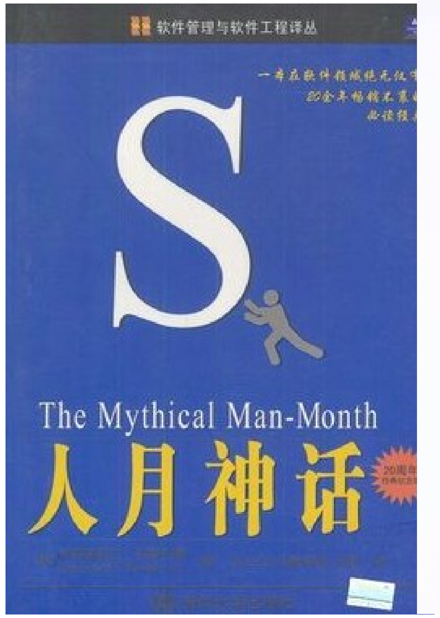
- Software quality
- Engineering efforts
- Financial costs
- Computing resources
- Legal & ethical considerations
- Social impact
- …

Q2. HOW TO BUILD A SOFTWARE?

Building a software == coding?

THE BUILDING OF IBM OS/360

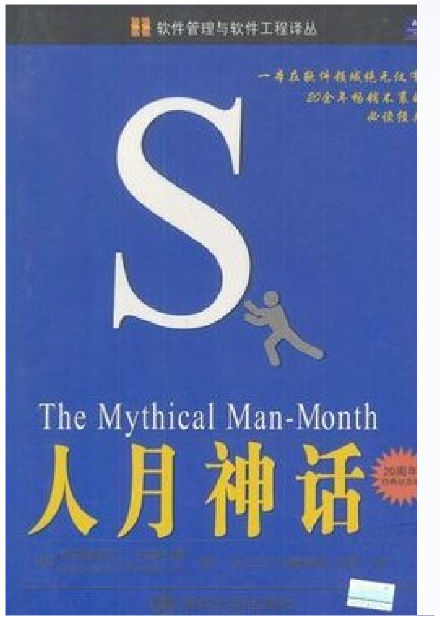
- Time: 1963-1966
- Human involved: 5000 man-month (one person’s working time for a month)
- Codebase: 1M lines of code
- Cost: hundreds of millions $



图灵奖得主、IBM 360系统之父
Frederick Brooks

THE BUILDING OF IBM OS/360

- Deferred releases
- Underestimated cost & memory resources
- Low-quality in first public release
- Thousands of bug fixes even after several releases

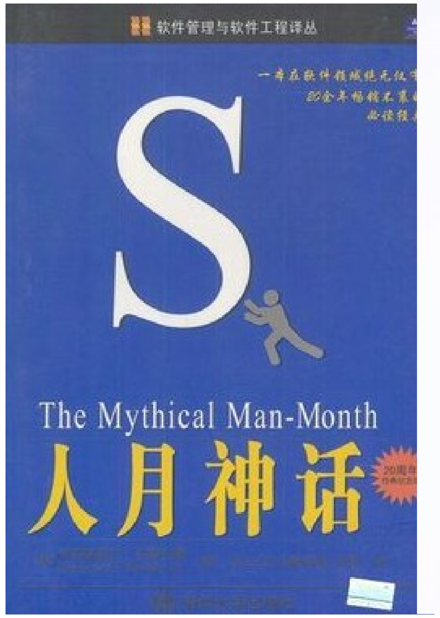


图灵奖得主、IBM 360系统之父
Frederick Brooks

THE BUILDING OF IBM OS/360

Software like a tar pit (焦油坑): The more you fight it, the deeper you sink!

……正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深， 最后无法逃脱灭顶的灾难。……程序设计工作正像这样一个泥潭，……一批批程序员被迫在泥潭中拼命挣扎，……谁也没有料到问题竟会陷入这样的困境……


软件管理与软件工程译丛

The Mythical Man-Month

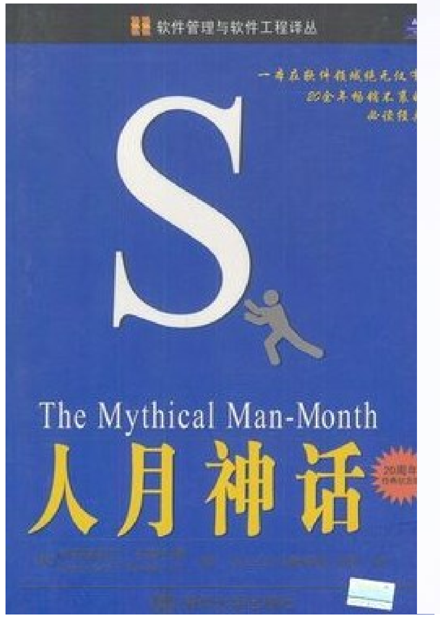
人月神话

图灵奖得主、IBM 360系统之父
Frederick Brooks

TAKEAWAYS

“Adding manpower to a late software project makes it later.”

向进度落后的项目中增加人力，只会让项目更加落后。

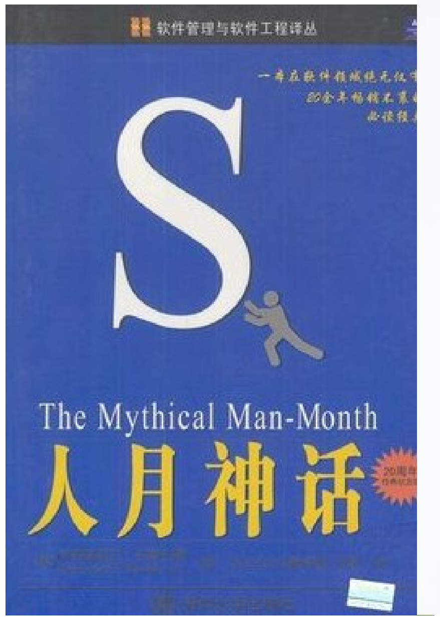


图灵奖得主、IBM 360系统之父
Frederick Brooks

TAKEAWAYS

“Plan to throw one away; you will, anyhow.”

要有丢弃一个版本的心理准备，因为你最终一定会这样做。

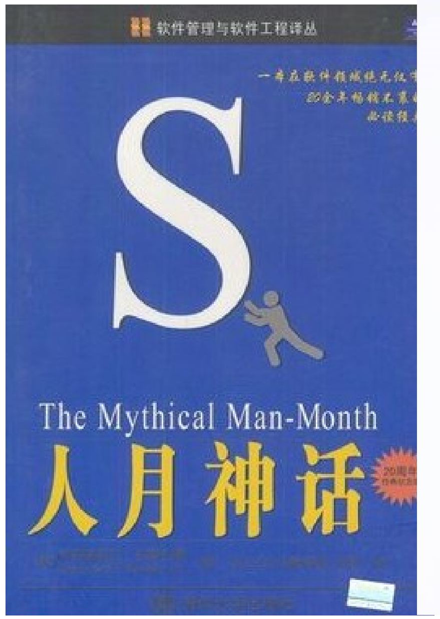


图灵奖得主、IBM 360系统之父
Frederick Brooks

TAKEAWAYS

“The complexity of software is an essential property, not an accidental one. This inherent complexity is what makes software difficult to build and harder to scale.”

软件开发本质上是复杂的。这种本质复杂性使得软件开发困难且难以扩展

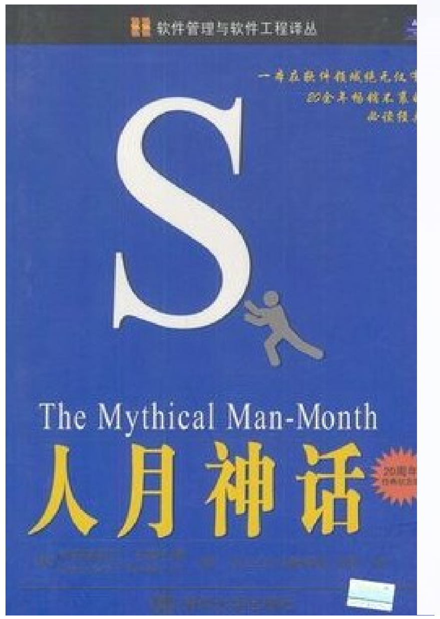


图灵奖得主、IBM 360系统之父
Frederick Brooks

TAKEAWAYS

No Silver Bullet: There is no single development method or technology that can dramatically improve software productivity.

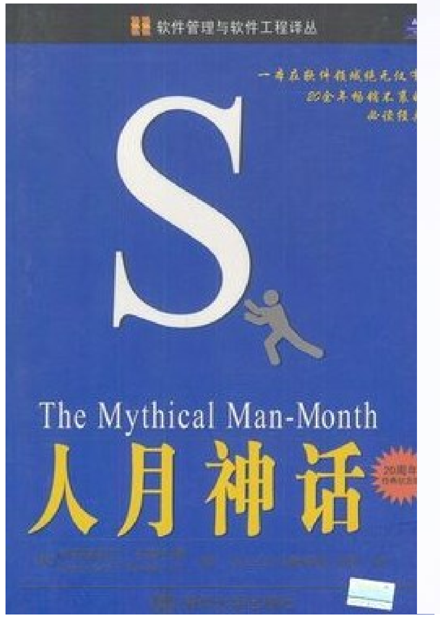
没有任何单一方法或技术可以大幅提升软件生产力。



图灵奖得主、IBM 360系统之父
Frederick Brooks

All programmers say "Oh, I can easily beat the xx lines / day cited by industrial programmers."

