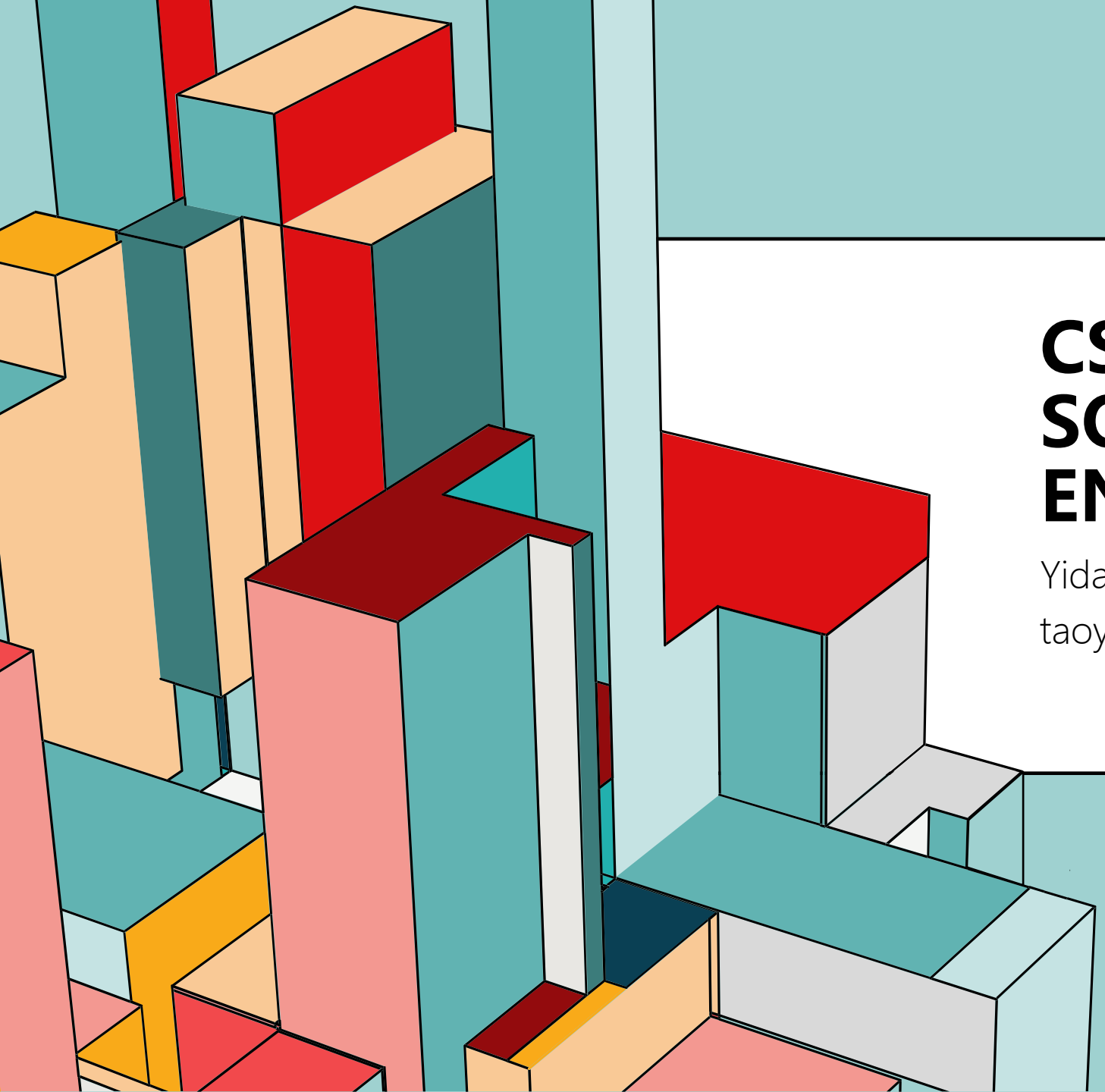


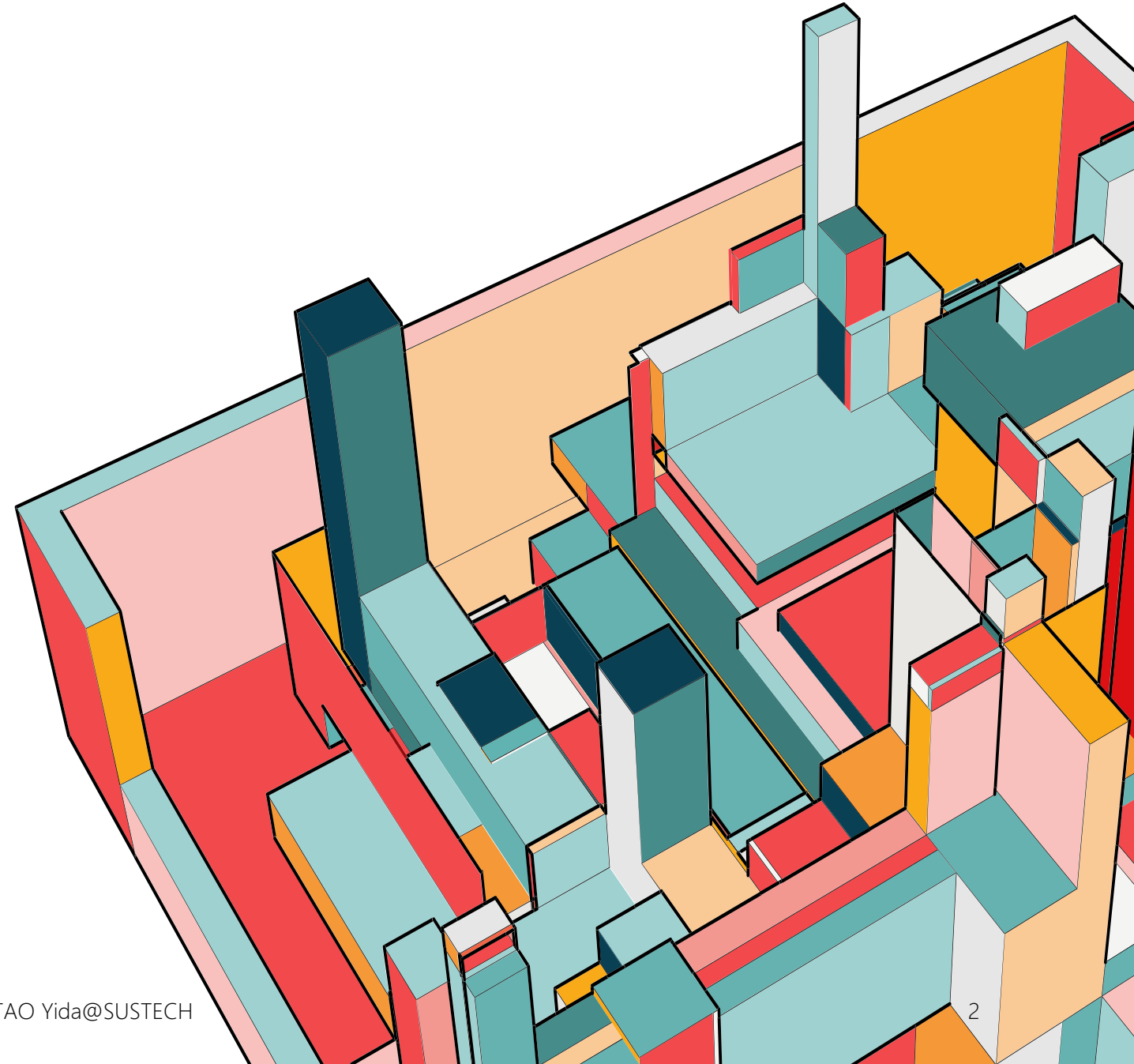
CS304 SOFTWARE ENGINEERING

Yida Tao

taoyd@sustech.edu.cn

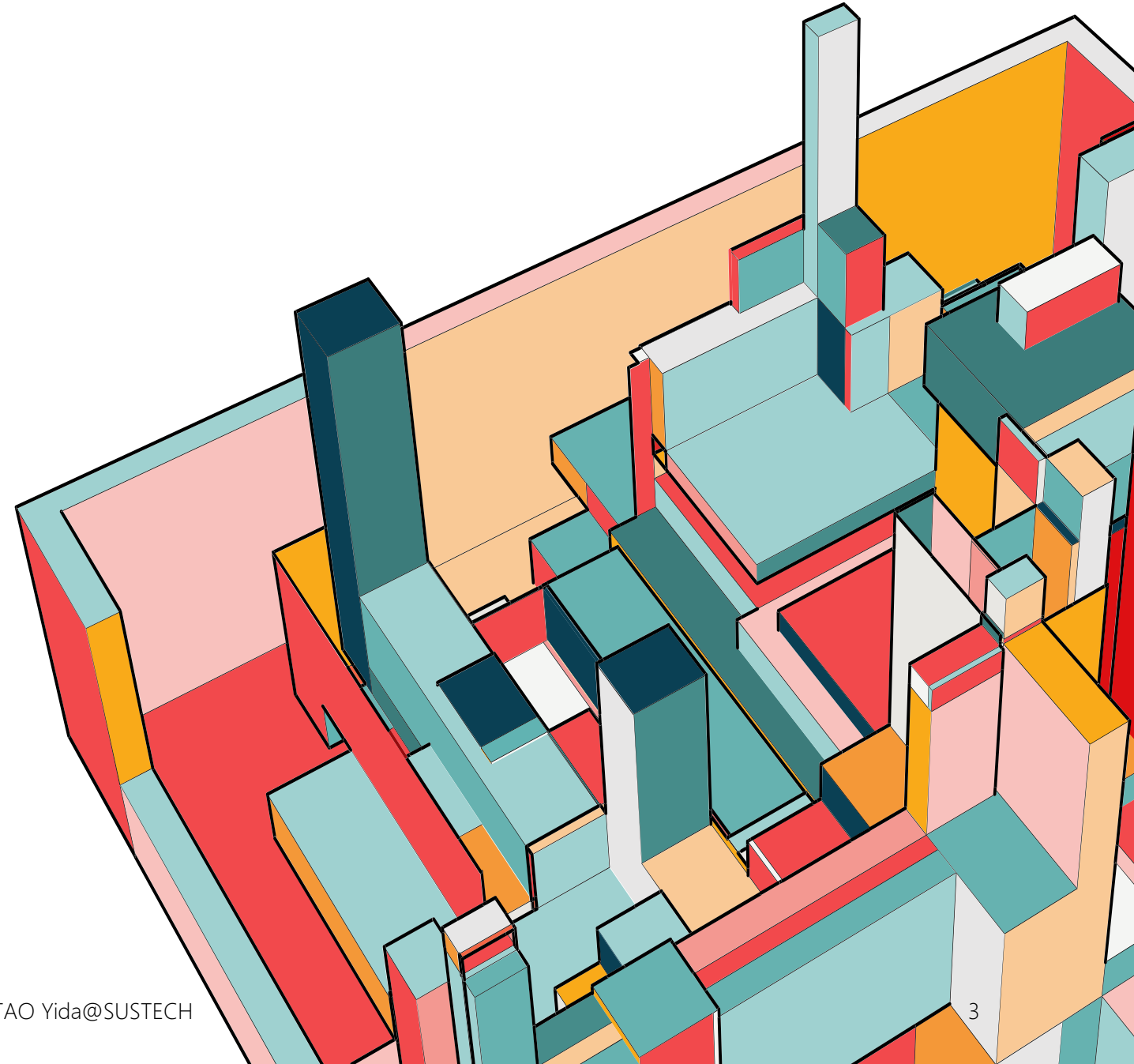
PREVIOUSLY

- **Programming assignment vs. Software**
- **Coding vs. Building a software**
- **Quality and efficiency**



LECTURE 2

- **Overview of software process**
- **Process models**
- **Agile & Scrum**
- **DevOps**



WHY SOFTWARE PROCESS?

- Writing code is easy

- Engineering good software system is HARD
  - Different roles are involved (customers, developers, managers, etc.)
  - Many aspects of complexity (time, scale, communication, etc.)
  - Trade-offs & making good decisions

WHY SOFTWARE PROCESS?

- Organizations want a well-defined, well-understood, repeatable software process (软件过程)

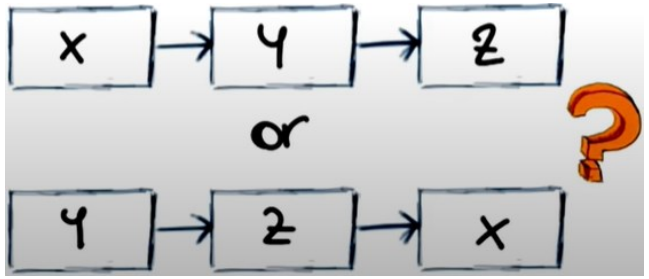
- Benefits
  - New team members know what to do
  - Know what's been done
  - Estimate time to completion, costs, etc.
  - Follow and repeat good practices

7 PHASES IN SOFTWARE PROCESS

Software process is a set of related phases/activities that take place in certain order, and leads to the production of a software product. All processes have some form of the following phases:

How do we put these phases together to build a software?

SOFTWARE PROCESS MODELS



- Software process models (or software lifecycle model) determines the order of these phases and criteria for transition of each phase

CHARACTERIZING SOFTWARE PROCESS MODELS

Broadly speaking, process models fall on a **continuum**

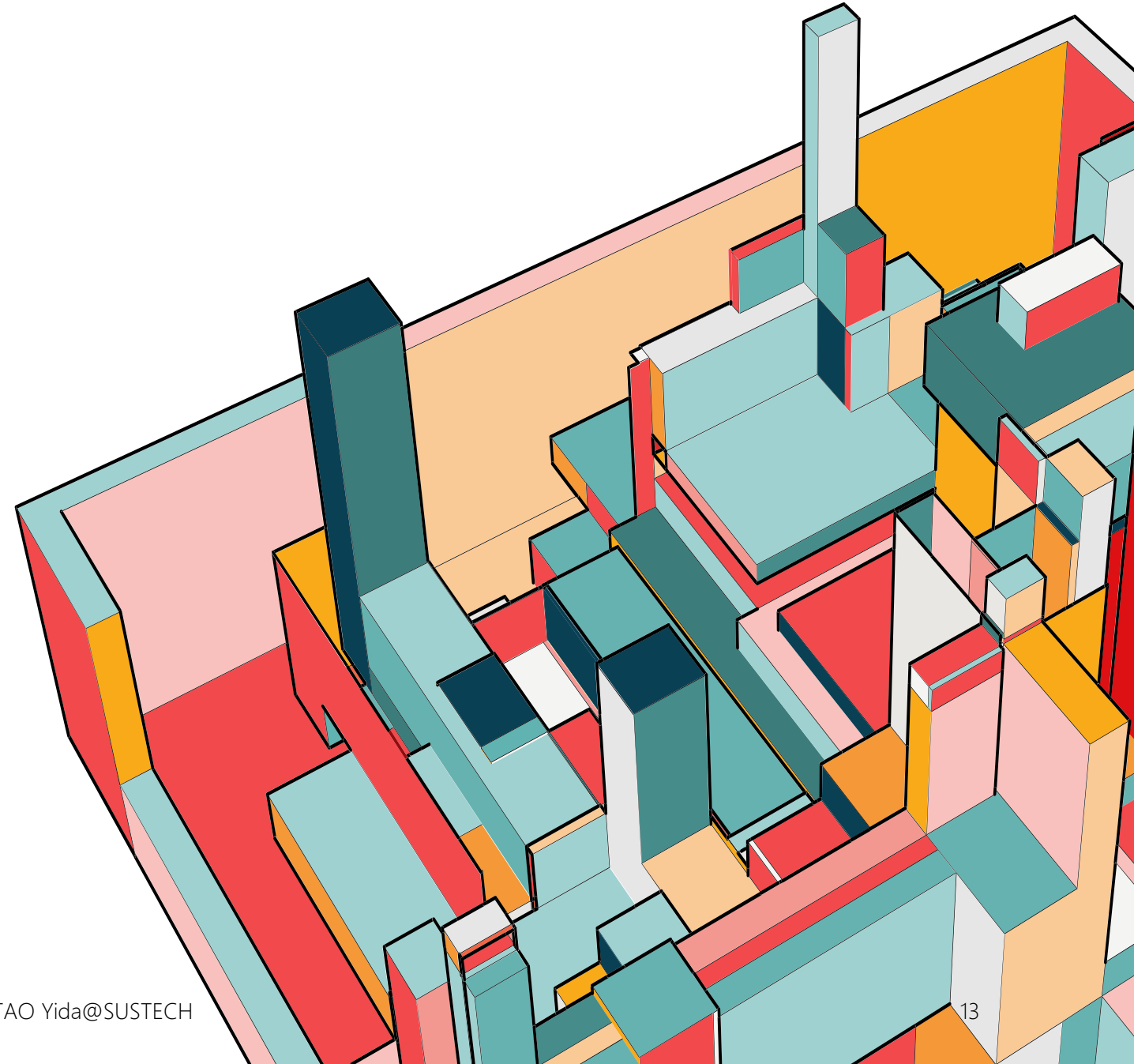
- **Plan-driven processes:** processes where all of the process activities are planned in advance and progress is measured against this plan.
- **Agile processes:** planning is incremental and it is easier to change the process to reflect changing customer requirements.

Agile — Plan-driven

There is no ideal process and most organizations have developed their own software development processes by finding a balance between plan-driven and agile processes.

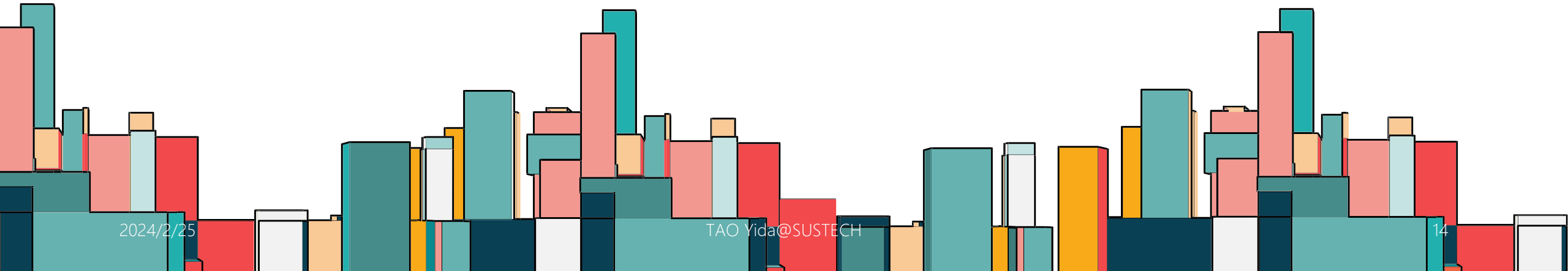
LECTURE 2

- **Overview of software process**
- **Process models**
- **Agile & Scrum**
- **DevOps**



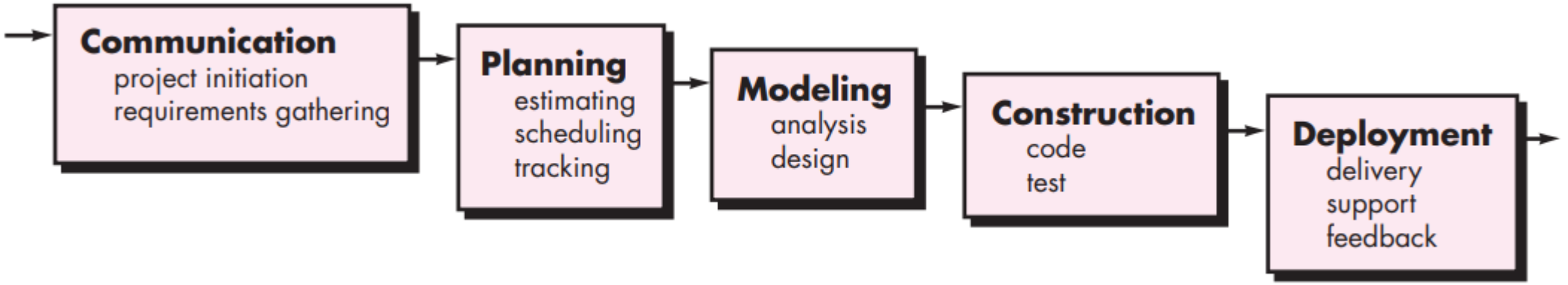
PROCESS MODELS

- Process models were originally proposed to bring order to the chaos of software development
- The general process models (or process paradigms) covered here could be extended or adapted in real practice
  - The Waterfall Model
  - Incremental Process Models
  - Evolutionary Process Models



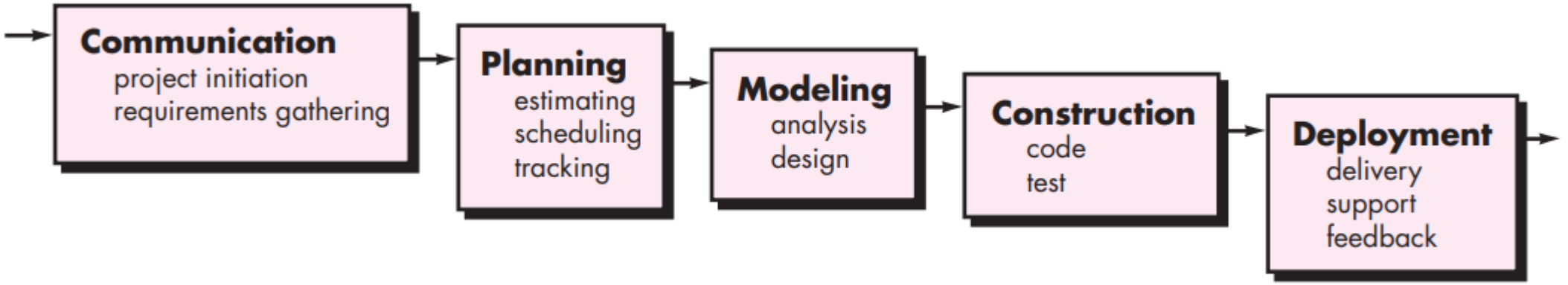
THE WATERFALL MODEL

The waterfall model (classic life cycle), suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through design, implementation, testing, deployment and maintenance



THE WATERFALL MODEL

- In principle, the result of each phase is one or more documents that are approved
- A phase should not start until the previous phase has finished.