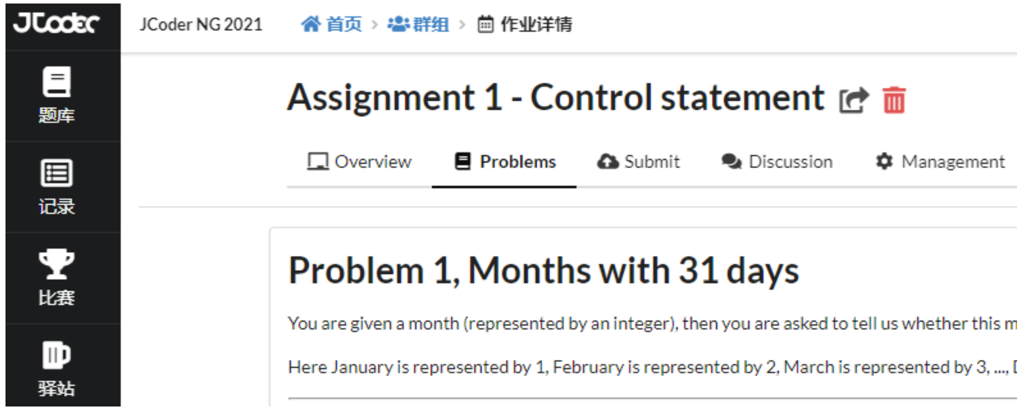
They are talking about just **coding something, not building a product**.



图灵奖得主、IBM 360系统之父
Frederick Brooks

BUILDING A SOFTWARE VS. PROGRAMMING

- **Programming** is a significant part of software engineering
  - Programming is how you generate a new software in the first place
- **Building a software** is programming integrated over **time, scale, and trade-offs**



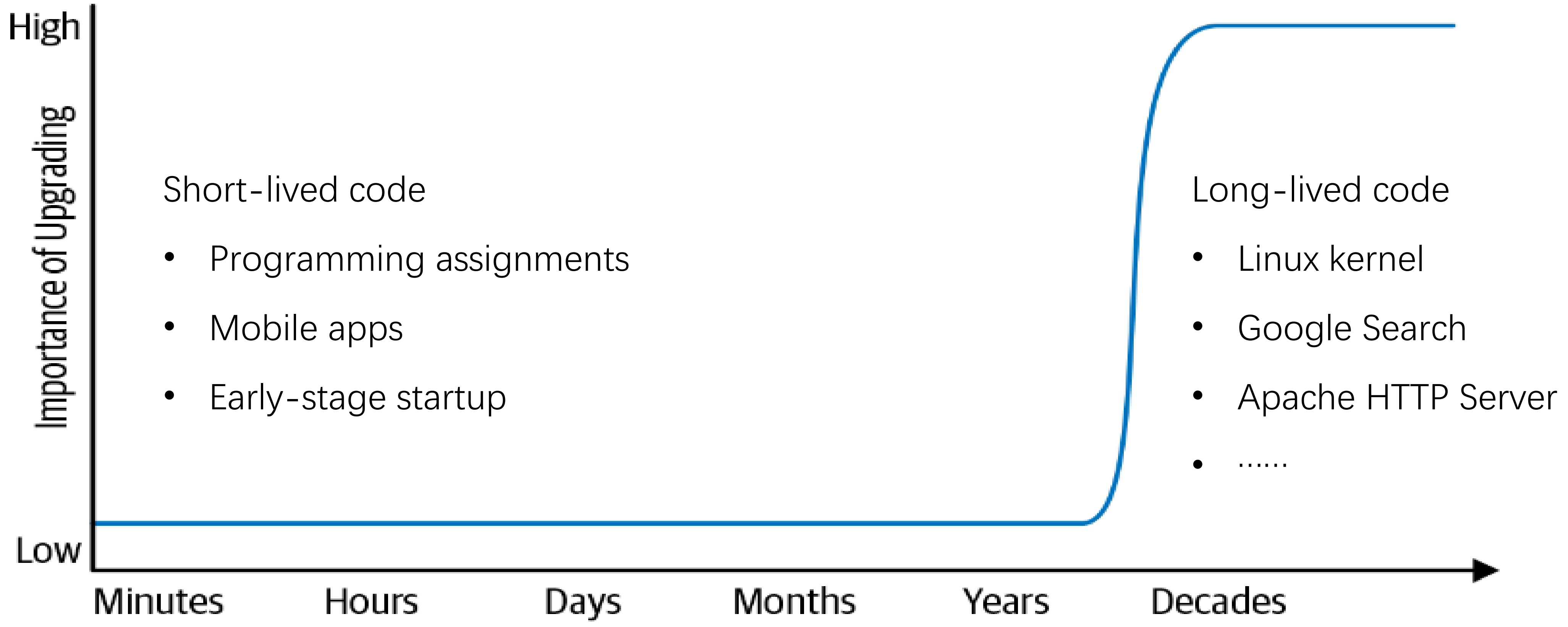
Assignment 1 - Control statement

Problem 1, Months with 31 days

You are given a month (represented by an integer), then you are asked to tell us whether this m

Here January is represented by 1, February is represented by 2, March is represented by 3, ..., I

LIFE SPAN OF SOFTWARE



Short-lived code

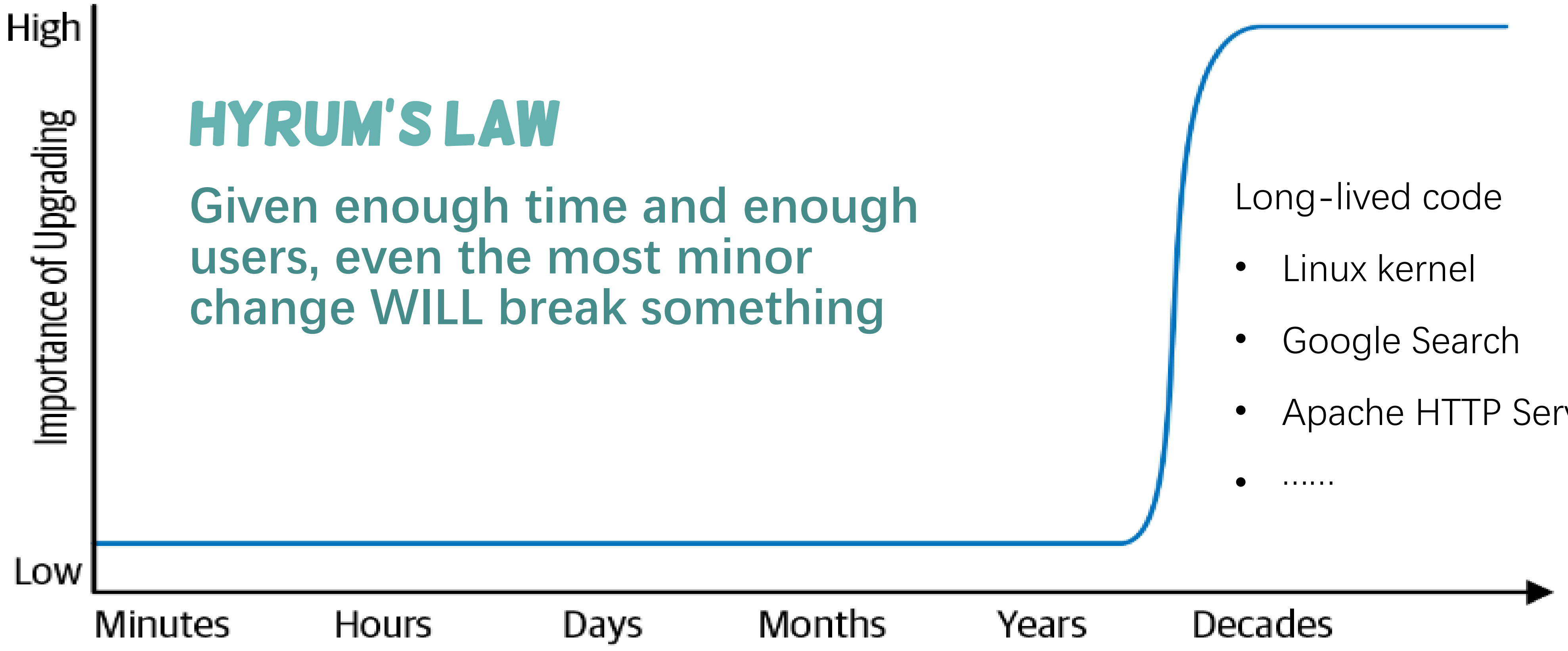
- Programming assignments
- Mobile apps
- Early-stage startup

Long-lived code

- Linux kernel
- Google Search
- Apache HTTP Server
- ……

Figure 1-1. Life span and the importance of upgrades

LIFE SPAN OF SOFTWARE



HYRUM'S LAW

Given enough time and enough users, even the most minor change WILL break something

Long-lived code

- Linux kernel
- Google Search
- Apache HTTP Server
- ……

Figure 1-1. Life span and the importance of upgrades

HYRUM'S LAW IN PRACTICE

```
>>> for i in {"apple", "banana", "carrot", "durian", "eggplant"}: print(i)
...
durian
carrot
apple
eggplant
banana
```

- User 1: assume that the hash iteration ordering is fixed and write some code that depends on the ordering
- User 2: assume that the hash iteration ordering is random, and use it as an inefficient random-number generator
- User 3: assume that the hash iteration ordering is random and never write any code that depend on the ordering.