PROBLEMS W/ THE WATERFALL MODEL

Sequential Flow

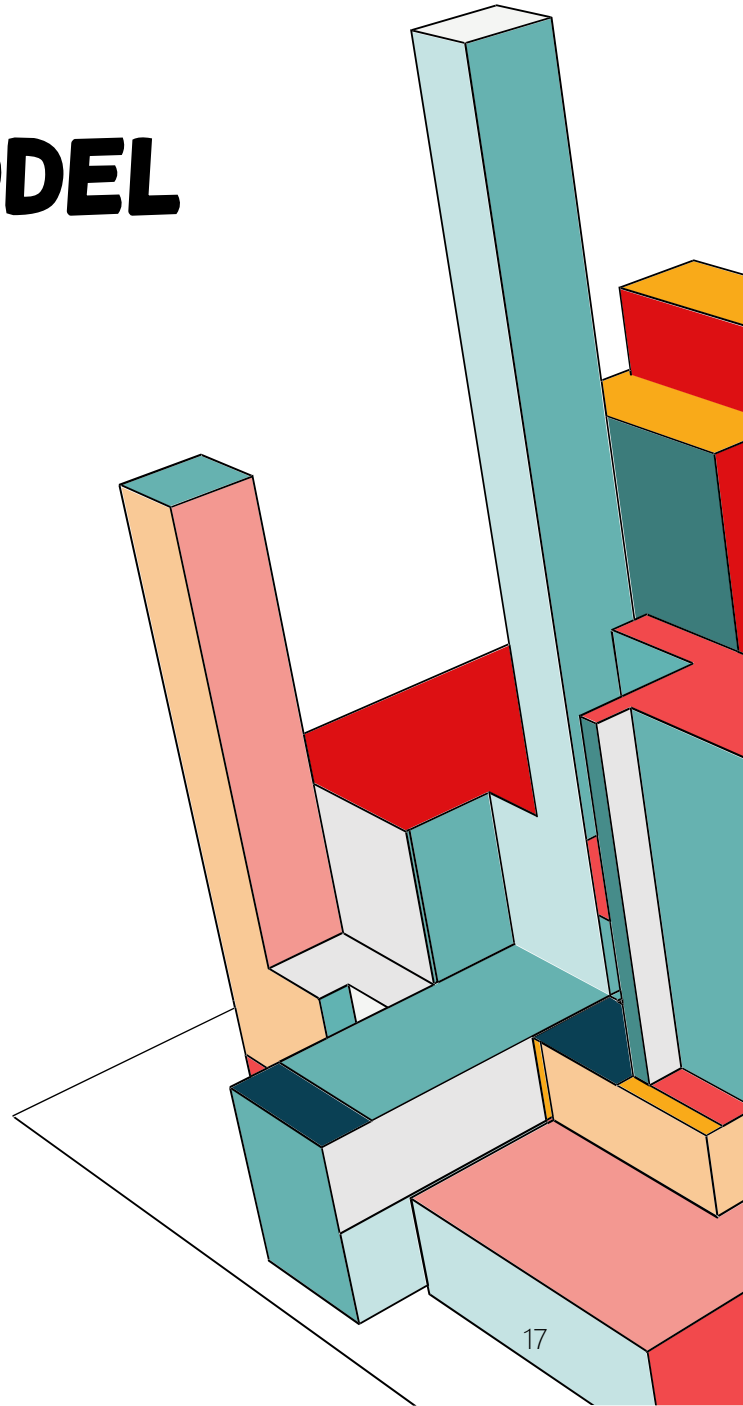
Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.

Clear Requirements

It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

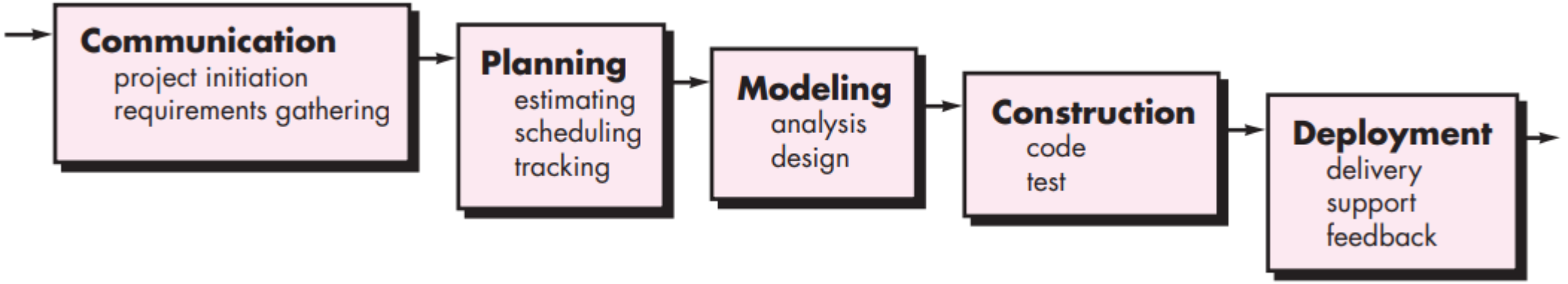
Customer Patience

A working version of the program(s) will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous



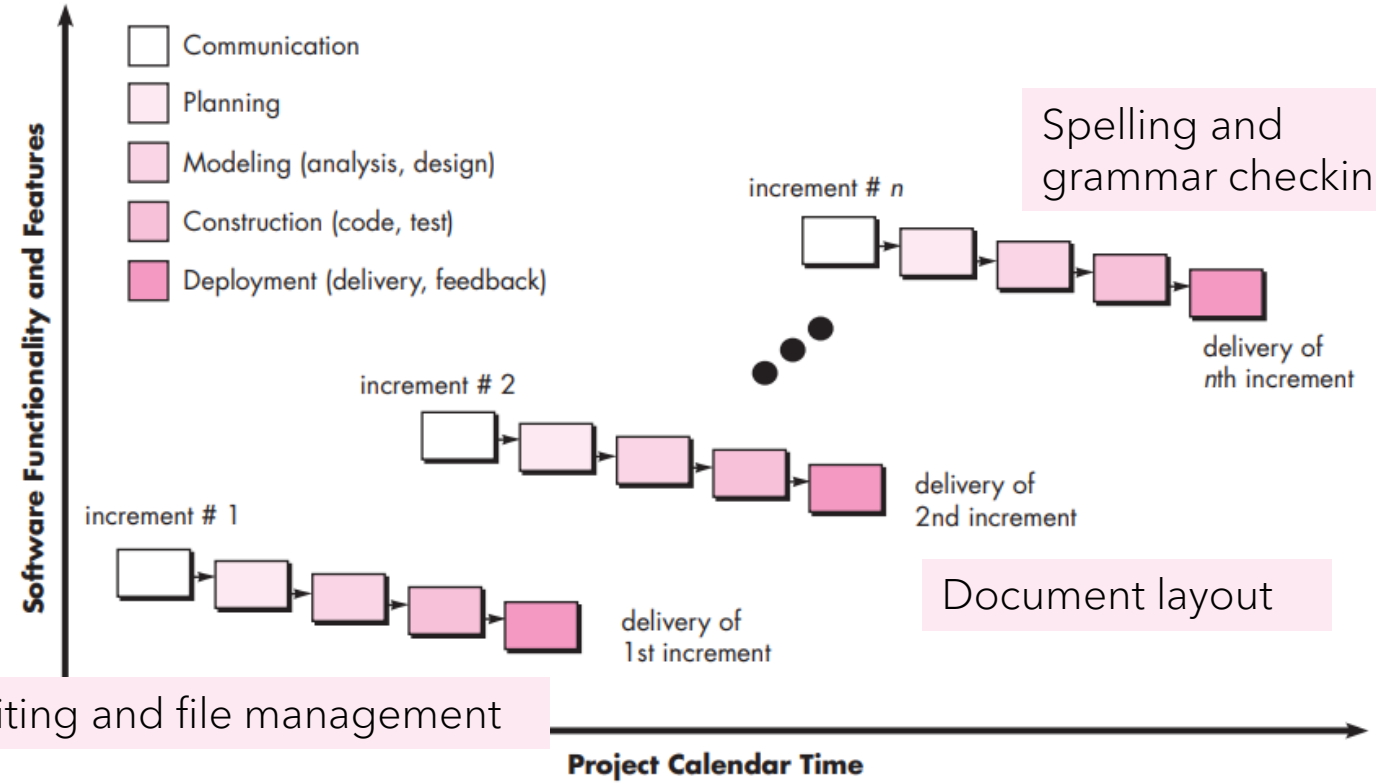
THE WATERFALL MODEL

The waterfall model might be adapted only when requirements are well defined and reasonably stable.



INCREMENTAL PROCESS MODELS

- The incremental model (增量模型) combines elements of linear and parallel process flows
- Each linear sequence produces deliverable "increments" of the software
- The first increment is often a core product with the basic requirements addressed. Supplementary features are developed in subsequent increments.



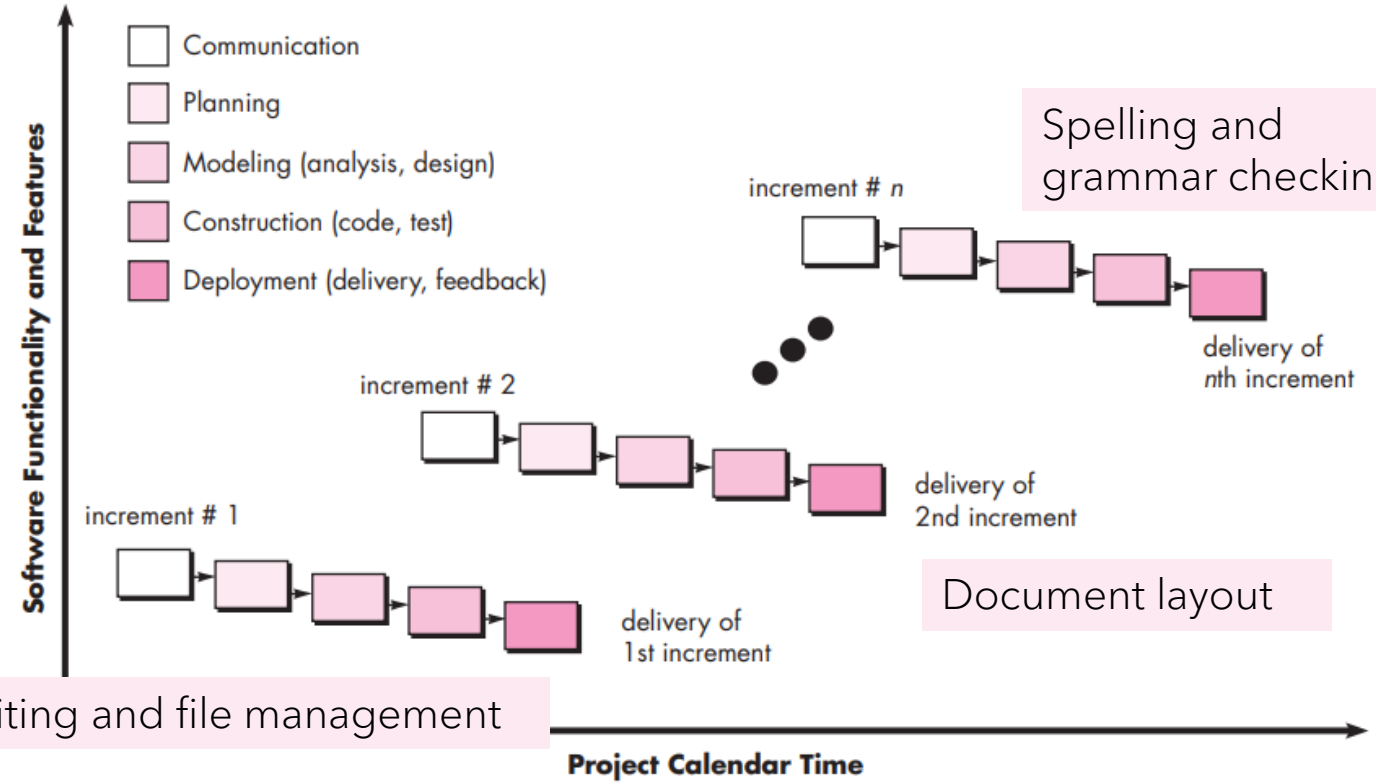
Spelling and grammar checking

Document layout

Basic editing and file management

INCREMENTAL PROCESS MODELS PROS

- Reducing cost due to requirement changes
- Easier to get customer feedback and adapt
- Easier to test and debug during a smaller iteration
- More rapid delivery and deployment of useful software
- Particularly useful when staffing is unavailable for a complete implementation



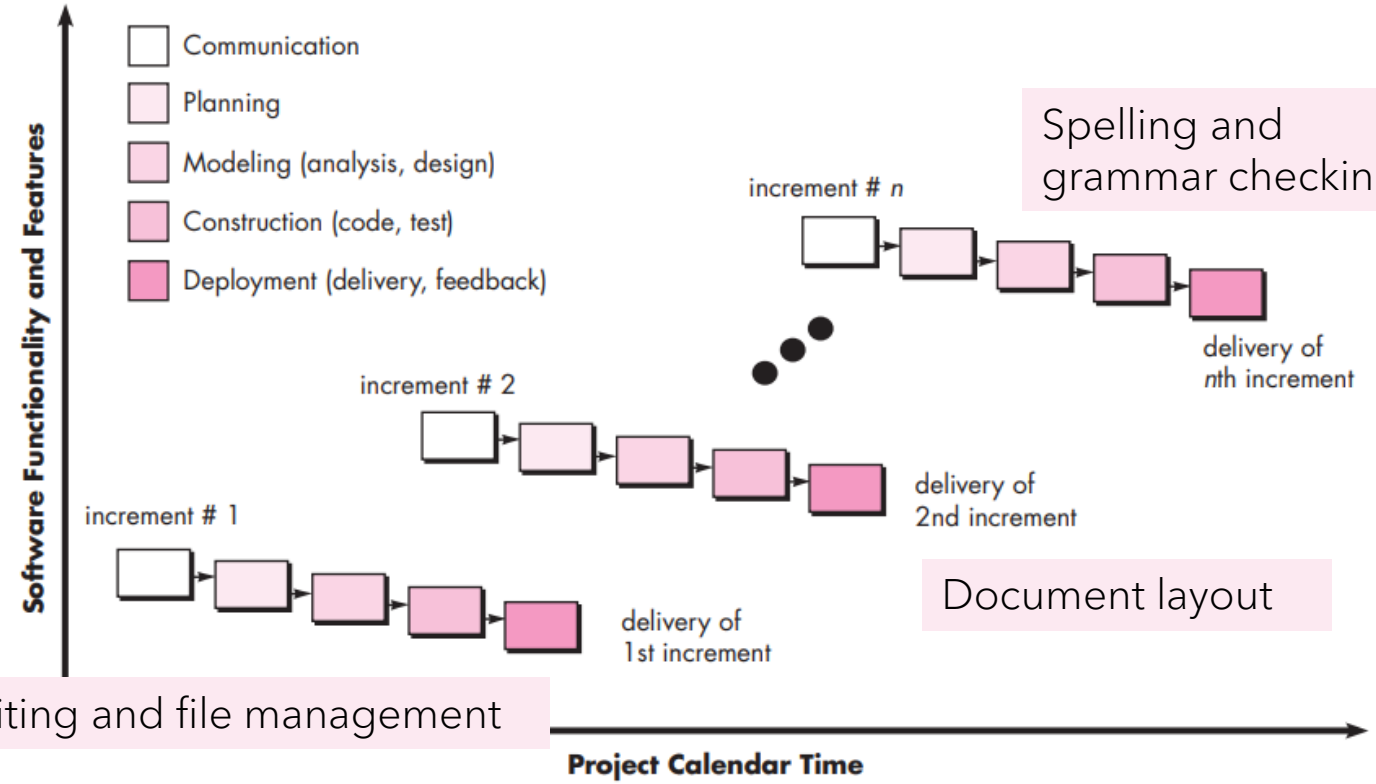
Spelling and grammar checking

Document layout

Basic editing and file management

INCREMENTAL PROCESS MODELS CONS

- Needs good planning and design of the whole system before breaking it down for incremental builds
- System structure tends to **degrade** as new increments are added. Unless effort is spent on **refactoring** to improve the software, regular change tends to corrupt its structure.
- Incorporating further software changes becomes increasingly difficult and costly.

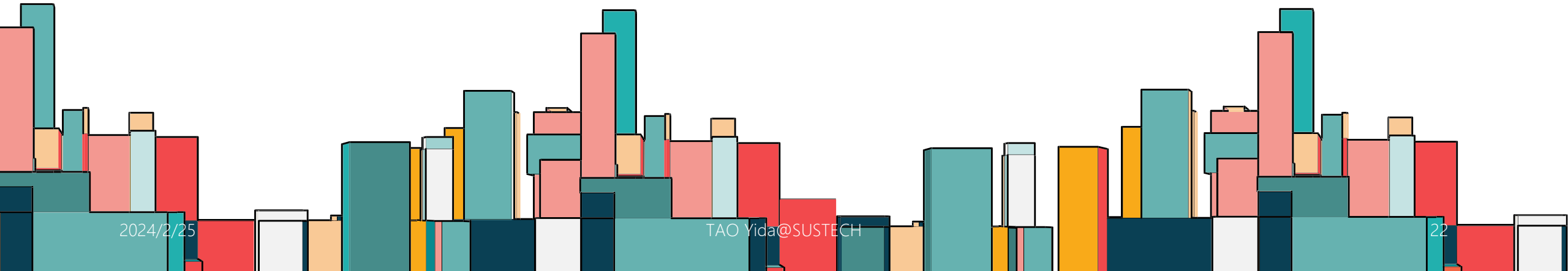


Spelling and grammar checking

Document layout

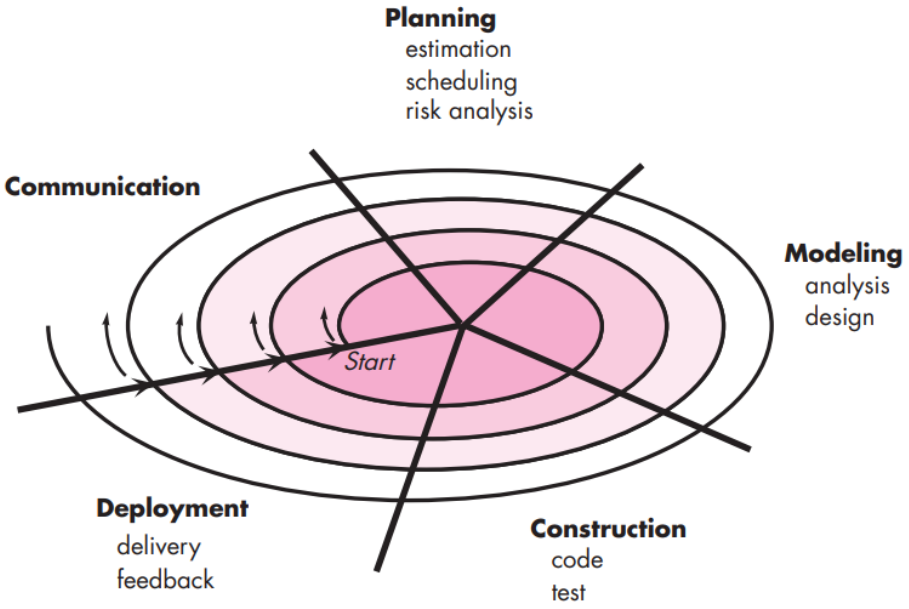
Basic editing and file management

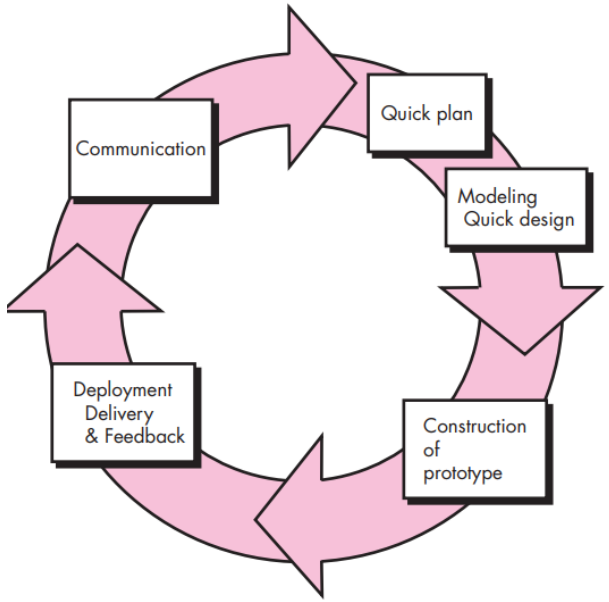
IS SOFTWARE DEVELOPMENT A LINEAR PROCESS?



EVOLUTIONARY PROCESS MODELS

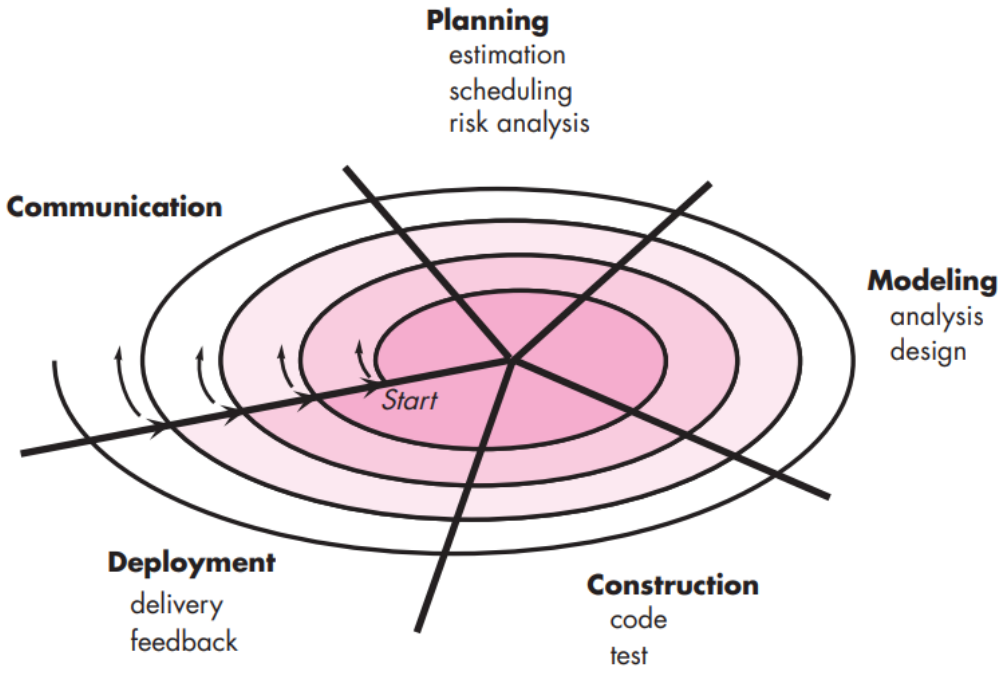
- Evolutionary models are iterative (迭代), which is explicitly designed to accommodate a product that evolves over time.
- A common evolutionary process model: the spiral model and prototyping





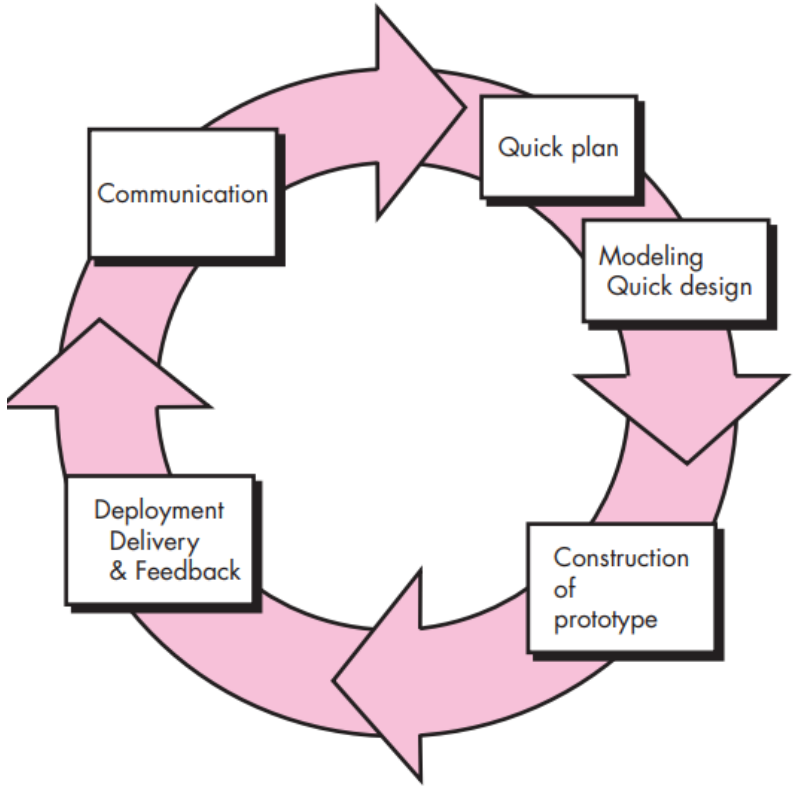
THE SPIRAL MODEL

- The spiral model couples the iterative nature with the controlled and systematic aspects of the waterfall mode
- Using the spiral model, software is developed in a series of evolutionary releases.
- During early iterations, the release might be a model or prototype.
- During later iterations, increasingly more complete versions of the engineered system are produced
- This model demands a direct consideration of technical risk at all stages of the project, effectively reducing potential risks



PROTOTYPING

- Prototyping (原型开发) might be used when customers define a set of general objectives for software, but do not identify detailed requirements for functions and features
- The prototype iteration begins with communication, followed by quick planning and modeling, which lead to the construction of a prototype that can be delivered and evaluated
- Customers could quickly see what appears to be a working version of the software and give feedback

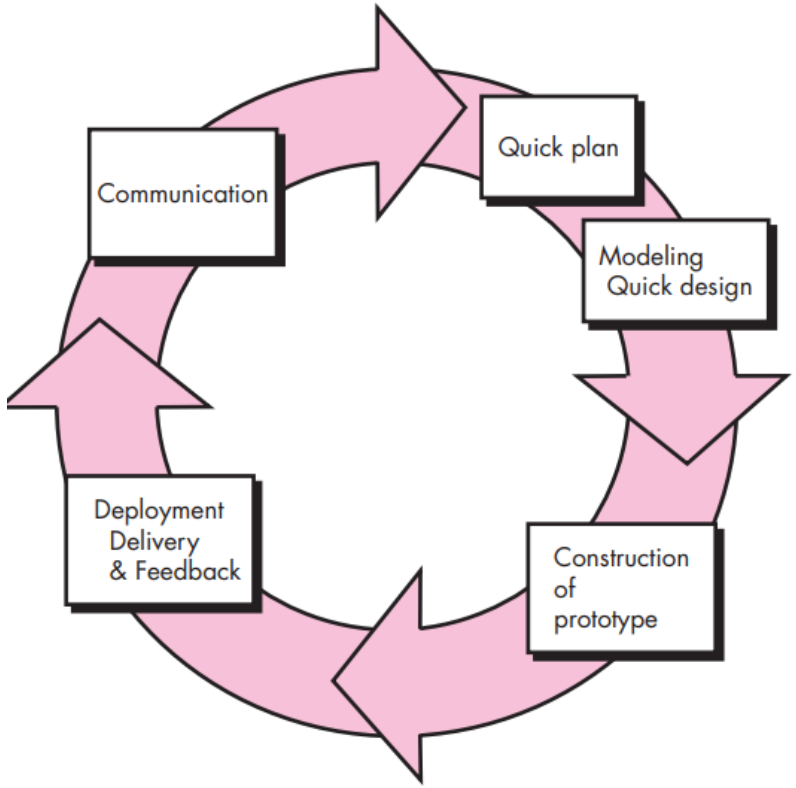


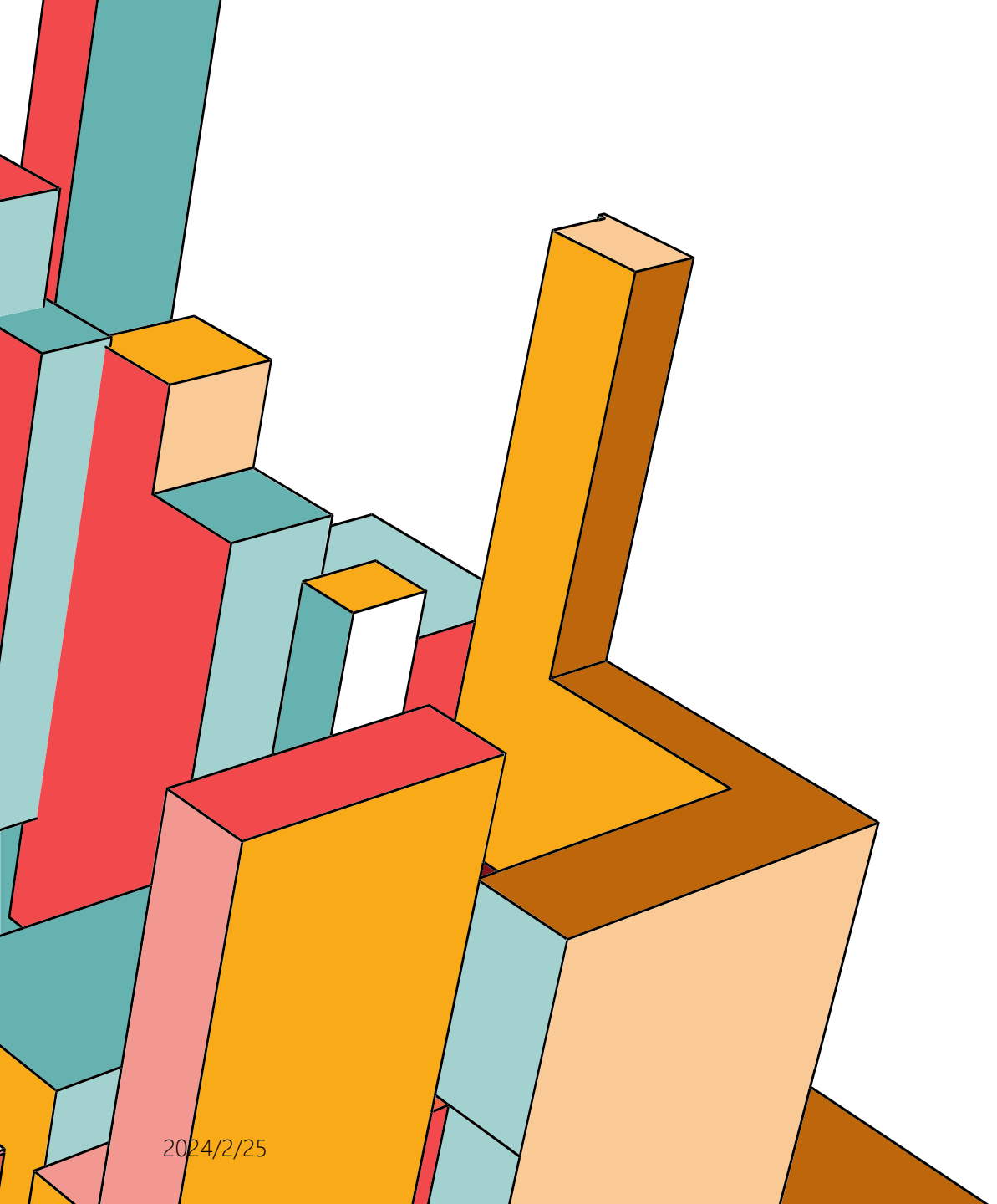
PROTOTYPING

- Developers often make implementation compromises, e.g., inappropriate OS, PL, and inefficient algorithms, in order to get a prototype working quickly

- The prototype is often discarded (at least in part), and the actual software is engineered with an eye toward quality.

✓ Prototyping should be used when the requirements are not clearly understood or are unstable
✓ Prototyping can also be used for developing UI or complex, high-tech systems in order to quickly demo the feasibility





ARGUMENTS ON PROCESS MODELS

Process models forget the frailties of the people who build computer software

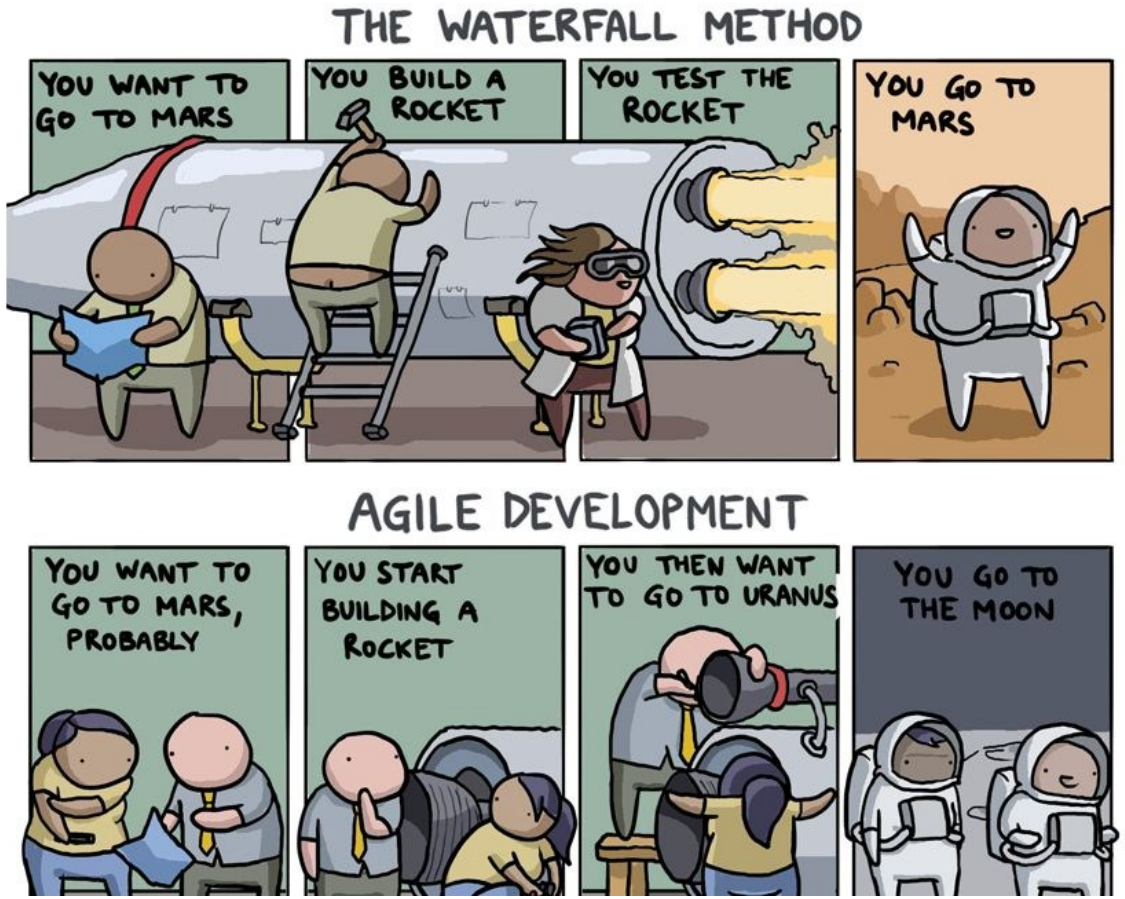
Software engineers are not robots. They exhibit great variation in working styles; significant differences in skill level, creativity, orderliness, consistency, and spontaneity. Some communicate well in written form, others do not

Process models often deal with people's common weaknesses with discipline

"Because consistency in action is a human weakness, high discipline methodologies are fragile." [Alistair Cockburn. 2002. Agile Software Development]

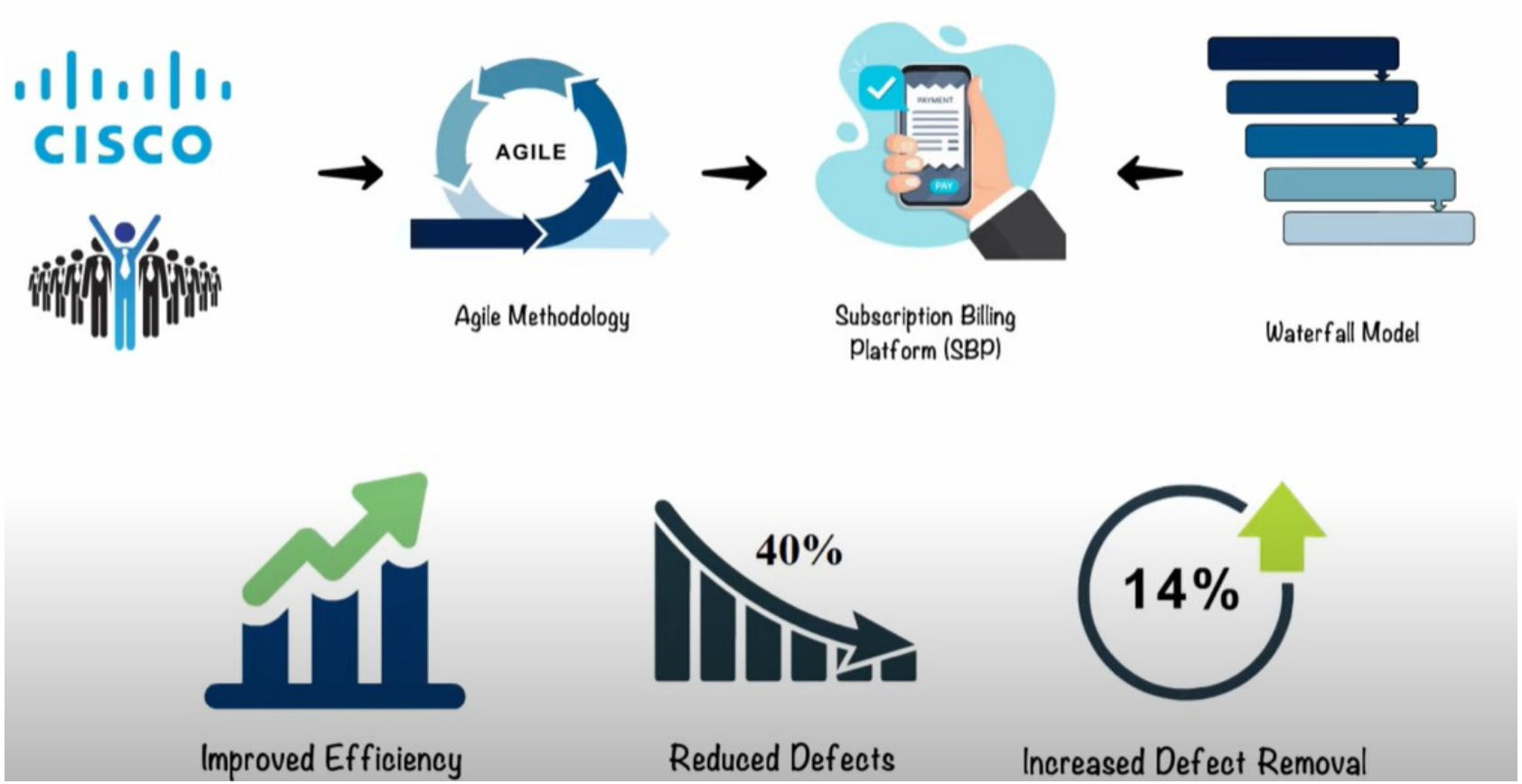
FRAGILE

AGILE VS. WATERFALL



https://www.pinterest.com/pin/678917712570197228/

AGILE VS. WATERFALL



https://www.youtube.com/watch?v=8eVXTyIZ1Hs&t=182s

WHAT IS AGILE, EXACTLY?

Agile (敏捷), in general, is the ability to create and respond to change.

WHAT IS AGILE, EXACTLY?