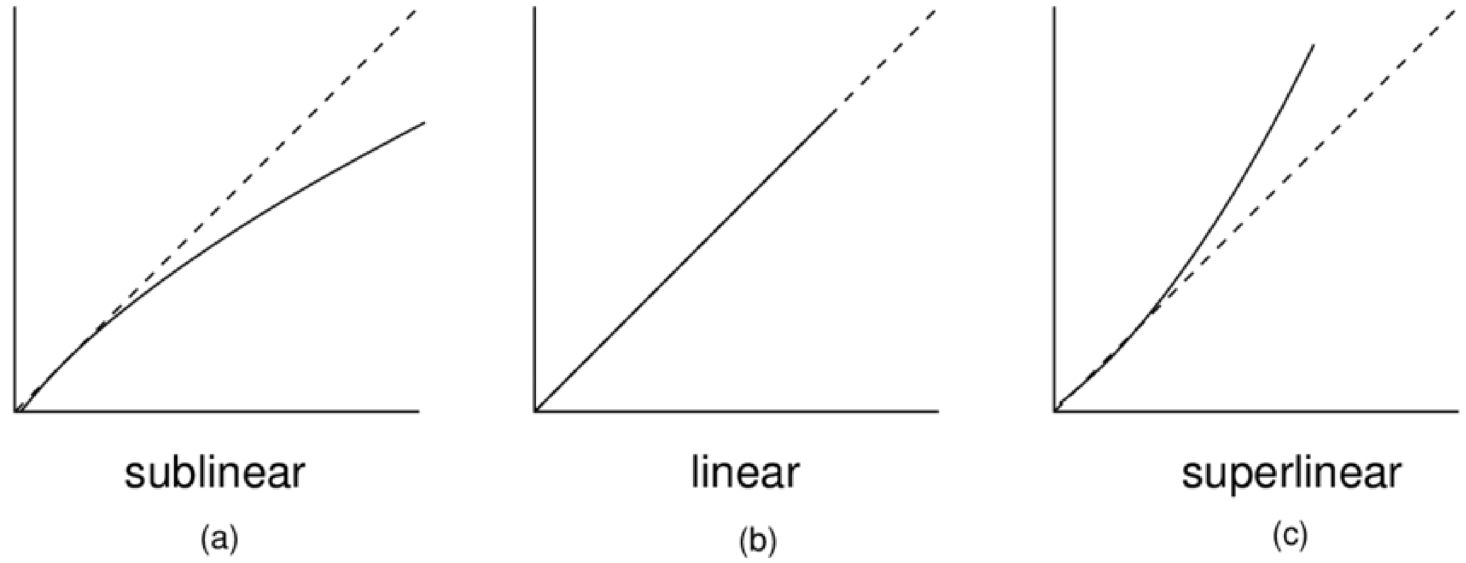
User 4 breaks if the results produced by user 3 are consumed by user 4, whose code depends on the order of the results

User 1 breaks if the hash ordering becomes randomized

User 2 breaks if the hash ordering becomes fixed

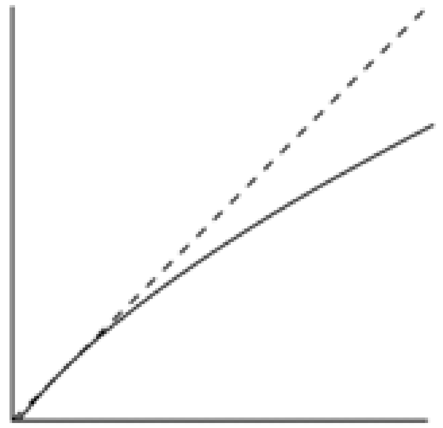
SCALE

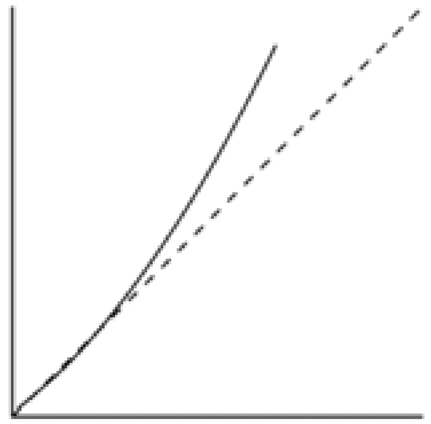
- Your organization's codebase is sustainable when you are able to change all of the things that you ought to change, safely, and can do so for the life of your codebase.
- If costs grow **superlinearly** over time, the operation clearly is **not scalable**.



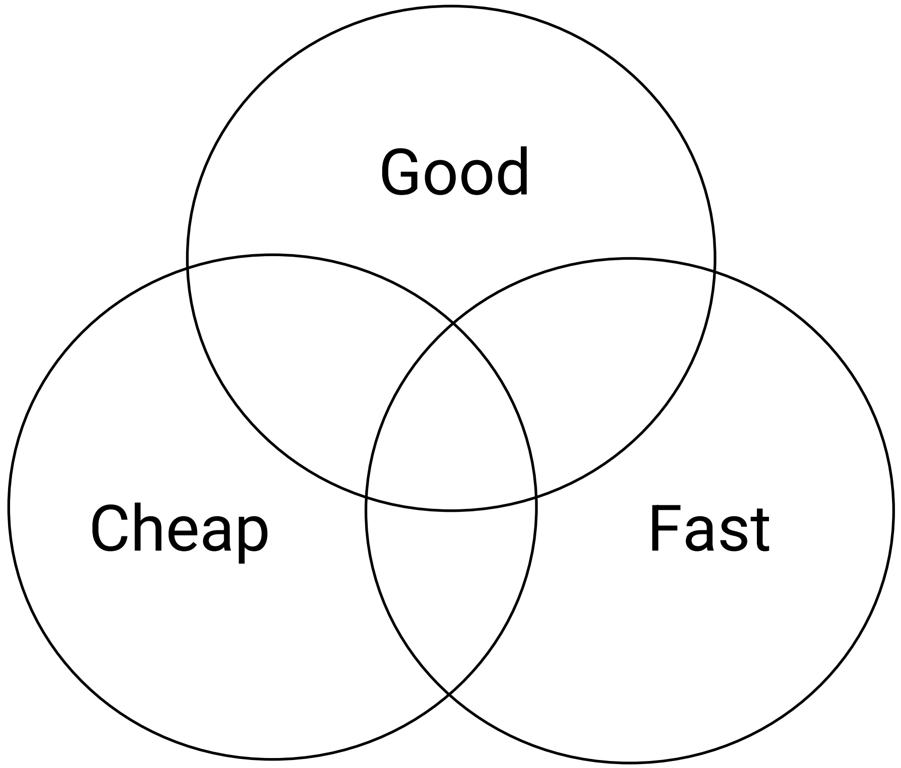
SCALE

- X axis: the demand (e.g., LOC)
- Y axis: resources costs
    - How many additional computing resources (e.g., memory, storage) are needed?
    - How many additional human involvement is needed?
    - How long does it take to do a full build?
    - How long does it take to pull a fresh copy of the repository?
    - How much will it cost to fix a bug or upgrade to a new language version?
    - How long does it take to respond to users?
    - ….


sublinear


superlinear

TRADEOFF



Q3. WHAT IS SOFTWARE ENGINEERING?

SOFTWARE CRISIS

The crisis manifested itself in several ways:

- Projects running over-budget
- Projects running over-time
- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code difficult to maintain
- Software was never delivered

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

THE ORIGIN OF SOFTWARE ENGINEERING

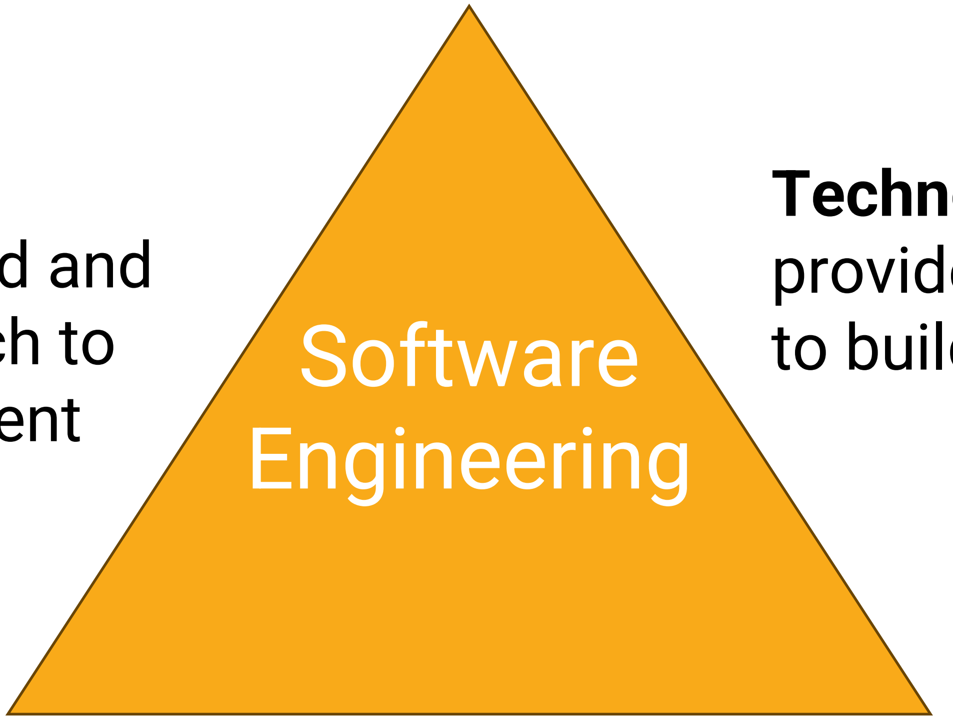
Software Engineering was first formally used by Professor Friedrich L. Bauer, at NATO conference, the first conference on software engineering, in 1968:

"The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."

Software engineering is now considered one of major computing disciplines.

NATO conference, 1968

WHAT IS SOFTWARE ENGINEERING?