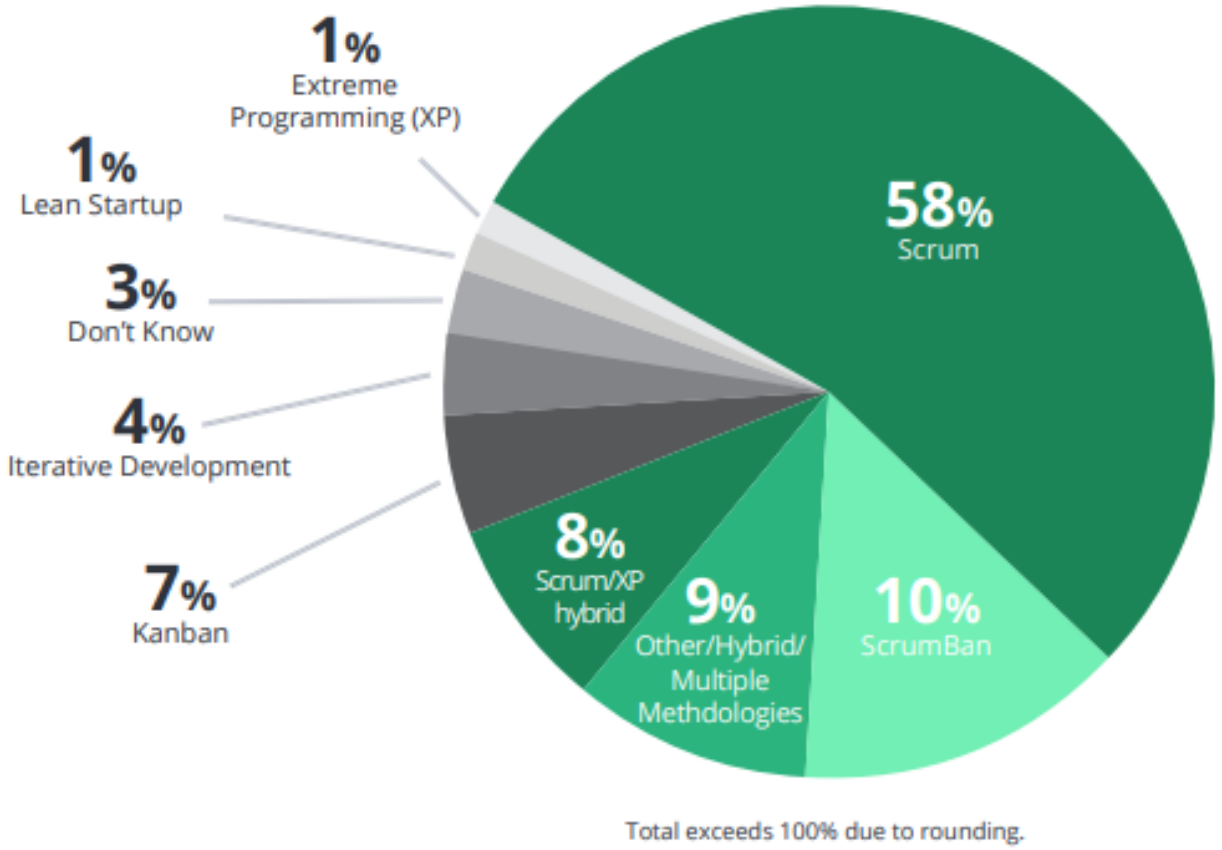
- Agile is an **iterative** approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches
- An agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

AGILE VALUES

In 2001, Kent Beck and 16 other noted software developers, writers, and consultants signed the "Manifesto for Agile Software Development." (敏捷软件开发宣言)

Agile, in the context of software engineering, refers to the methods and best practices for organizing projects based on the values and principles documented in the Agile Manifesto.

AGILE METHODOLOGY



Total exceeds 100% due to rounding.

Source: 14th Annual State of Agile Report

- However, there is NO one right way to implement Agile
- There are many different types of agile methodologies

THE ORIGIN OF SCRUM

- The term “scrum” was borrowed from the game of rugby, to stress the importance of teamwork to deal with complex problems
- Process of sport games = Process of software development

SCRUM ROLES

Product Owner

Define and adjust product features; Accept or reject work results

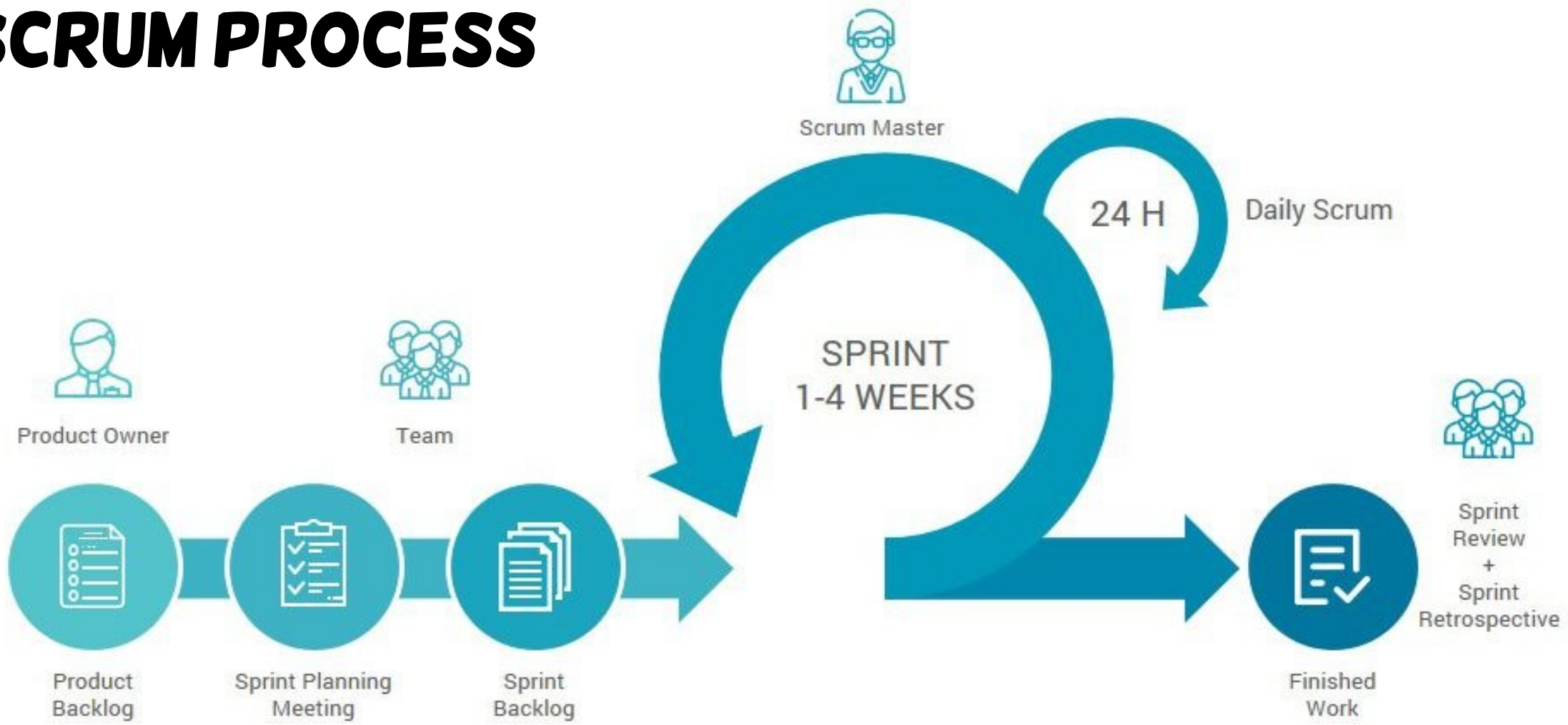
Scrum Master

Coach the team and enact Scrum values and principles

Scrum Team

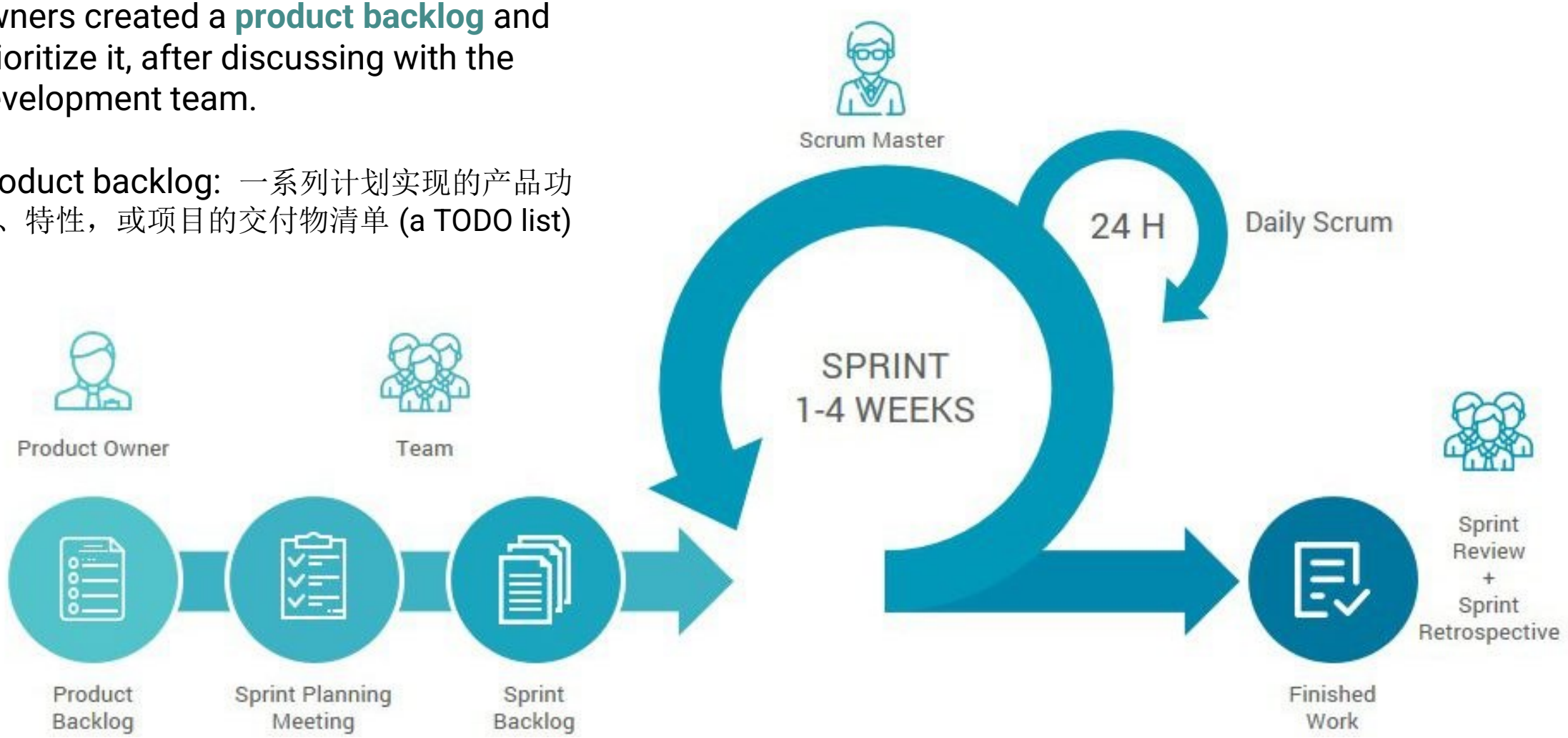
Cross-functional members like developers, testers, designers (Typically 5-9 size)

SCRUM PROCESS



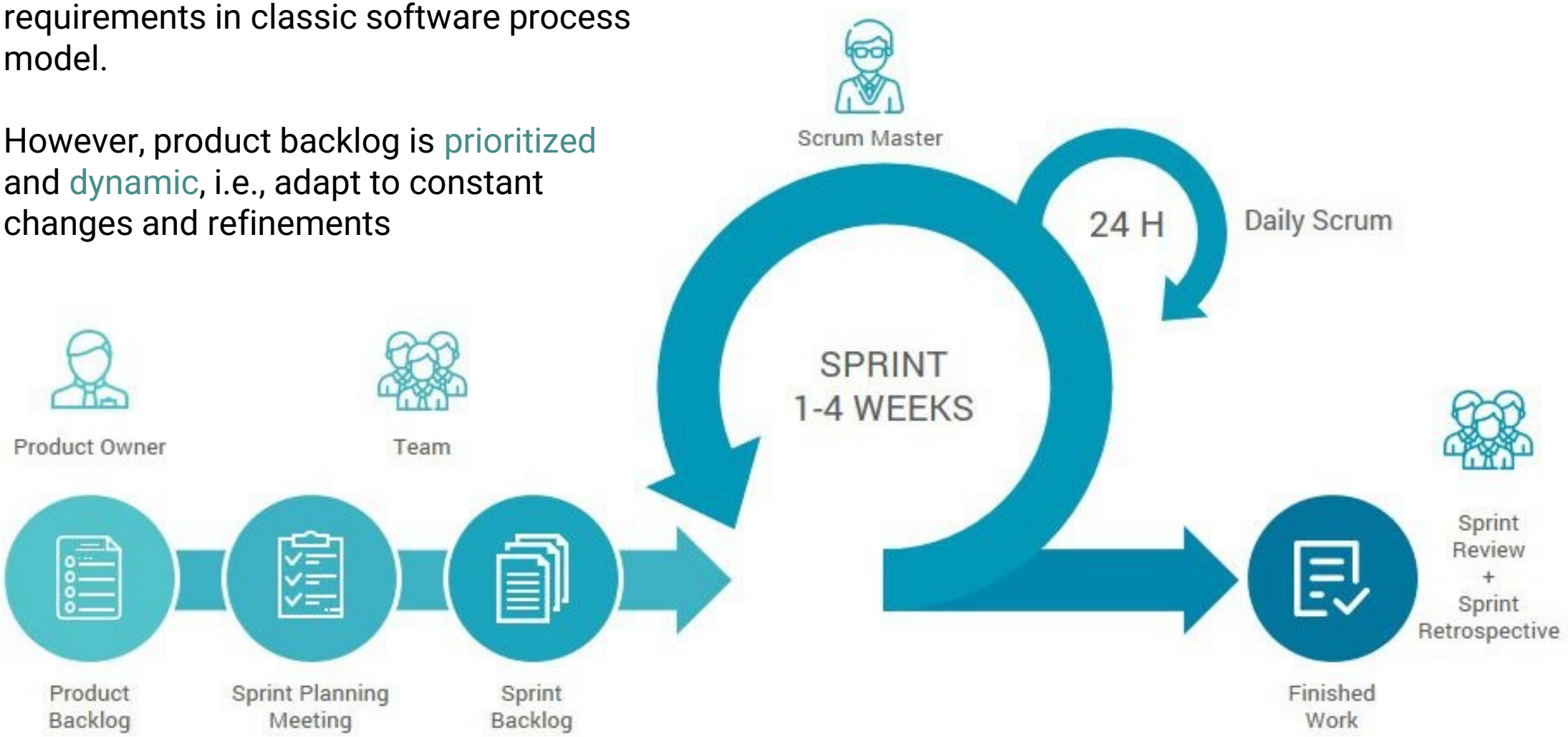
The scrum process starts with product owners created a **product backlog** and prioritize it, after discussing with the development team.

Product backlog: 一系列计划实现的产品功能、特性，或项目的交付物清单 (a TODO list)



Product backlog is similar to requirements in classic software process model.

However, product backlog is prioritized and dynamic, i.e., adapt to constant changes and refinements



EXAMPLE OF PRODUCT BACKLOG

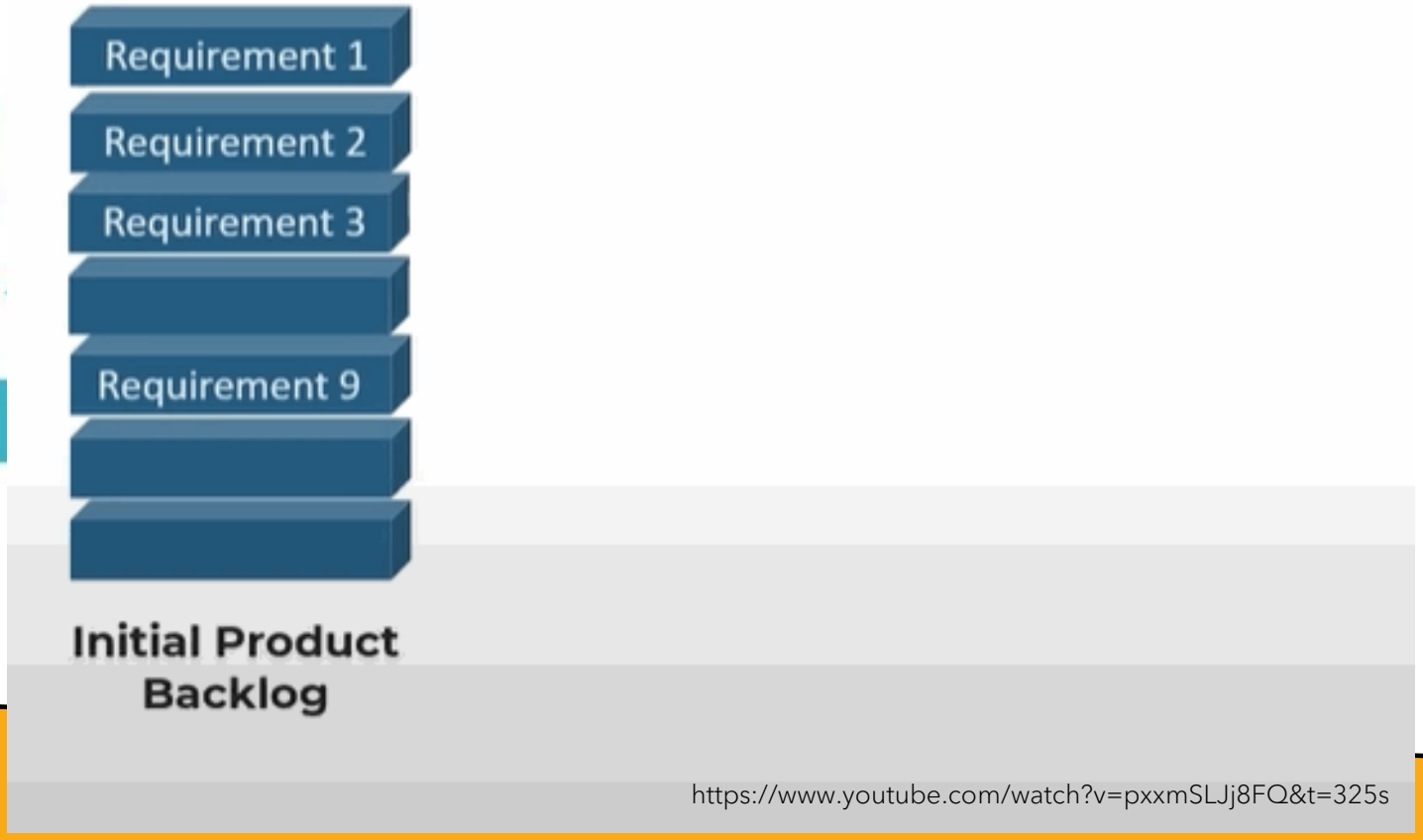
More details on week 3

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Next, the product owner and the development team attend the Sprint Planning Meeting to select the items to deliver in the next sprint.

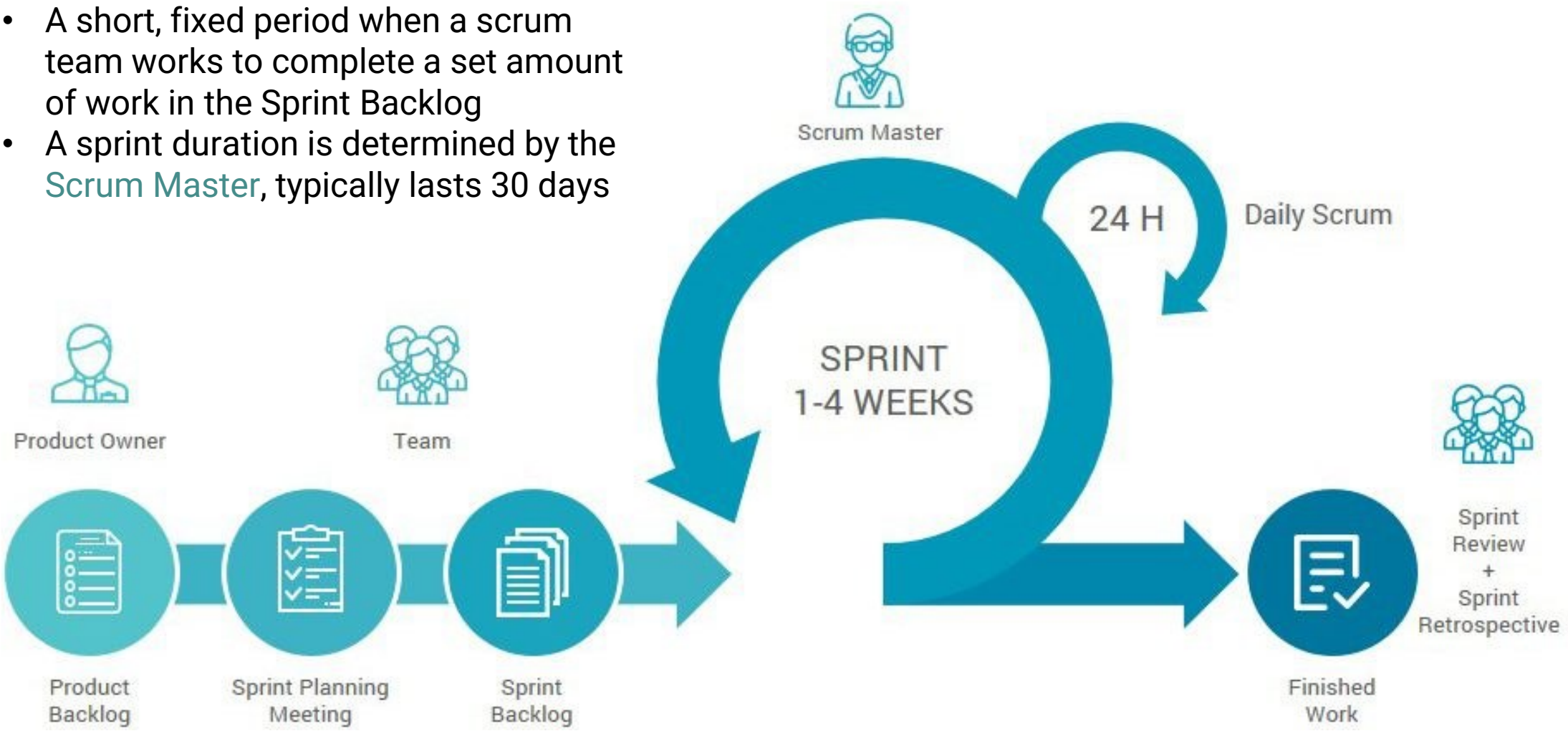
Sprint (冲刺):Scrum项目管理方法中的一个常规、可重复的较短工作周期。在这个周期里，项目团队需快速完成预定的工作量（i.e., sprint backlog）



https://www.youtube.com/watch?v=pxxmSLJj8FQ&t=325s

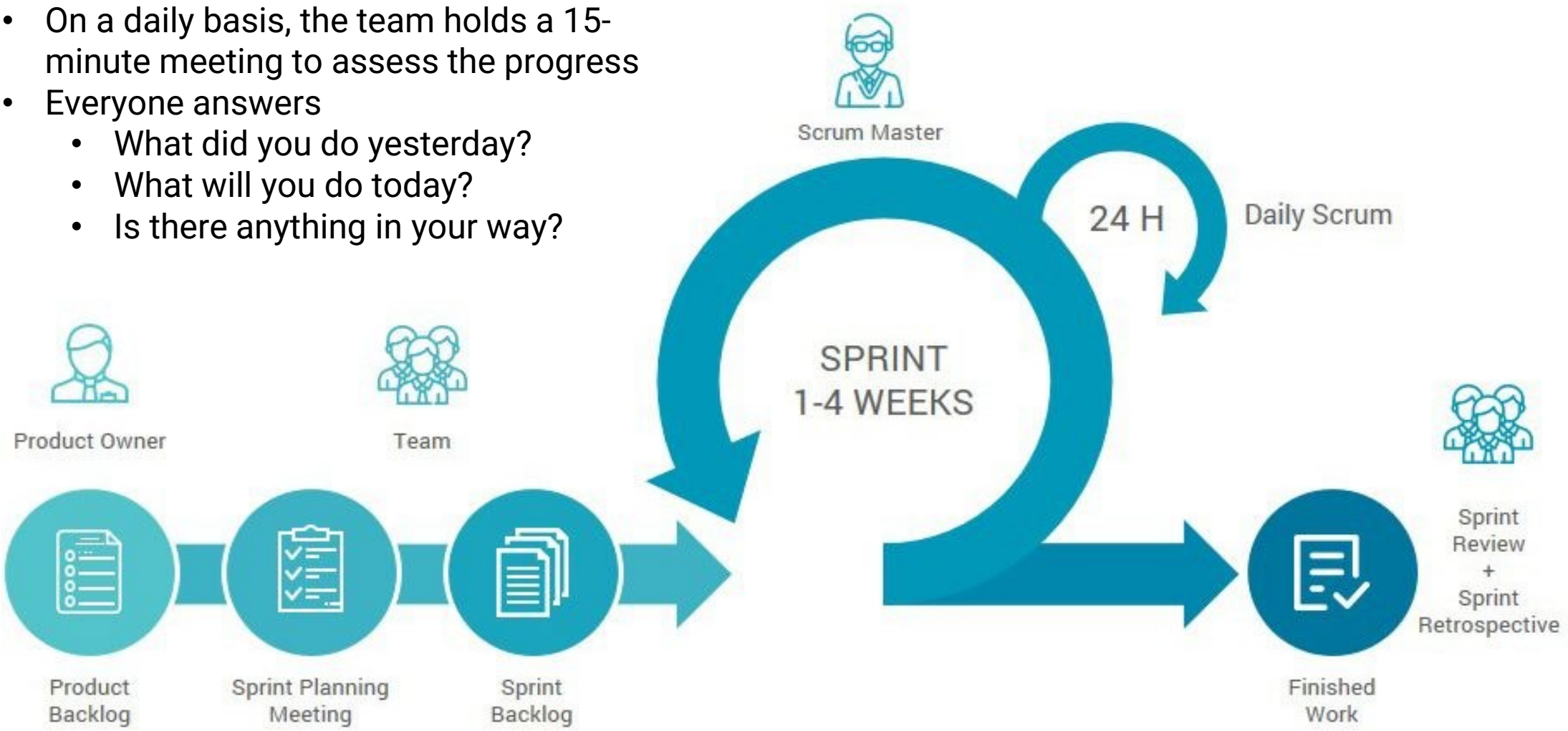
Sprint

- A short, fixed period when a scrum team works to complete a set amount of work in the Sprint Backlog
- A sprint duration is determined by the Scrum Master, typically lasts 30 days



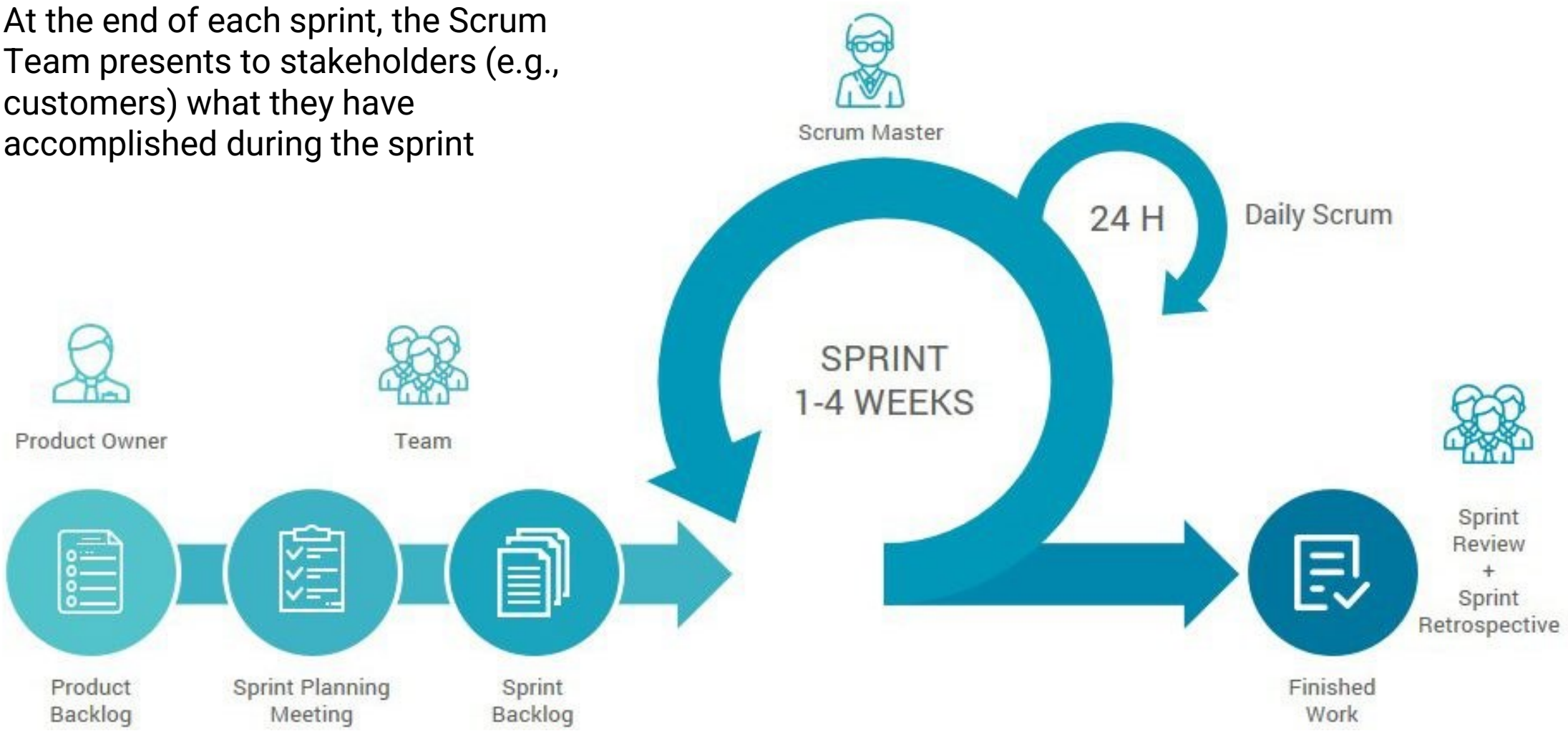
Daily Scrum (例会)

- On a daily basis, the team holds a 15-minute meeting to assess the progress
- Everyone answers
  - What did you do yesterday?
  - What will you do today?
  - Is there anything in your way?



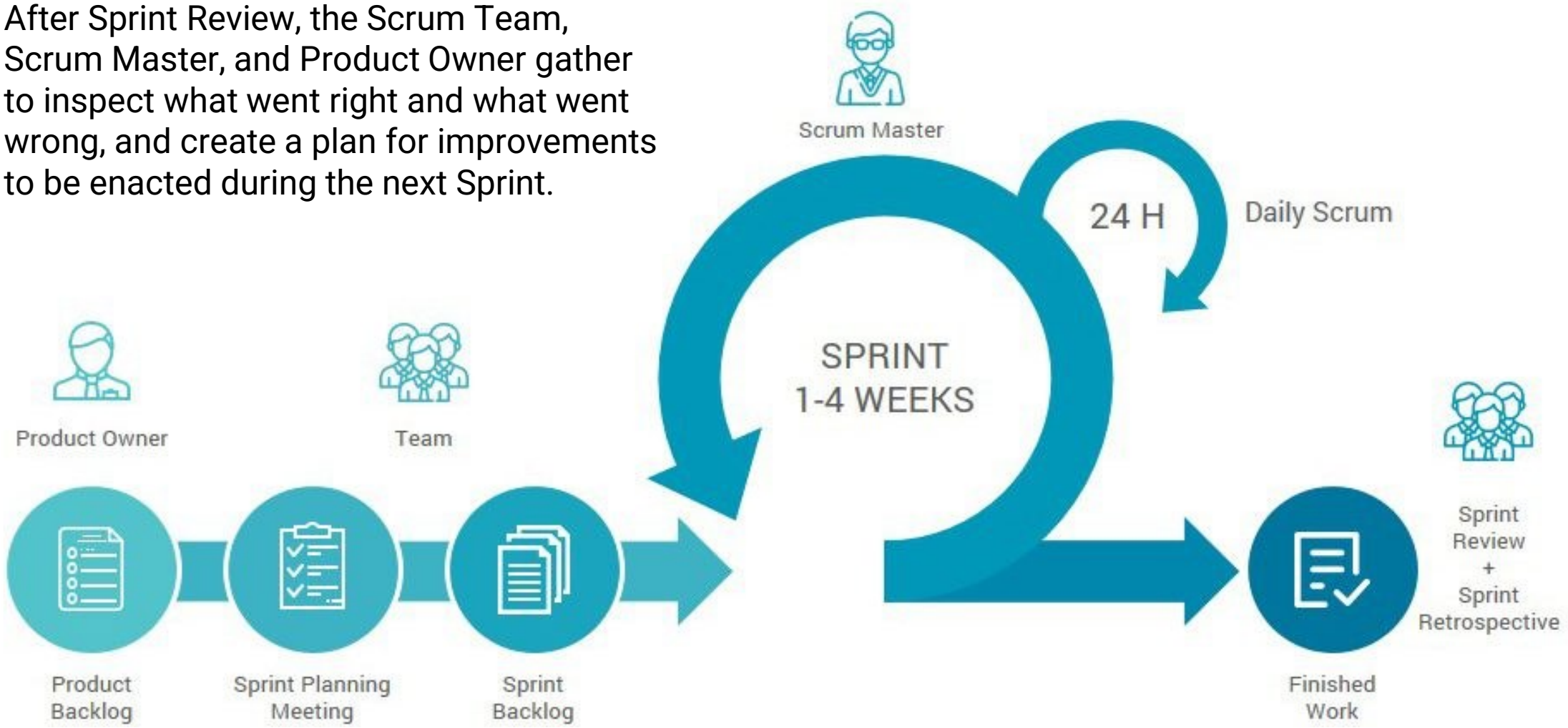
Sprint Review（评审）

At the end of each sprint, the Scrum Team presents to stakeholders (e.g., customers) what they have accomplished during the sprint



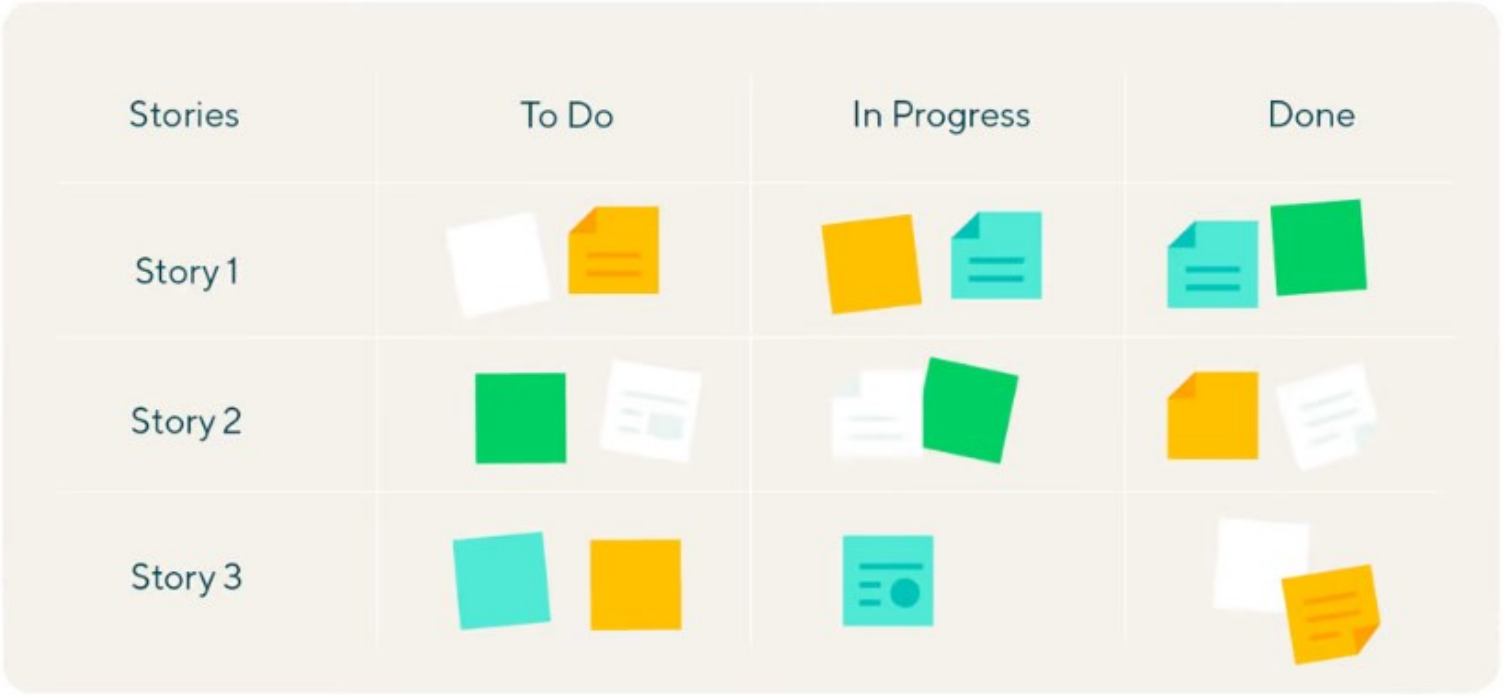
Sprint Retrospective（回顾）

After Sprint Review, the Scrum Team, Scrum Master, and Product Owner gather to inspect what went right and what went wrong, and create a plan for improvements to be enacted during the next Sprint.

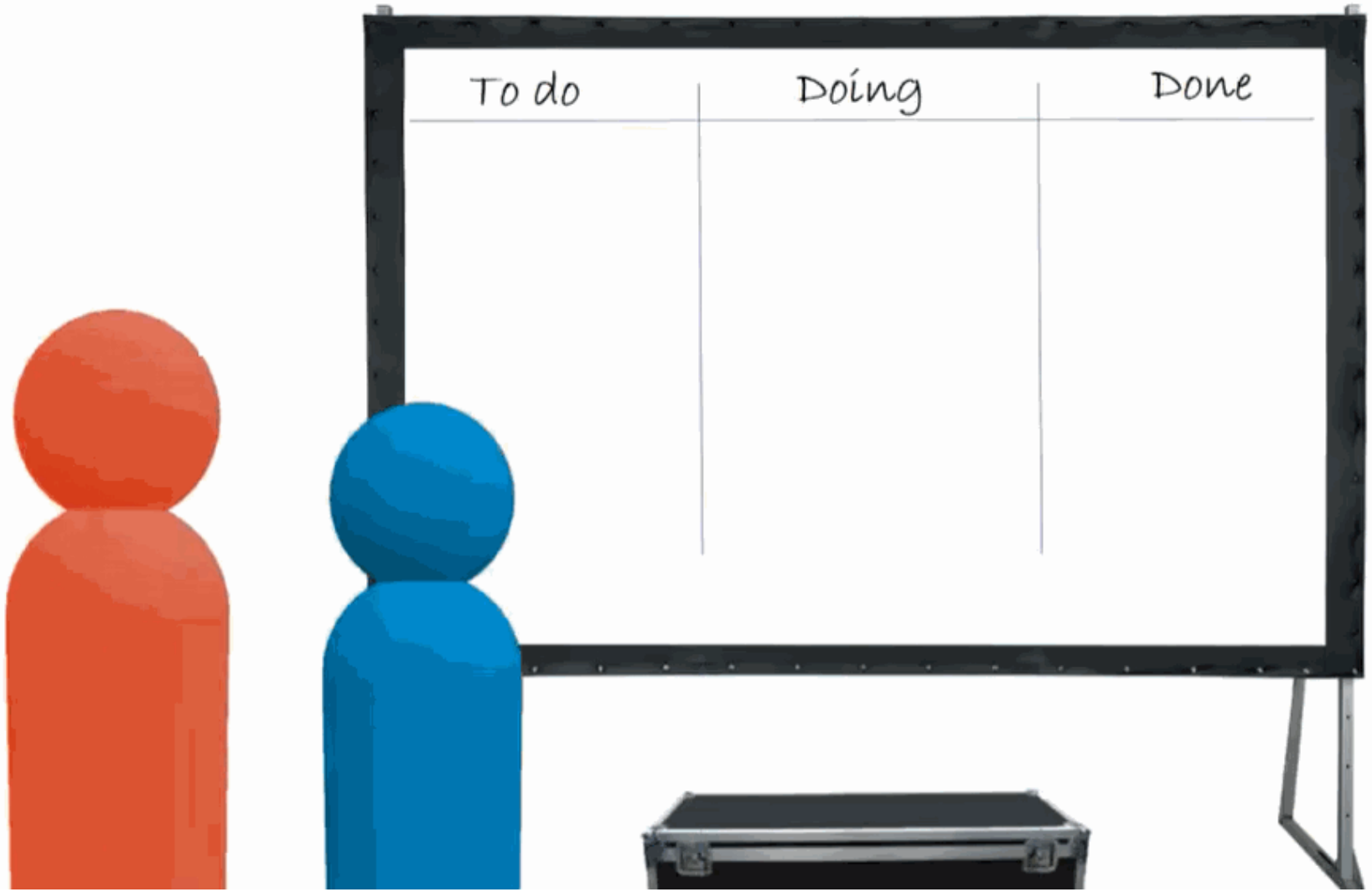


SCRUM BOARDS

- A Scrum Board is a tool that helps Scrum Teams make Sprint Backlog items visible.
- The board is traditionally divided up into three categories: To Do, Work in Progress and Done.