Engineering:
ensures a structured and systematic approach to software development

Technology:
provides the tools and platforms to build and implement solutions

Management: organizes teams and processes for efficient execution

WHAT IS SOFTWARE ENGINEERING?

Software engineering: the branch of engineering that deals with the design, development, testing, and maintenance of software applications, while ensuring that the software to be built is:

- Correct
- Consistent
- On budget
- On time
- Align with the requirements.

Q4. OBJECTIVE OF SOFTWARE ENGINEERING?

1. **Quality**: improve the quality of software
2. **Efficiency**: improve the efficiency of building software

SOFTWARE BUGS ARE ANNOYING

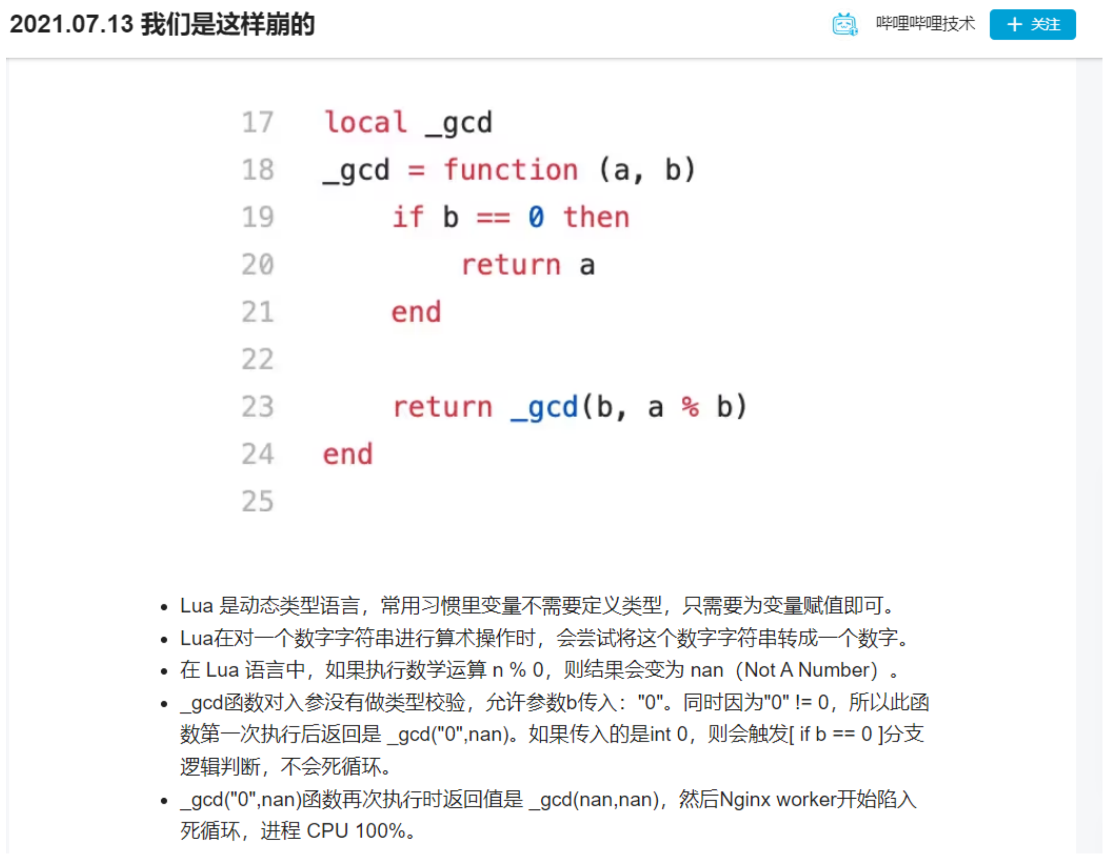
一行代码导致B站崩溃了三小时。

近日，B站官方发布了一篇文章"2021.07.13 我们是这样崩溃的"，详细回顾了2021.07.13日B站崩溃事件。

B站同学从23：25到23：55一直尝试各种方式恢复服务，都未能达到预期效果，最后只能全部重建SLB集群。

从事件发生-> 初步定位 ->尝试修复->全部重建->临时解决->逐步恢复，时间从22：52持续到01：50，经历了大约三个小时的时间，被称为B站的至暗时刻。

有人就在此算了一笔账，称就是这7行代码，让b站老板一下亏了大约1，5750，0000元。



2021.07.13 我们是这样崩的

哔哩哔哩技术 + 关注

```
local _gcd
_gcd = function (a, b)
    if b == 0 then
        return a
    end

    return _gcd(b, a % b)
end

```

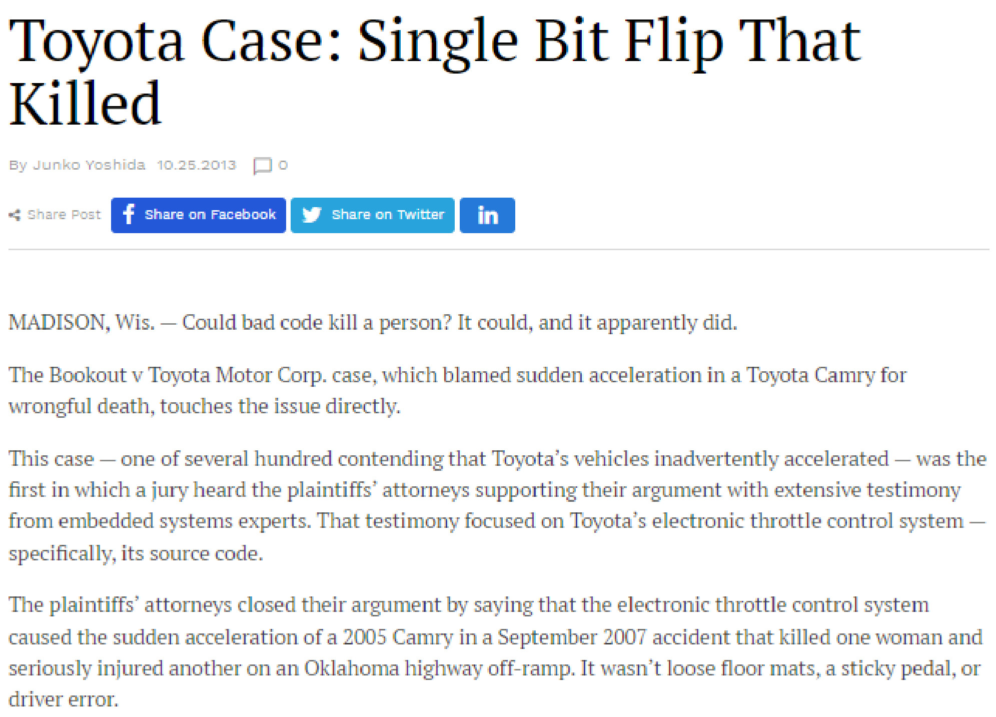
- Lua 是动态类型语言，常用习惯里变量不需要定义类型，只需要为变量赋值即可。
- Lua在对一个数字字符串进行算术操作时，会尝试将这个数字字符串转成一个数字。
- 在 Lua 语言中，如果执行数学运算 n % 0，则结果会变为 nan（Not A Number）。
- _gcd函数对入参没有做类型校验，允许参数b传入："0"。同时因为"0" != 0，所以此函数第一次执行后返回是 _gcd("0",nan)。如果传入的是int 0，则会触发[if b == 0]分支逻辑判断，不会死循环。
- _gcd("0",nan)函数再次执行时返回值是 _gcd(nan,nan)，然后Nginx worker开始陷入死循环，进程 CPU 100%。

SOFTWARE BUGS ARE EXPENSIVE

- Ariane 5 Rocket Explosion (1996) – A software bug in the rocket’s guidance system caused an integer overflow, leading to a catastrophic failure just 37 seconds after launch, resulting in a loss of $370 million.

- Knight Capital Trading Glitch (2012) – A faulty trading algorithm caused Knight Capital to lose $440 million in just 45 minutes due to unintended stock trades.

SOFTWARE BUGS ARE DEADLY



Toyota Case: Single Bit Flip That Killed

By Junko Yoshida 10.25.2013 0

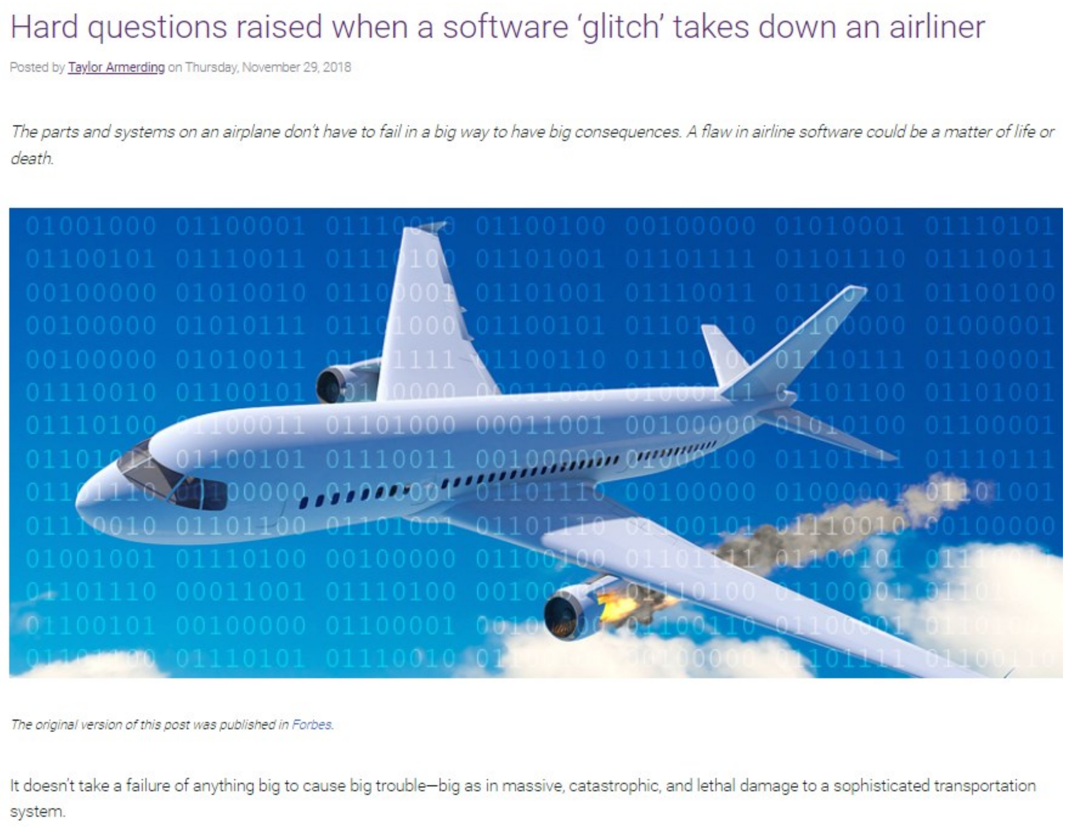
Share Post | Share on Facebook | Share on Twitter | in