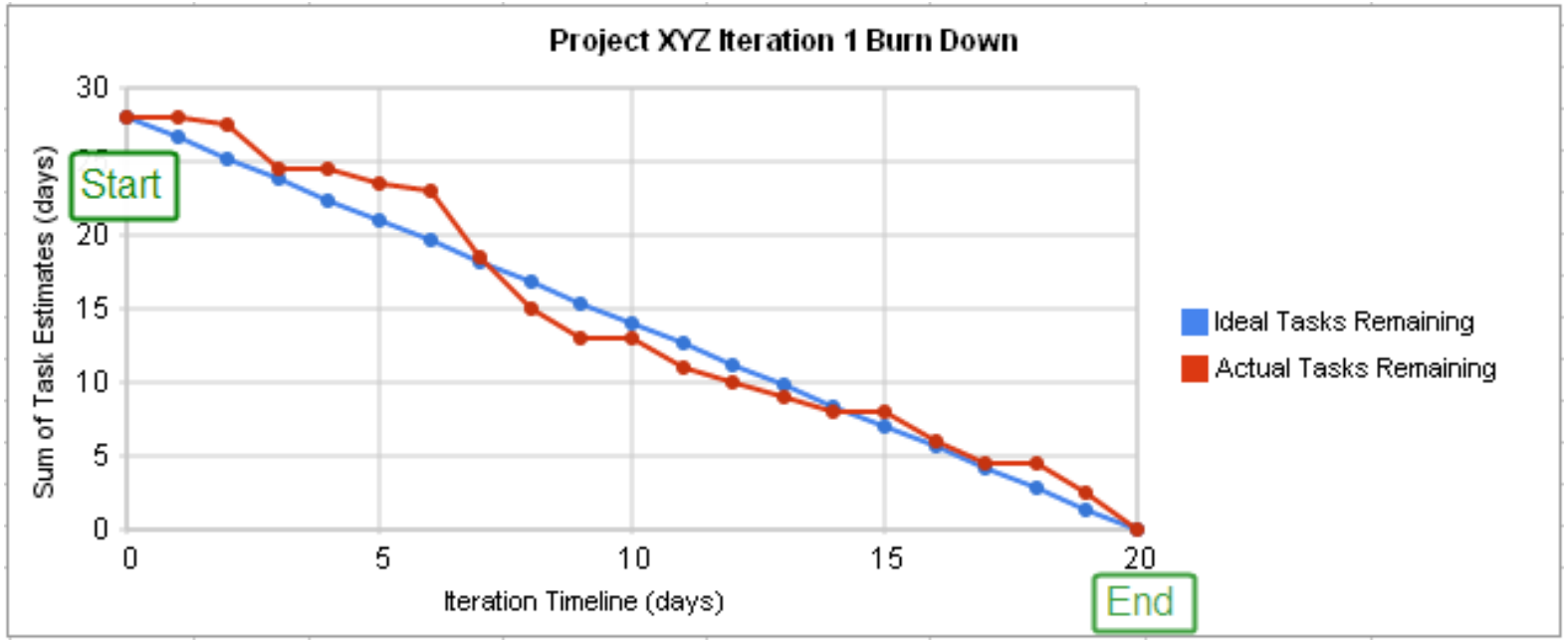


The scrum master



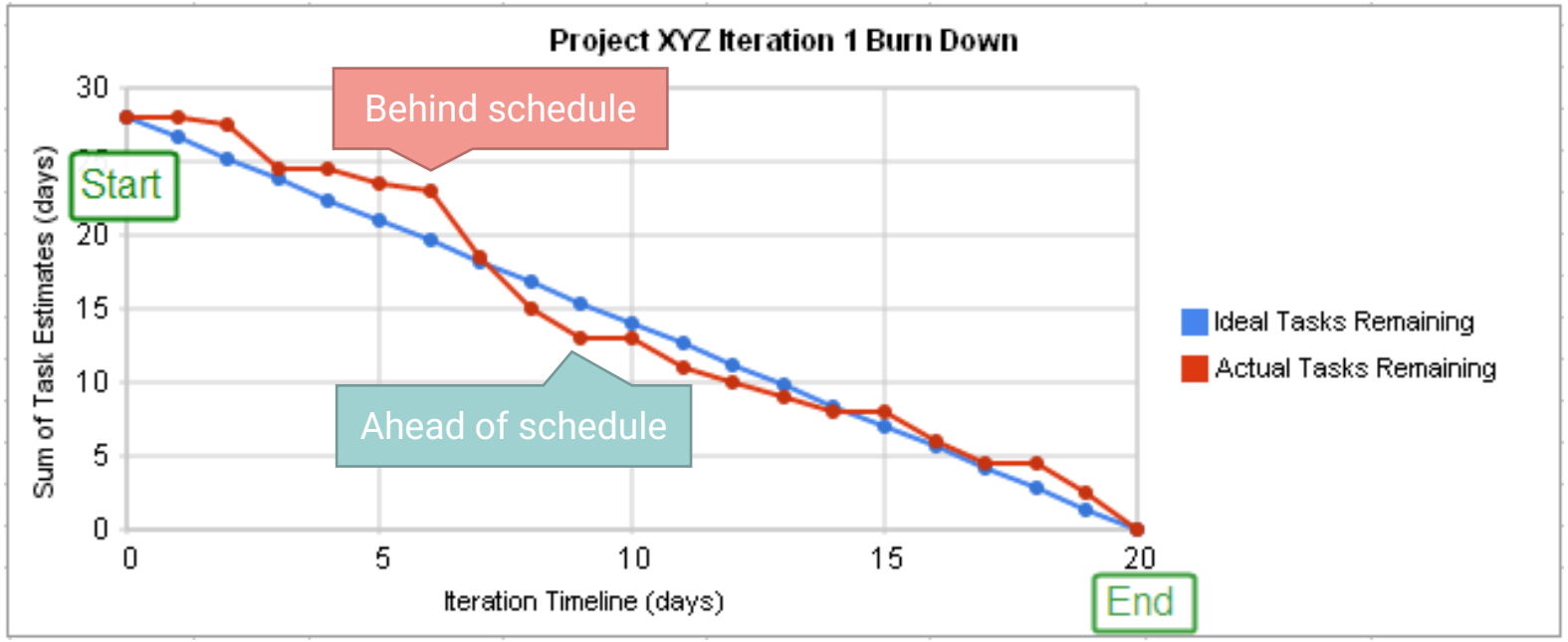
BURNDOWN CHARTS

- A graphical representation of work left to do versus time.
- The backlog or effort remaining is often on the Y-axis, with time in the sprint along the X-axis
- Useful for predicting when all of the work will be completed



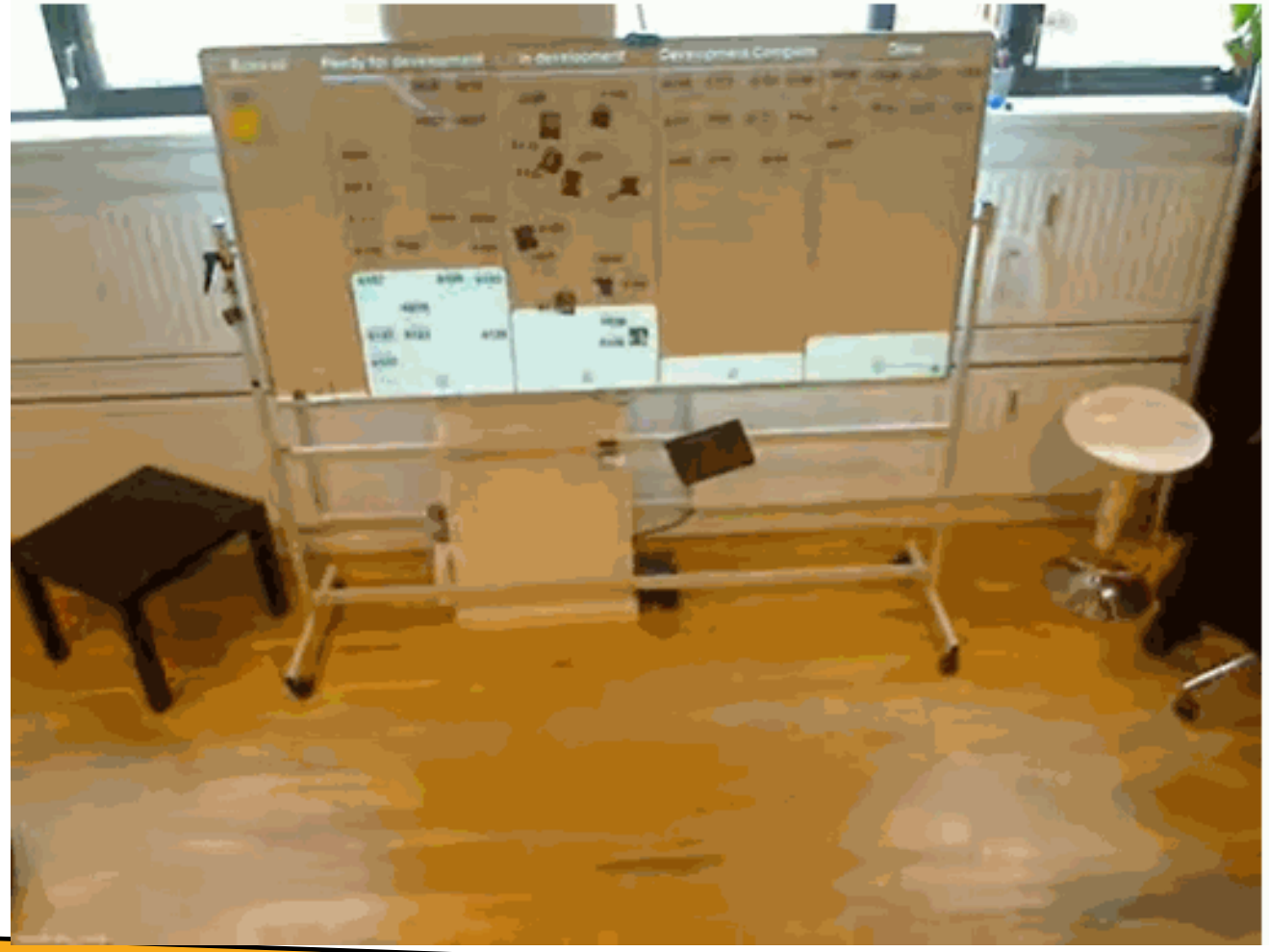
BURNDOWN CHARTS

- The top left corner is your starting point, and the successful project end is the bottom right.
- A straight line from start to finish represents your ideal work remaining
- A second line represents the actual work remaining



SCRUM BOARD AND BURNDOWN CHARTS IN ACTION

https://www.scruminc.com/sprint-burndown-chart/



SCRUM SUMMARY

Roles	Ceremonies	Artifacts
Product Owner	Sprint Planning	Product Backlog
Scrum Master	Daily Scrum	Sprint Backlog
Scrum Team	Sprint Review	Burndown Charts
	Sprint Retrospective	

SCRUM CASE STUDY

Example 1 : The adoption of Scrum by Google's marketing team led to a 30% improvement in campaign effectiveness and a 20% improvement in teamwork.

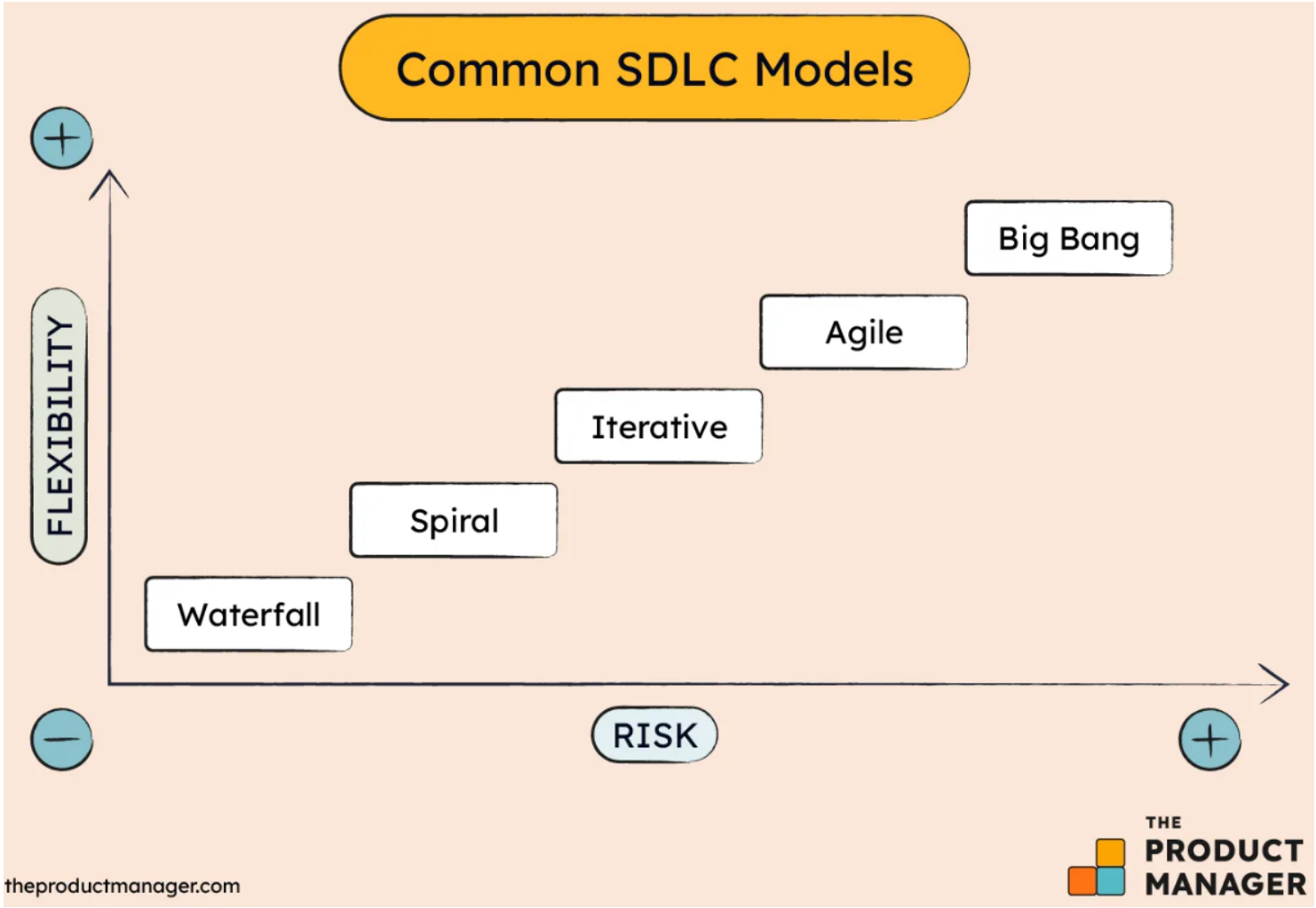
Example 2: Scrum was adopted by Tesla for their autonomous driving project, which led to a 15% acceleration of the development cycle and a 10% decrease in errors.

Example 3: A well-known hospital adopted Scrum for its project to optimize patient care, which resulted in a 25% decrease in patient wait times and a 15% boost in staff satisfaction.

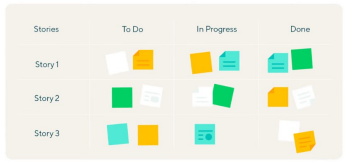
https://agilewaters.com/examples-of-the-best-scrum-case-studies-in-real-life-2023/

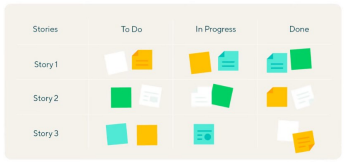
SOFTWARE PROCESS SUMMARY

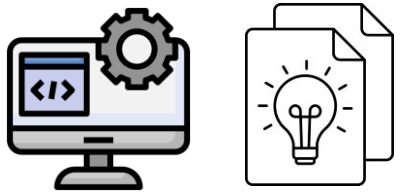
Big Bang model:
Developers jump right into coding without much planning, without any kind of clear roadmap

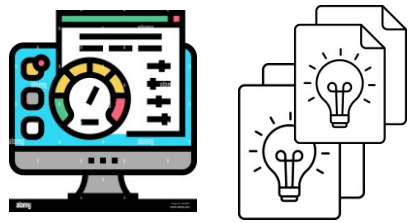


OUR TEAM PROJECT









Week 1
Team up

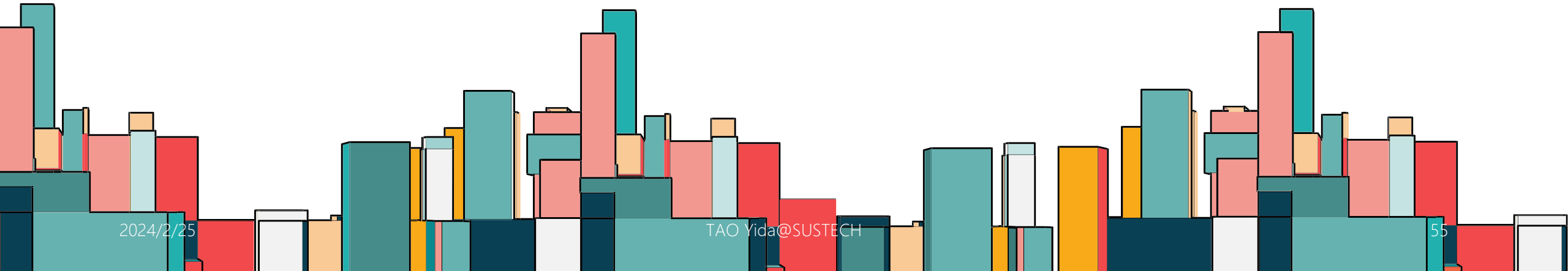
Week 5
Proposal

Week 9
Sprint 1

Week 16
Sprint 2

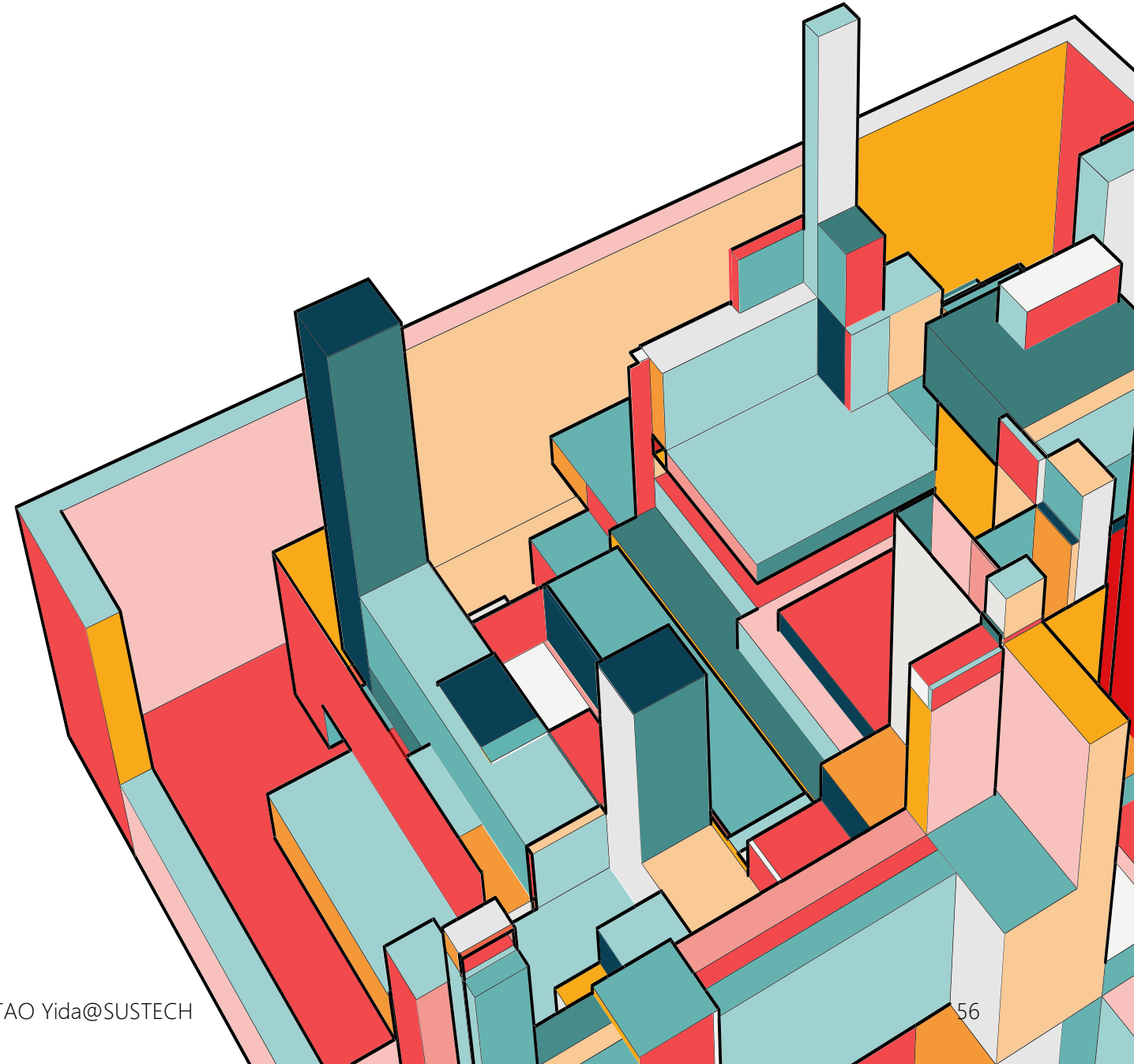
WHAT ABOUT THE OPERATIONS TEAM?