MADISON, Wis. — Could bad code kill a person? It could, and it apparently did.

The Bookout v Toyota Motor Corp. case, which blamed sudden acceleration in a Toyota Camry for wrongful death, touches the issue directly.

This case — one of several hundred contending that Toyota's vehicles inadvertently accelerated — was the first in which a jury heard the plaintiffs' attorneys supporting their argument with extensive testimony from embedded systems experts. That testimony focused on Toyota's electronic throttle control system — specifically, its source code.

The plaintiffs' attorneys closed their argument by saying that the electronic throttle control system caused the sudden acceleration of a 2005 Camry in a September 2007 accident that killed one woman and seriously injured another on an Oklahoma highway off-ramp. It wasn't loose floor mats, a sticky pedal, or driver error.



Hard questions raised when a software 'glitch' takes down an airliner

Posted by Taylor Armerding on Thursday, November 29, 2018

The parts and systems on an airplane don't have to fail in a big way to have big consequences. A flaw in airline software could be a matter of life or death.

The original version of this post was published in Forbes.

It doesn't take a failure of anything big to cause big trouble—big as in massive, catastrophic, and lethal damage to a sophisticated transportation system.

IMPROVE THE EFFICIENCY

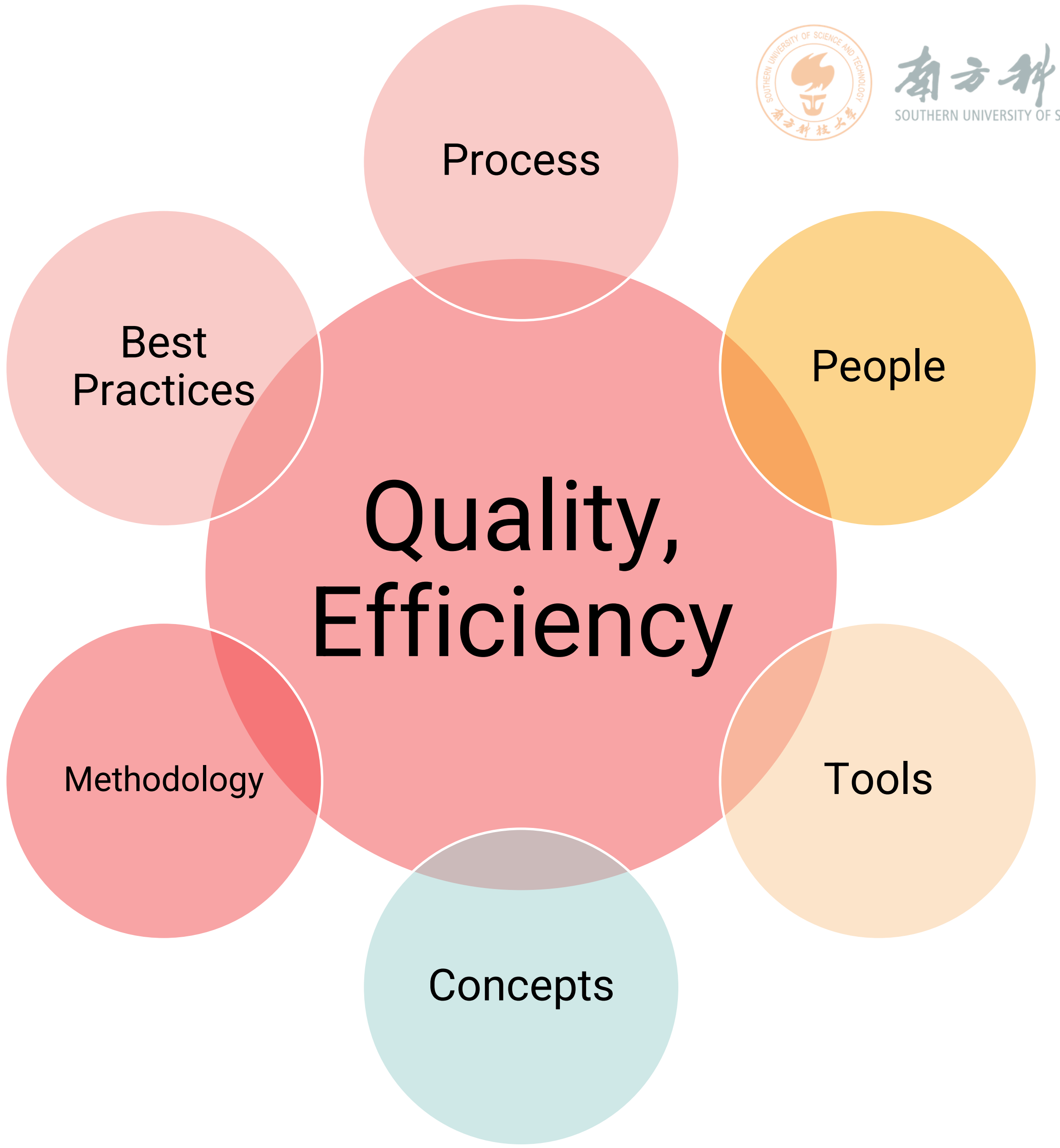
DevOps, a software engineering practice, **improves efficiency dramatically**:

- 8000x faster deployment time
- 21% less time spent on unplanned work and rework
- 44% more time on new work
- 50% lower change-failure rates
- 50% less time spent fixing security issues
- 50% higher market cap growth over 3 years

https://services.google.com/fh/files/misc/state-of-devops-2014.pdf

Q5. WHAT WILL WE LEARN IN THE COURSE?

COURSE THEMES



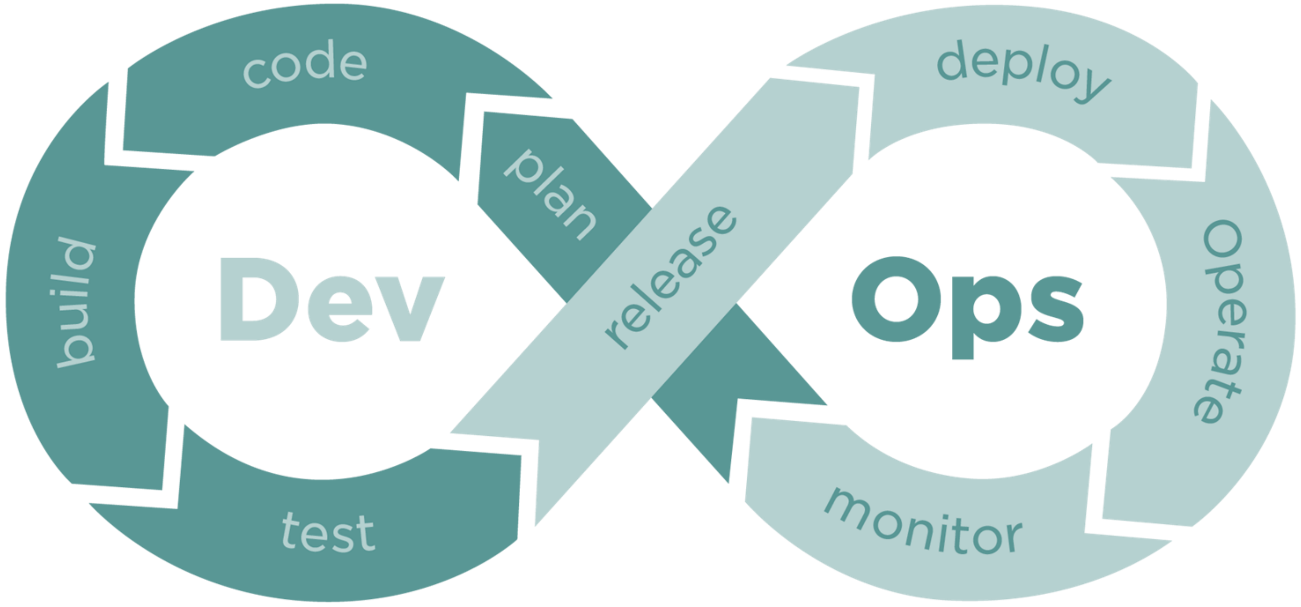
COURSE THEMES

The themes may change as the course progresses.