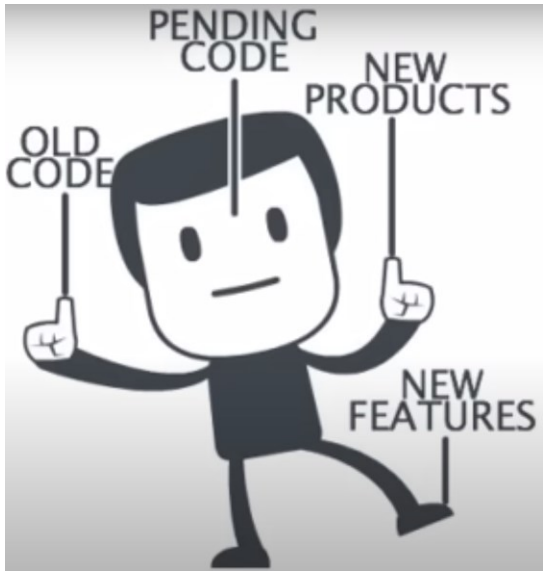
- Agile is typically used within development teams.
- What about the operations team (运维团队) that is responsible for deploying the software and managing the stability of the service and the infrastructure?

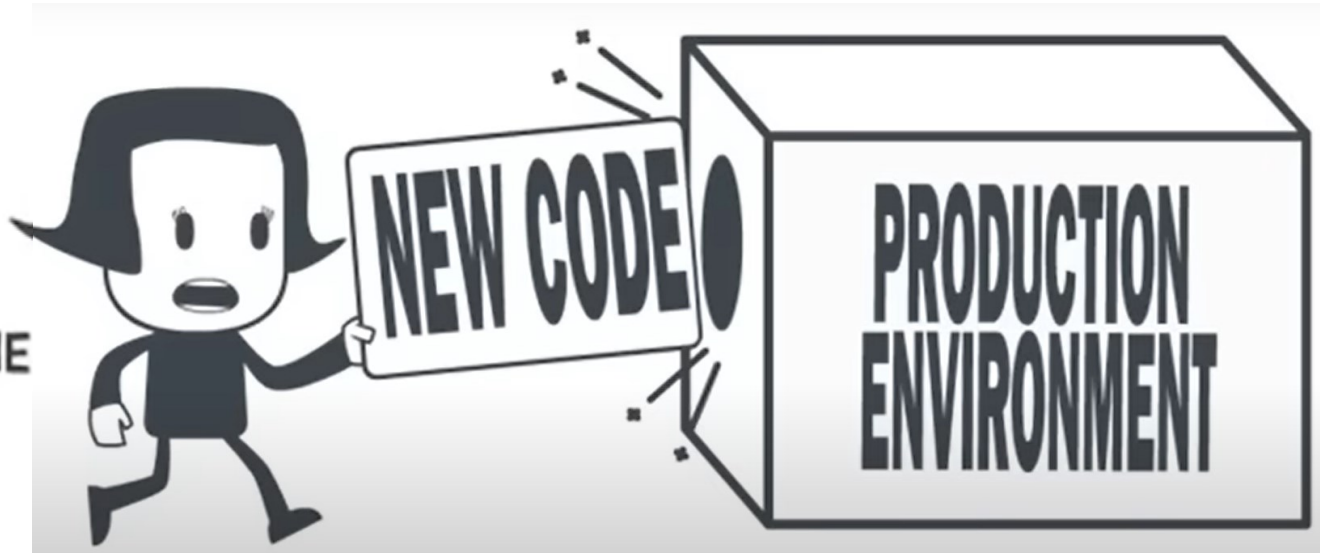


LECTURE 2

- **Overview of software process**
- **Process models**
- **Agile & Scrum**
- **DevOps**





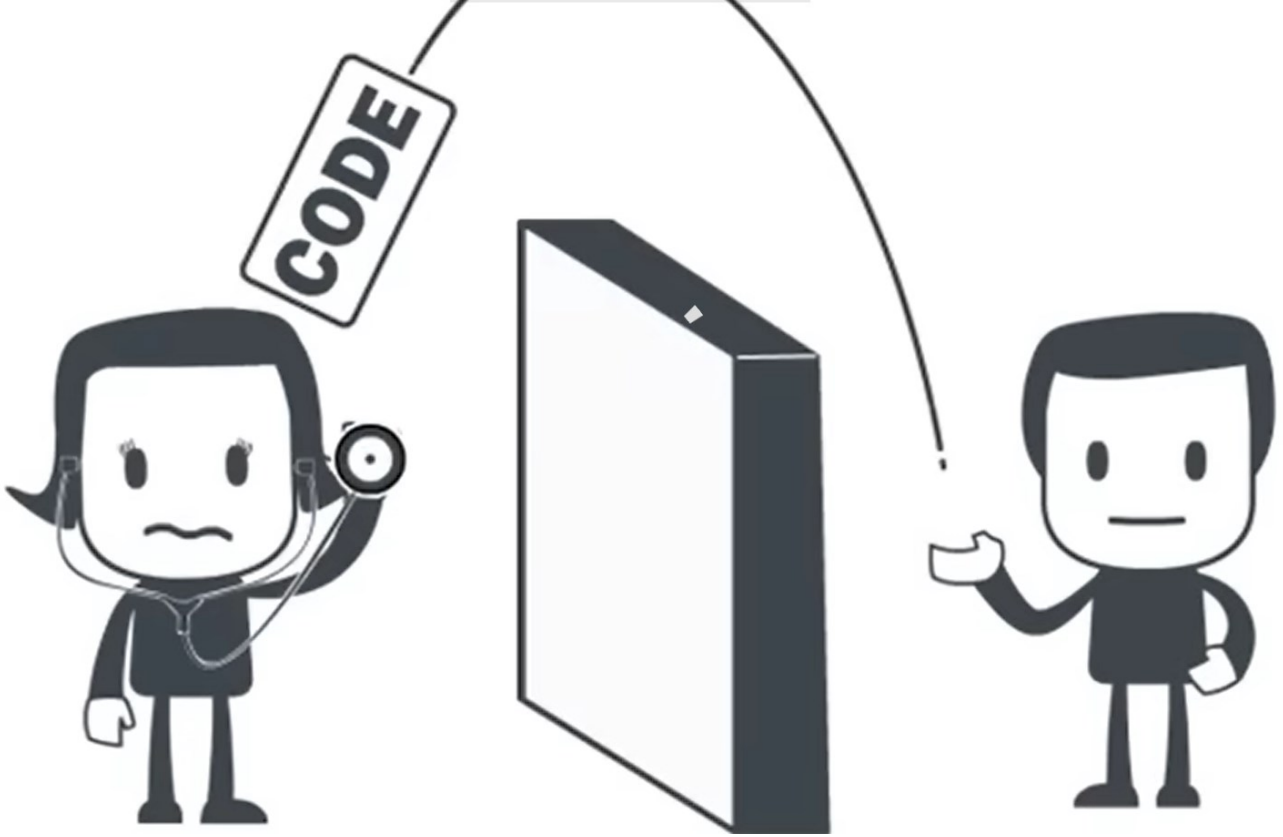


Dev (开发团队) responsibility:

- Design
- Develop
- Test
- Document

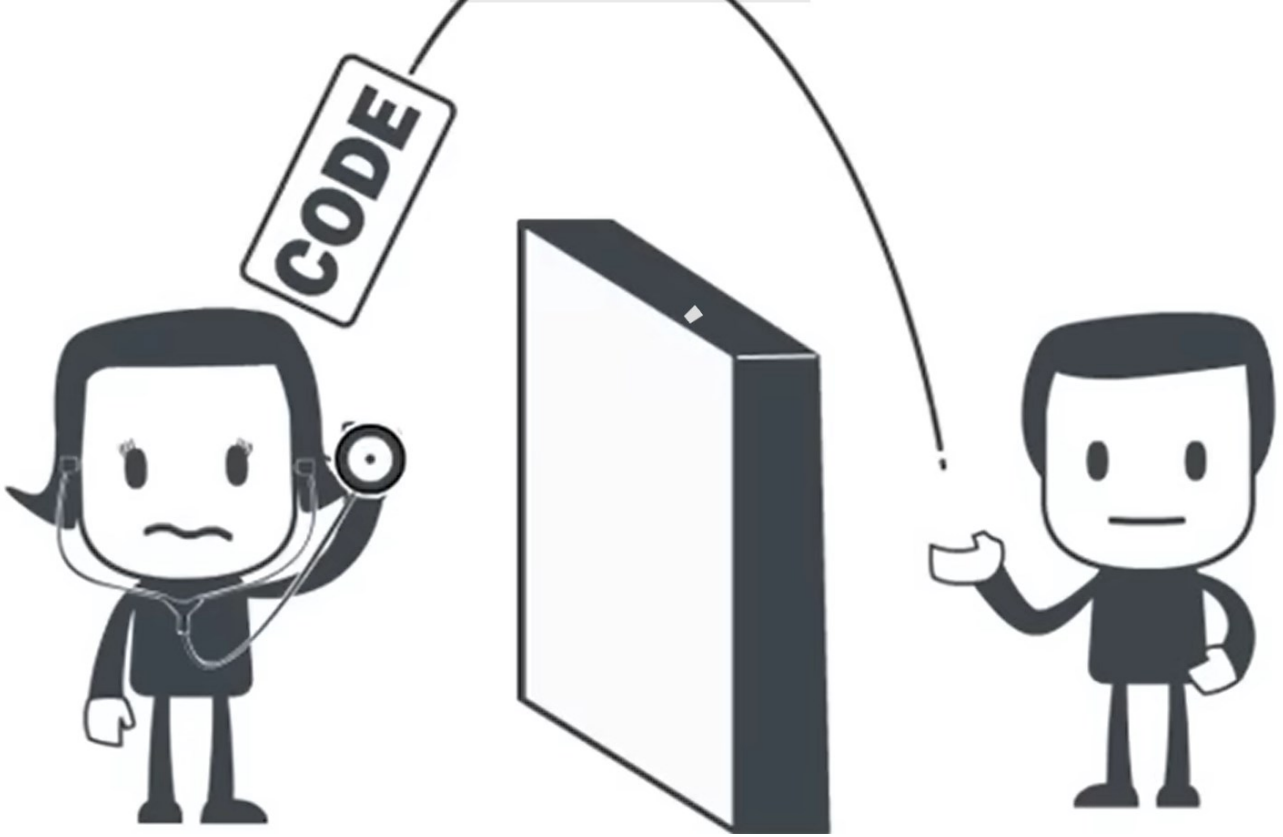
Ops (运维团队) responsibility:

- Deploy
- Release
- Stability of service
- Manage infrastructure
- Maintain & Feedback



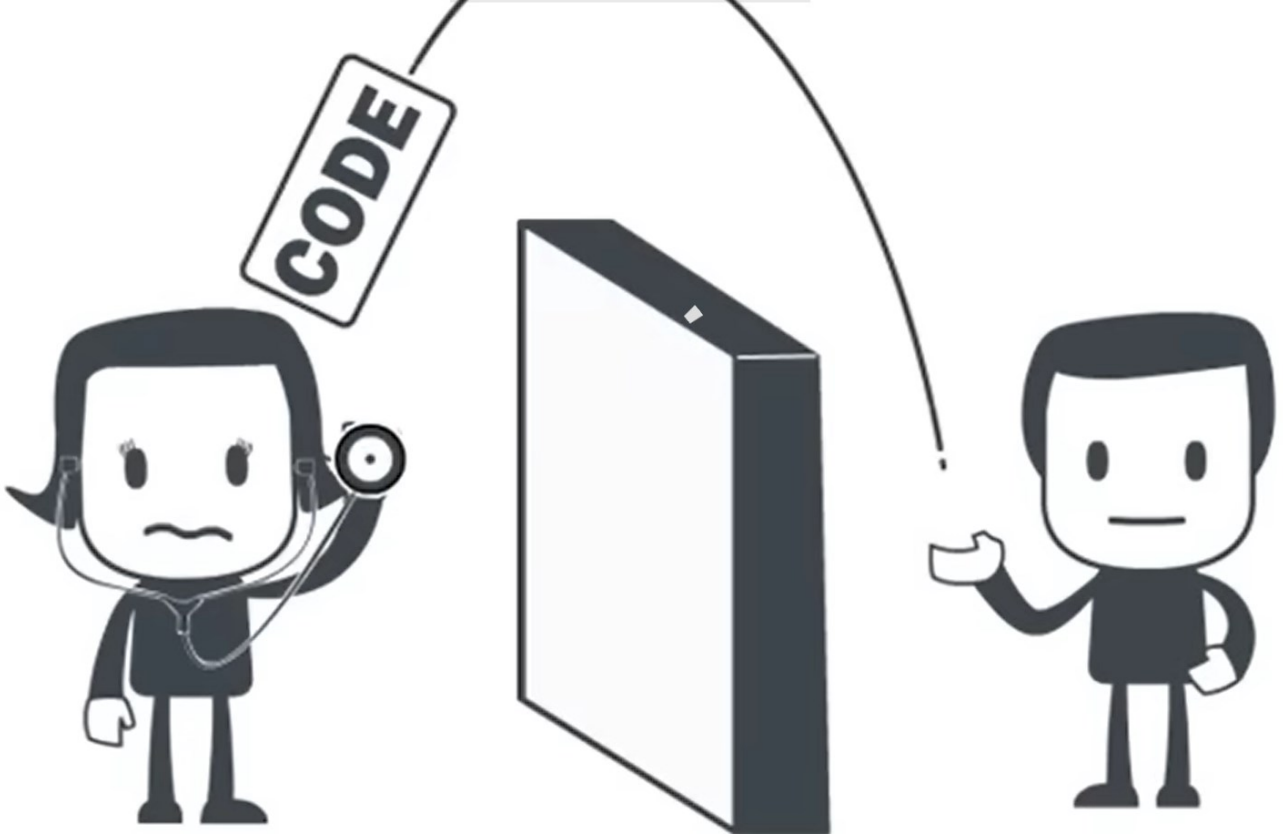
In traditional IT operations, Dev and Ops teams typically work sequentially and individually

Problems?



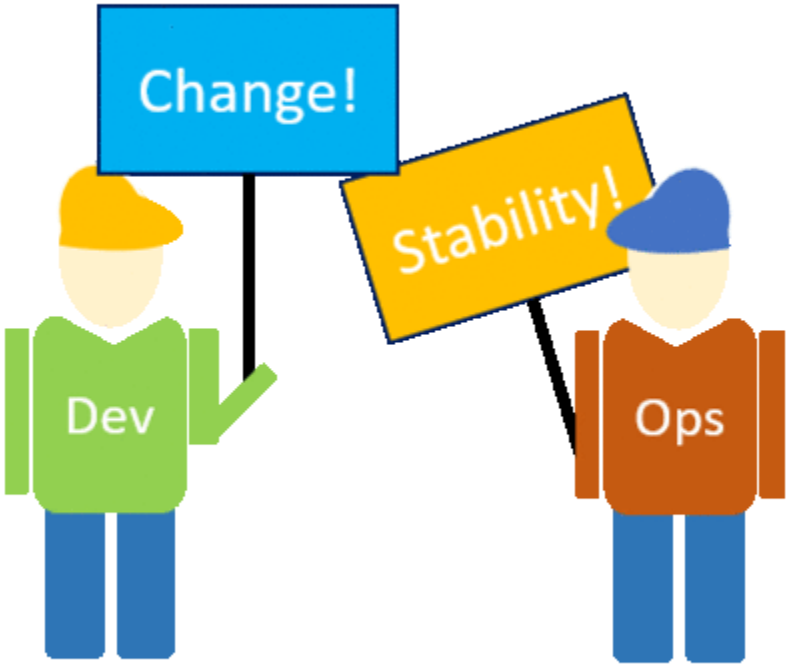
Problem 1: “It works on my machine. It’s your problem now.”

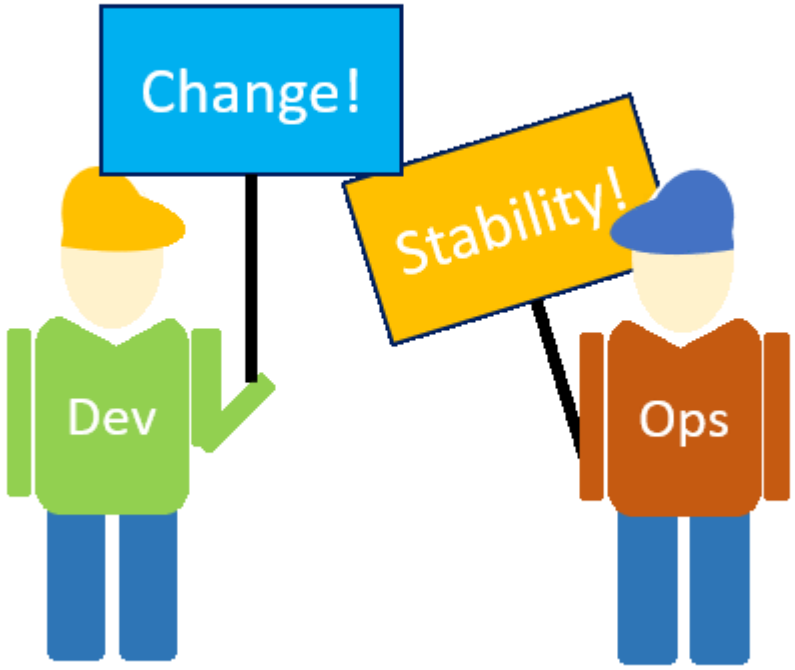
- In traditional software organization, dev-to-ops handoffs are typically one-way, often limited to a few scheduled times in an application's release cycle
- Once in production, the operations team take charge



Problem 2: slow feedback, long time to ship

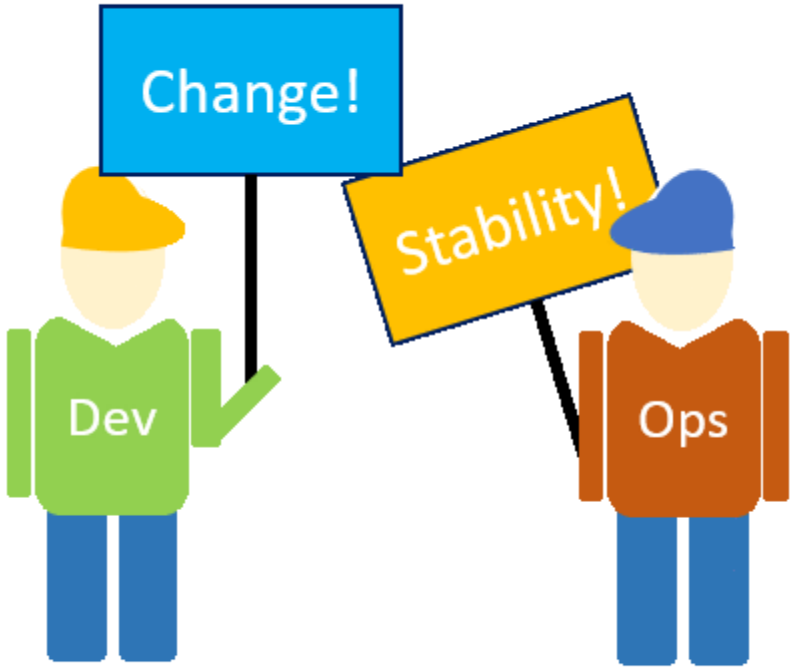
- If there are bugs in the code, the virtual assembly line of Devs-to-QAs-to-OPs is revisited with a patch, with each team waiting on the other for next steps, which might take weeks
- Such a model comes with significant overhead and extends the timeline