1. Intro to Software Engineering
2. Software Process & DevOps
3. Version Control Systems
4. Software Requirements
5. Software Design
6. Software Architecture
7. Build & Dependency Management
8. Software Documentation
9. Software Testing
10. Continuous Integration
11. Continuous Deployment
12. Software Quality & Metrics
13. Software Evolution & Maintenance
14. AI in Software Engineering

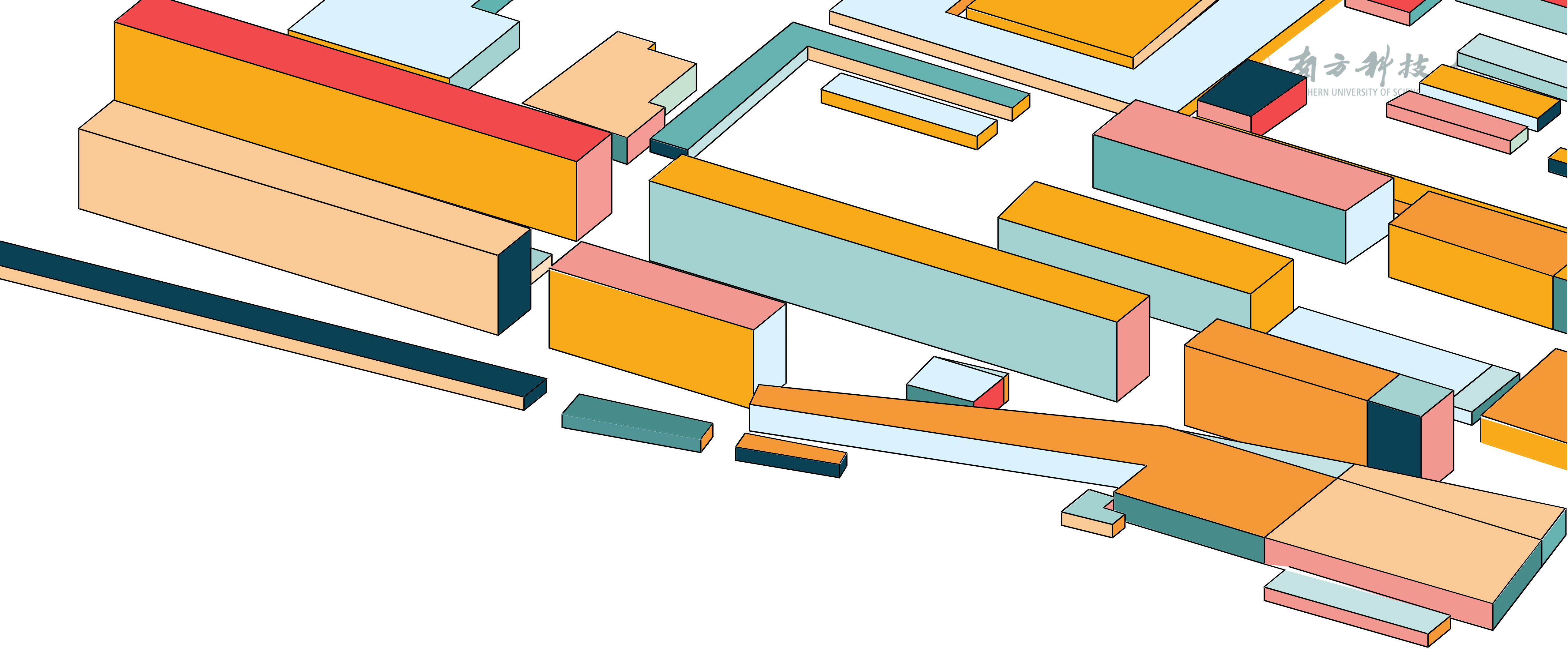
GETTING LOST IN THE COURSE?

1. Intro to Software Engineering
2. Software Process & DevOps
3. Version Control Systems
4. Software Requirements
5. Software Design
6. Software Architecture
7. Build & Dependency Management
8. Software Documentation
9. Software Testing
10. Continuous Integration
11. Continuous Deployment
12. Software Quality & Metrics
13. Software Evolution & Maintenance
14. AI in Software Engineering

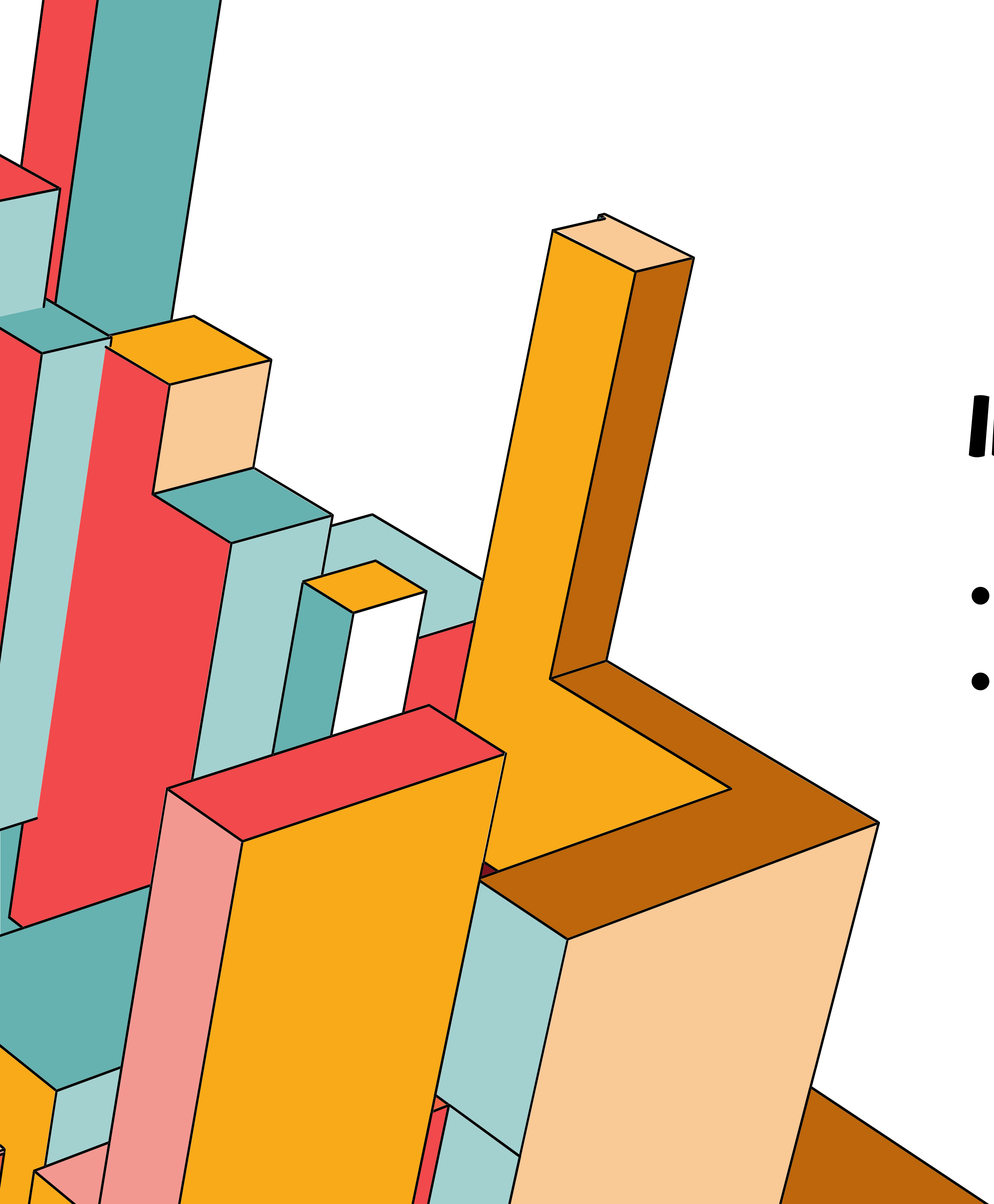


GETTING LOST IN THE COURSE?

1. Intro to Software Engineering
2. Software Process & DevOps
3. Version Control Systems
4. Software Requirements
5. Software Design
6. Software Architecture
7. Build & Dependency Management
8. Software Documentation
9. Software Testing
10. Continuous Integration
11. Continuous Deployment
12. Software Quality & Metrics
13. Software Evolution & Maintenance
14. AI in Software Engineering



COURSEWORK

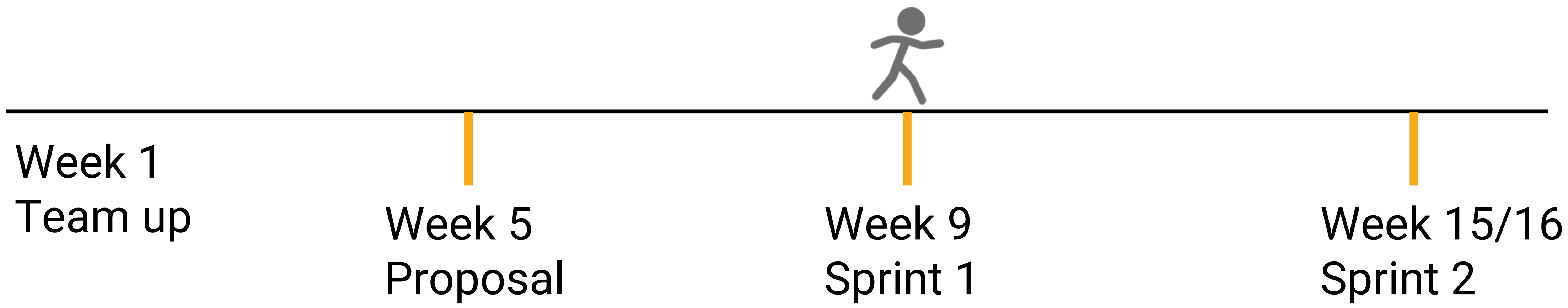


INDIVIDUAL COURSEWORK

- Assignments (x2)
- Weekly lab practices (x10)

TEAM PROJECT

- Medium-sized projects that mirror realistic settings
- Work in a fixed team of 4-5 people throughout the semester
- 3 milestones, each with reports, deliverables, and presentations



Find your team NOW!

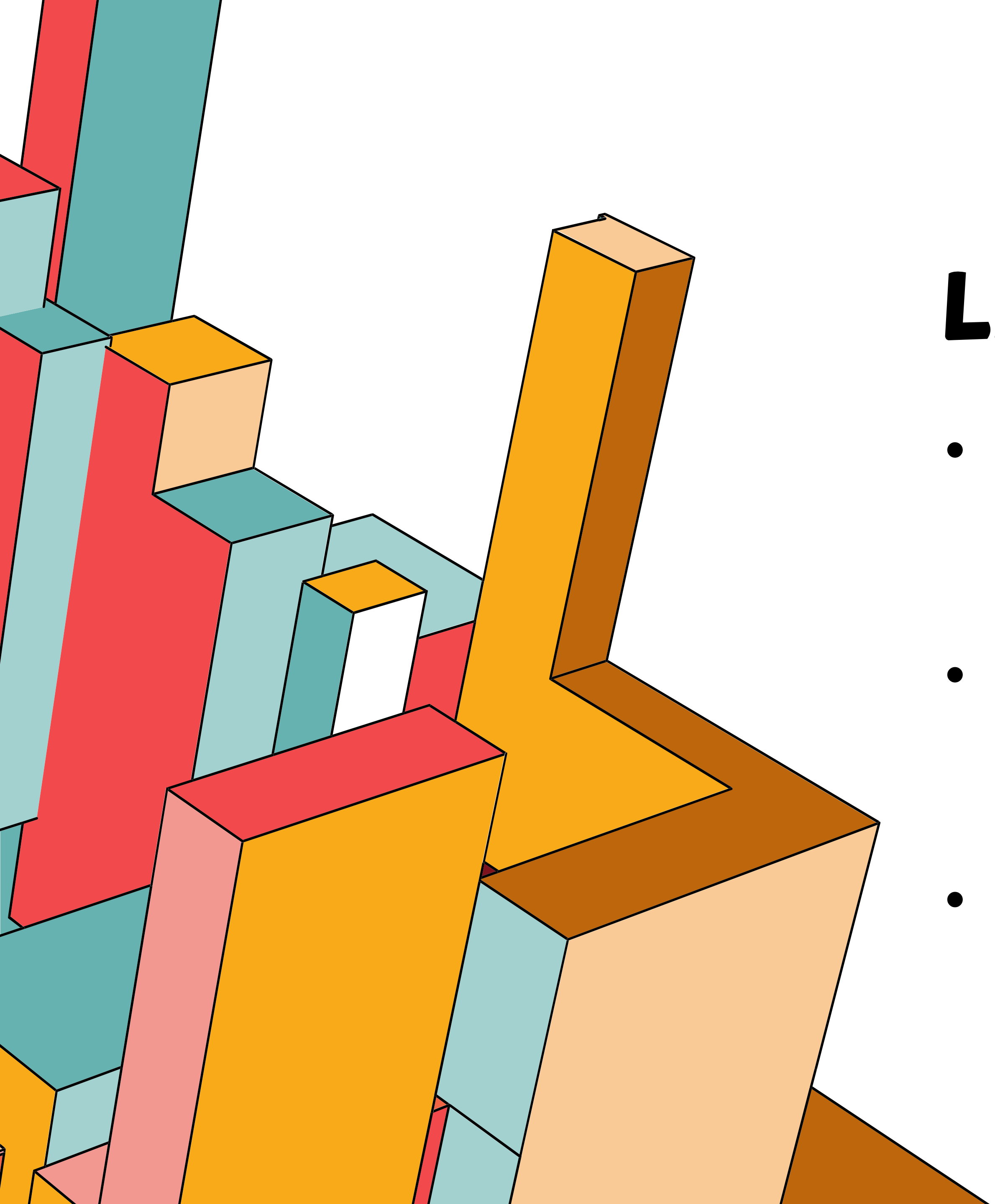
TEAM PROJECT

1. Personal Health Assistant
2. Smart Photo Album
3. Coursework Grading System
4. Intelligent Course-Aware IDE
5. SUSTech Car Racing Game

More project details on Lab 1

GRADING POLICY

Lectures (attendance + quiz)	10%
Individual Assignments	10%
Lab (attendance + practices)	15%
Team Project - Proposal: 5% - Sprint 1: 10% - Sprint 2: 20%	35%
Final Exam (close-book)	30%



LATE DAY POLICY

- Individual assignments
  - No late day
- Lab practices
  - 20% penalty after the deadline
- Project milestones
  - No late day

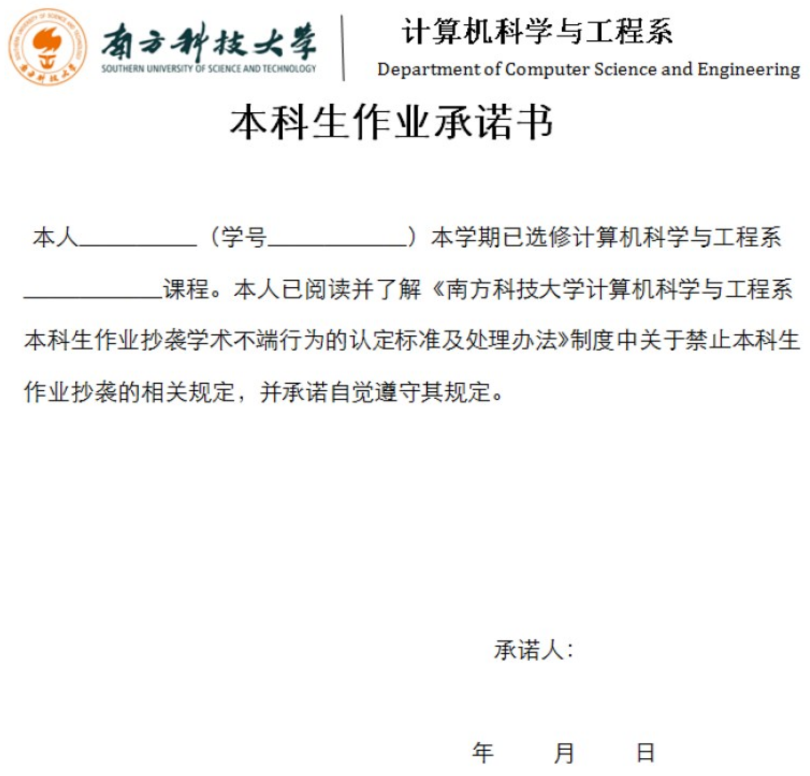
GETTING LOST IN THE COURSE?

- The principles, concepts, practices and methodology introduced in this course all emerge from growing experiences of building real software
- To acquire a better understanding of the lectures, take labs and the team project seriously.

ACADEMIC INTEGRITY

You are encouraged to leverage AI tools throughout the development process of your assignments and team project.

However, any code, scripts, configurations, documentation, or other artifacts generated or influenced by AI must be accompanied by clear and concise references.



计算机科学与工程系
Department of Computer Science and Engineering

本科生作业承诺书

本人________（学号________）本学期已选修计算机科学与工程系________课程。本人已阅读并了解《南方科技大学计算机科学与工程系本科生作业抄袭学术不端行为的认定标准及处理办法》制度中关于禁止本科生作业抄袭的相关规定，并承诺自觉遵守其规定。