Reason: Conflicting objectives for Devs and Ops

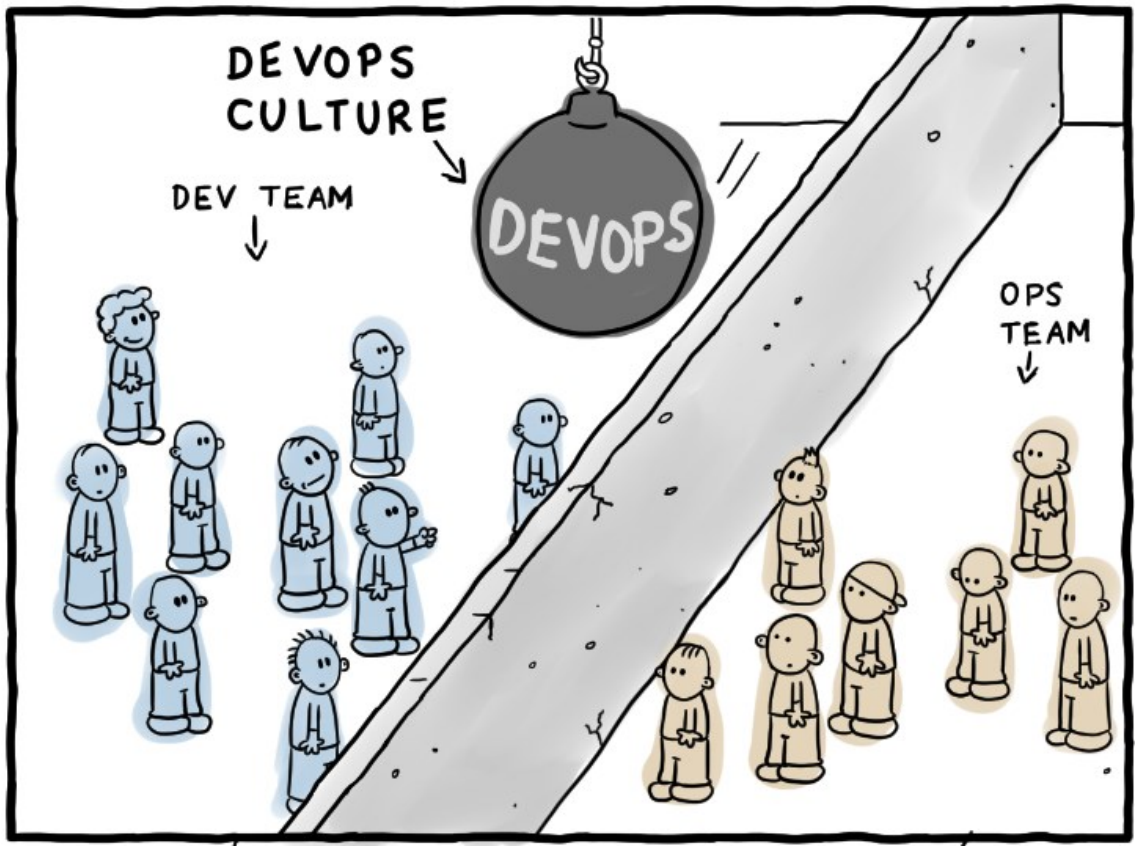
Image source: https://ralfneubauer.info/devops-gifs-reactions/

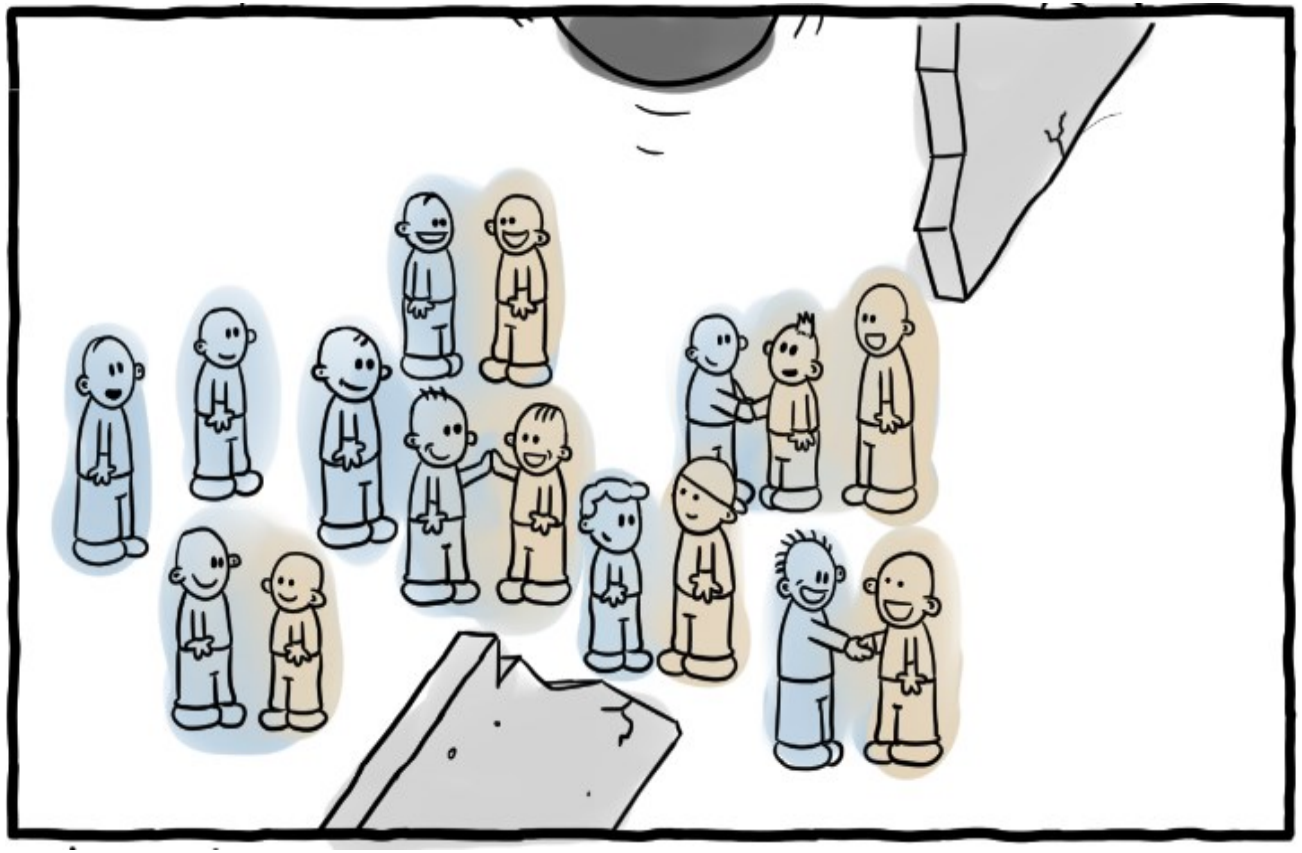


DevOps: Dev and Ops working closely and collaboratively with the same objective

Image source: https://ralfneubauer.info/devops-gifs-reactions/

BEFORE/AFTER DEVOPS

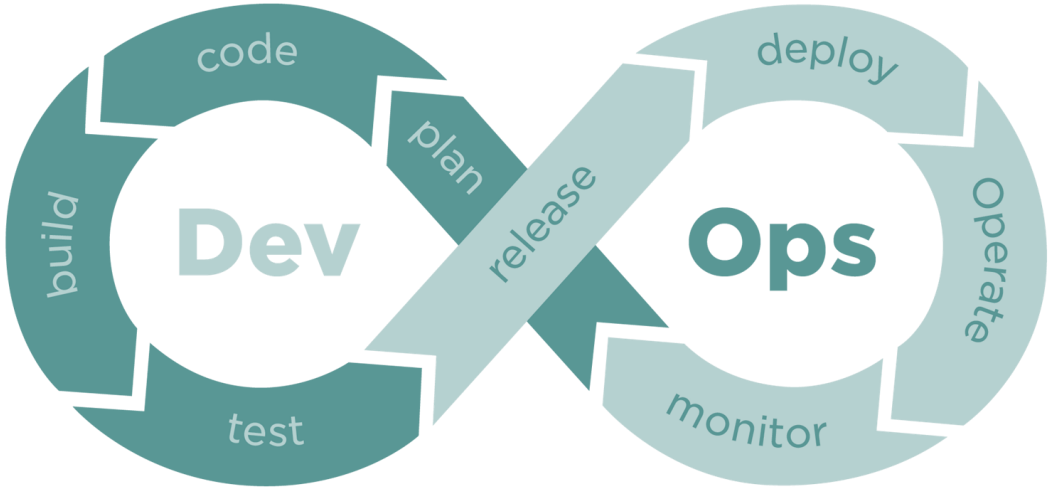




https://turnoff.us/geek/devops-explained/

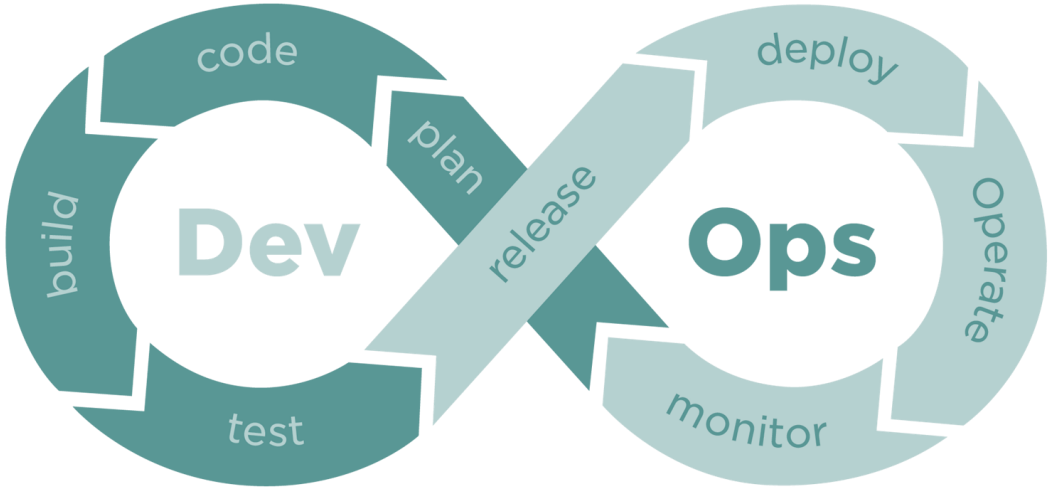
DEVOPS

- DevOps integrates developers and operations teams to improve collaboration and productivity by
  - Automating infrastructure
  - Automating workflows
  - Continuously measuring application performance
  - Continuous feedback

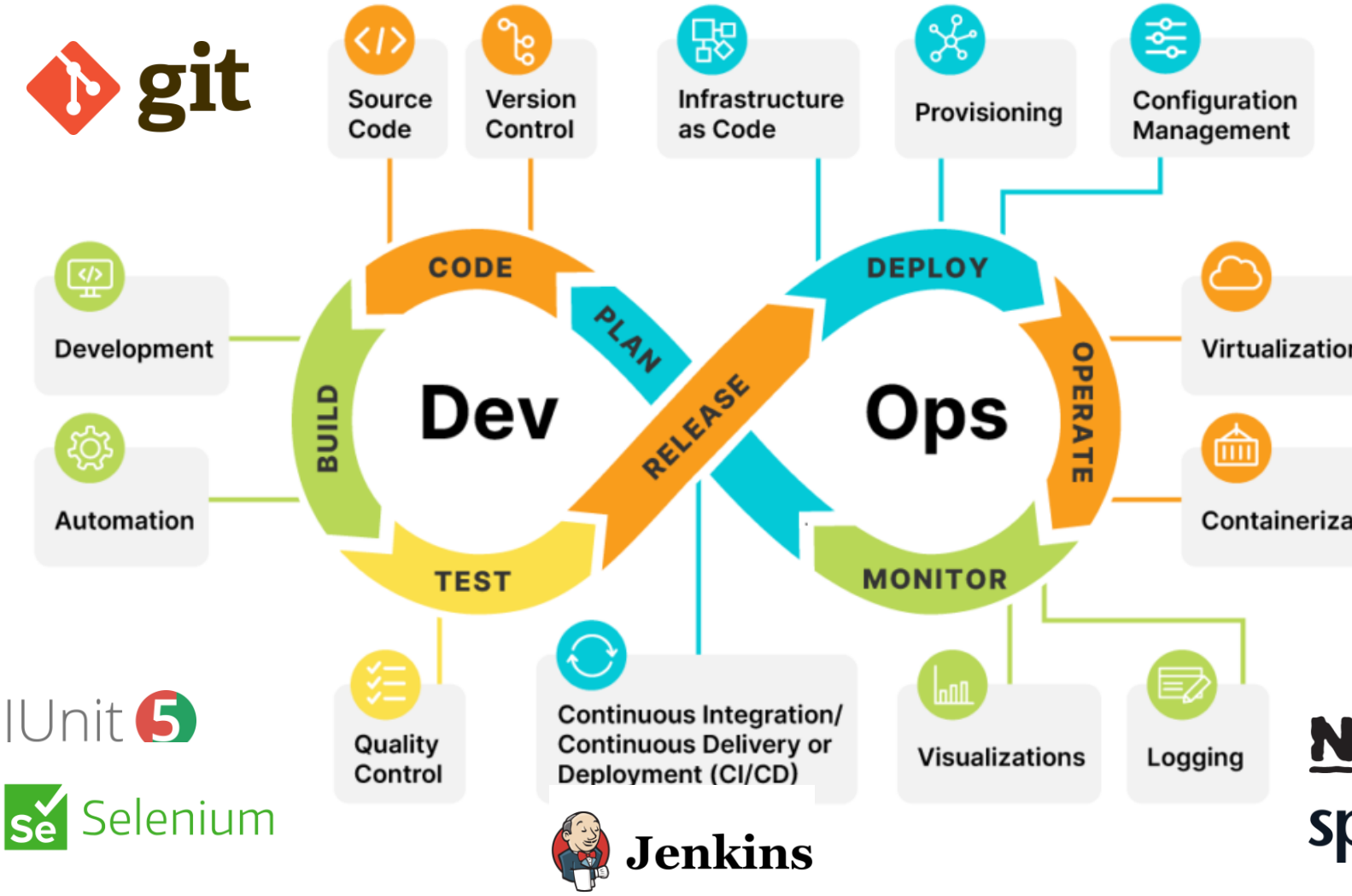


DEVOPS

- DevOps requires that development teams and operations teams **communicate frequently** and approach their work with empathy for their teammates.
- Developers work closely with IT operations to speed software builds, tests, and releases, without sacrificing reliability.
- Whoever needs power the most, get it — through self service and automation.



DEVOPS PHASES & TOOLS



git

ANSIBLE

Maven

Gradle

docker

aws

kubernetes

IUnit 5

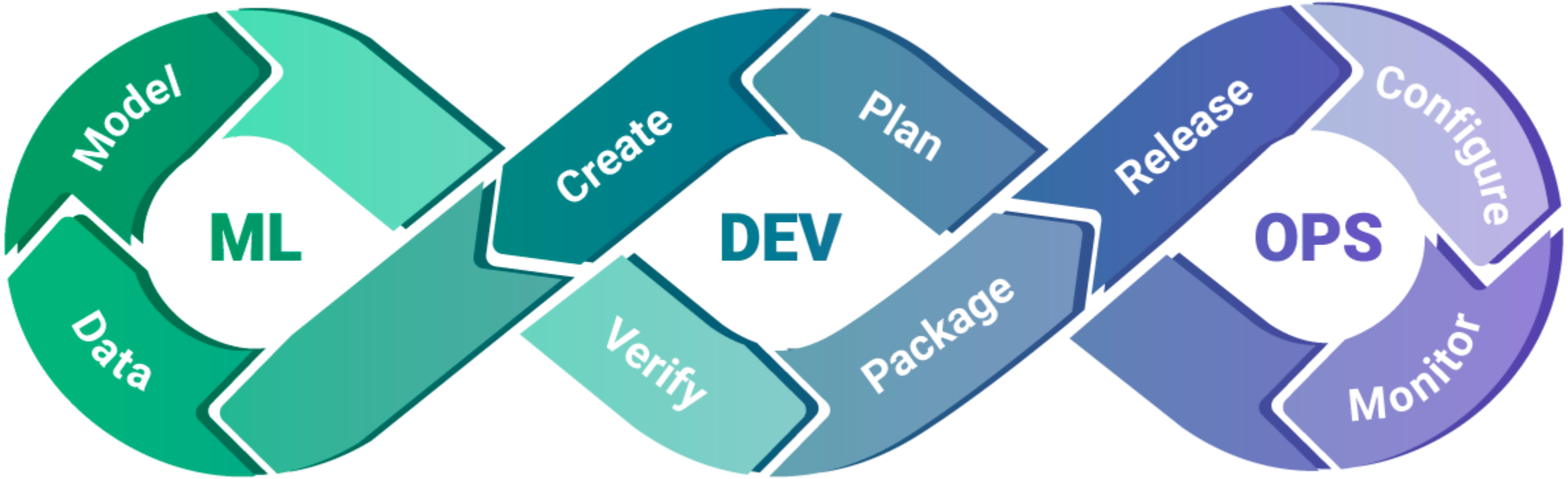
Selenium

Jenkins

Nagios

splunk>

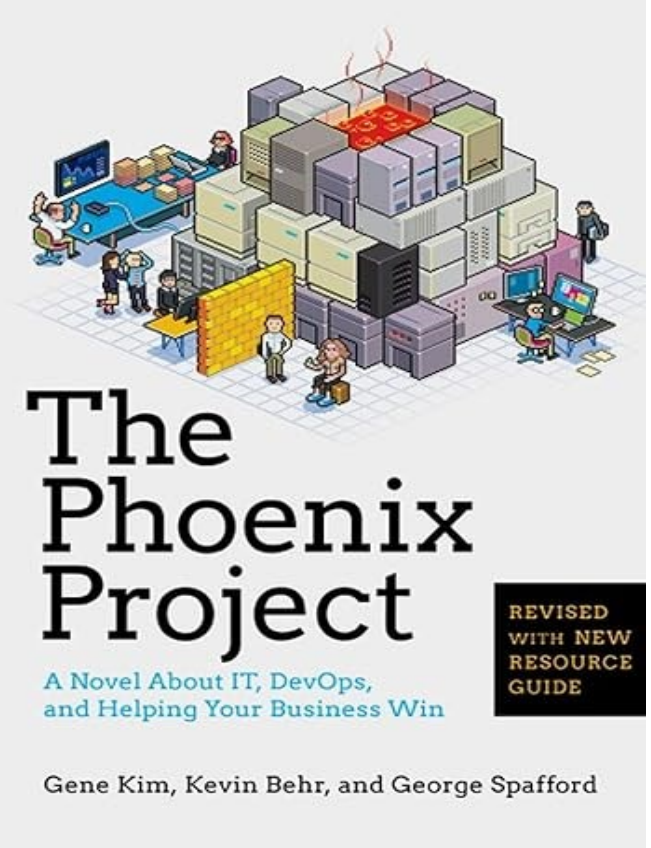
AI + DEVOPS

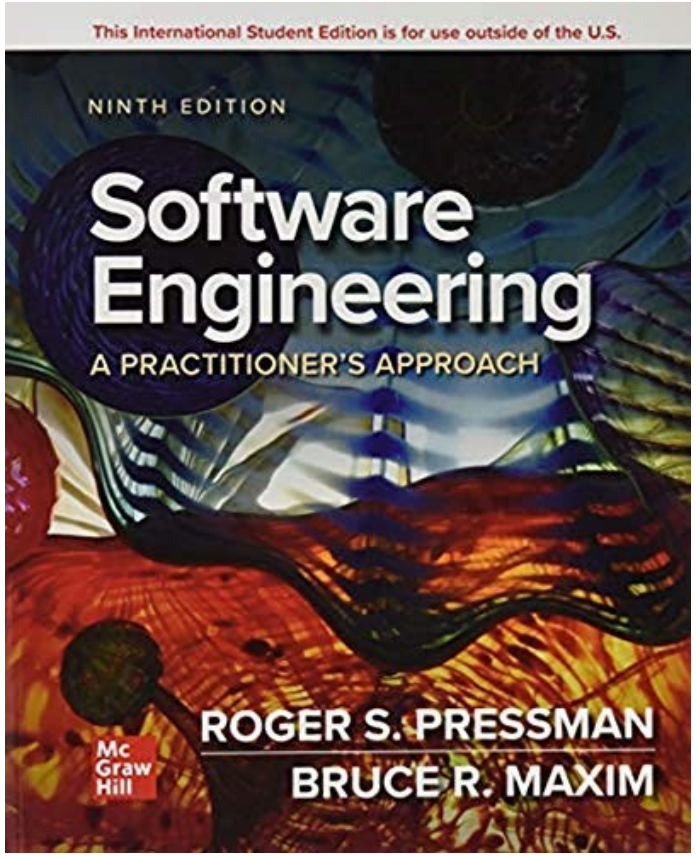


https://softjourn.com/insights/how-ai-is-transforming-devops

READINGS

- Chapter 2: Process Models
- Chapter 3: Agile Development





NEXT

- Software Requirements