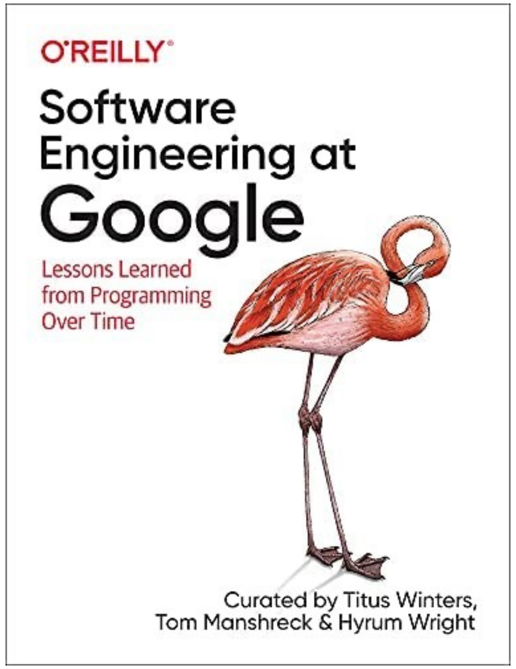
承诺人：

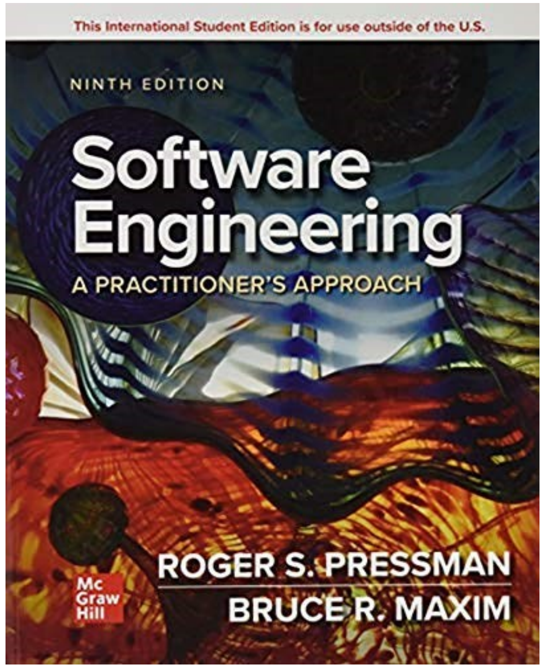
年　　月　　日

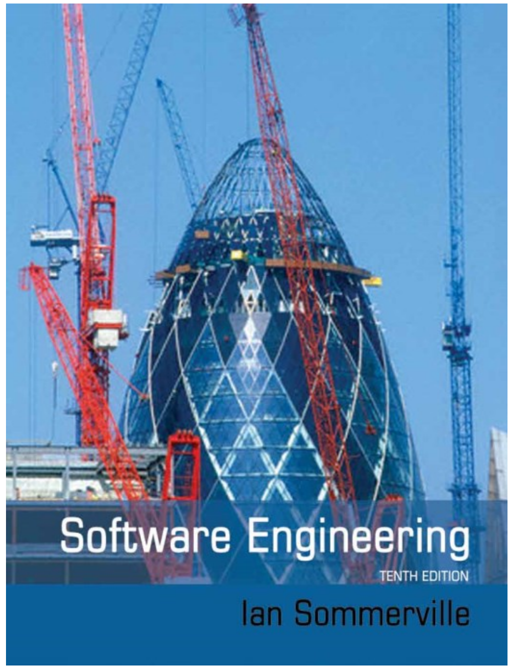
Please submit the form before the end of the course selection & drop period!

REFERENCE

There is no single textbook, but you may find the following books useful


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright

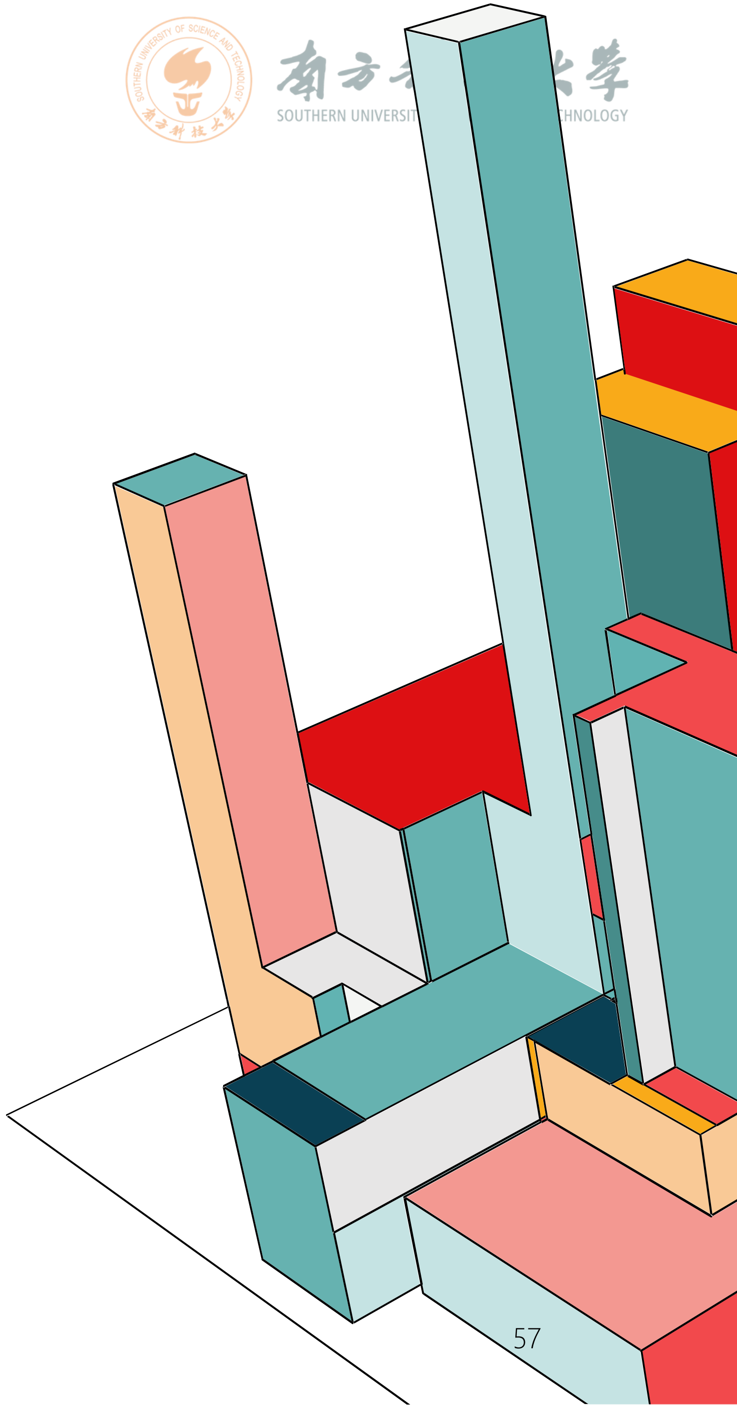

This International Student Edition is for use outside of the U.S.
NINTH EDITION
Software Engineering
A PRACTITIONER'S APPROACH
ROGER S. PRESSMAN
BRUCE R. MAXIM
Mc Graw Hill

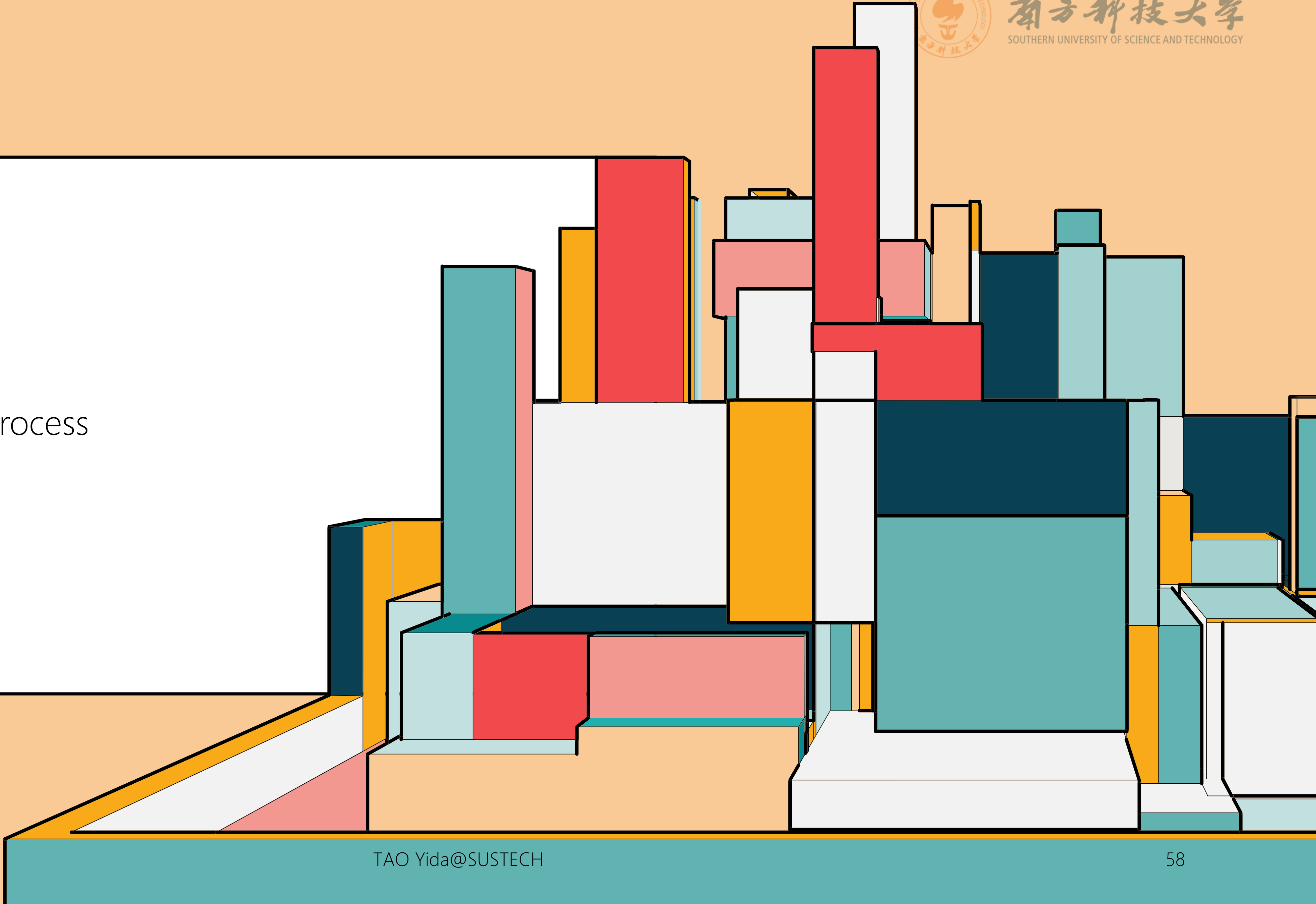


教育部高等学校软件工程专业教学指导委员会
软件工程专业推荐教材
高等学校软件工程专业系列教材
现代软件工程基础
彭鑫 游依勇 赵文耘 ◎ 编著
教学课件
教学视频
示例代码
课程实践
清華大学出版社

READINGS

- Chapter 4. What is Software Engineering? Software Engineering at Google by Titus Winters, et al.





NEXT

- Software process
- DevOps