



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

CS304 SOFTWARE ENGINEERING

Yida Tao

taoyd@sustech.edu.cn



PREVIOUSLY

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



PREVIOUSLY

- Writing code is easy
- Engineering good software system is HARD
 - Different roles are involved
 - Many aspects of complexity (time, scale, etc.)
 - Trade-offs & making good decisions



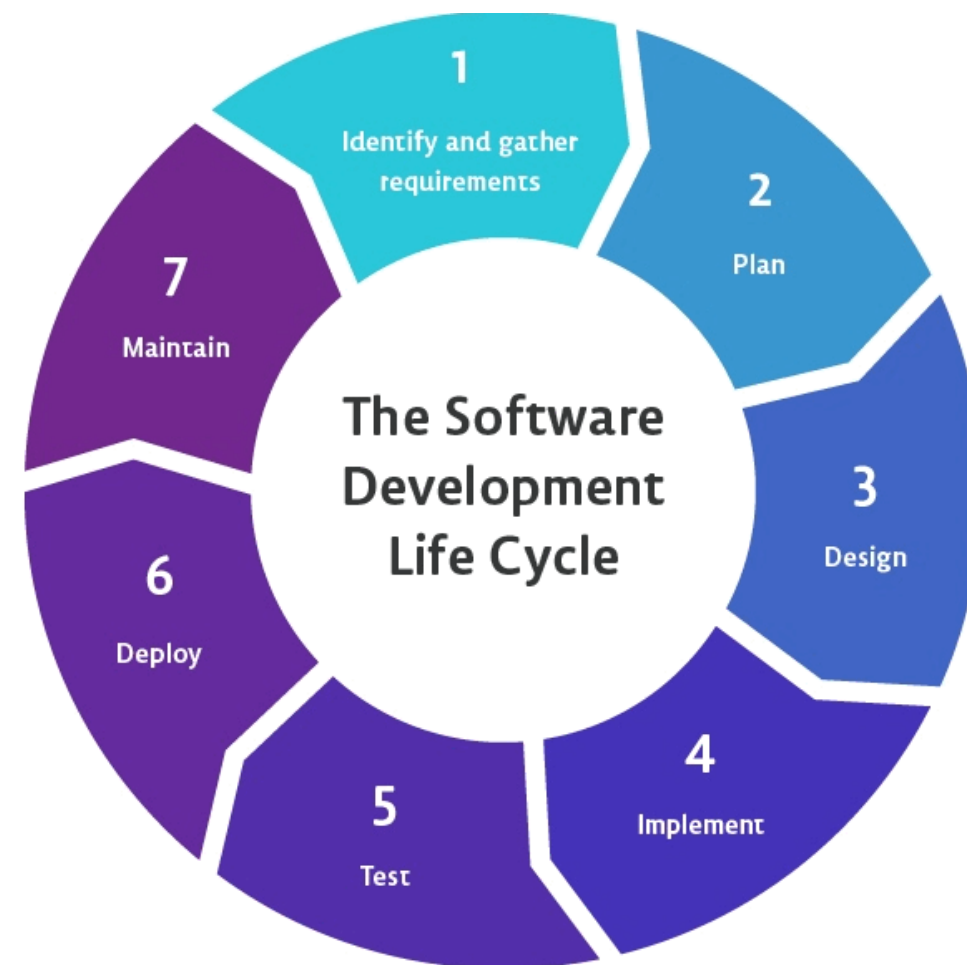
LECTURE 2

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



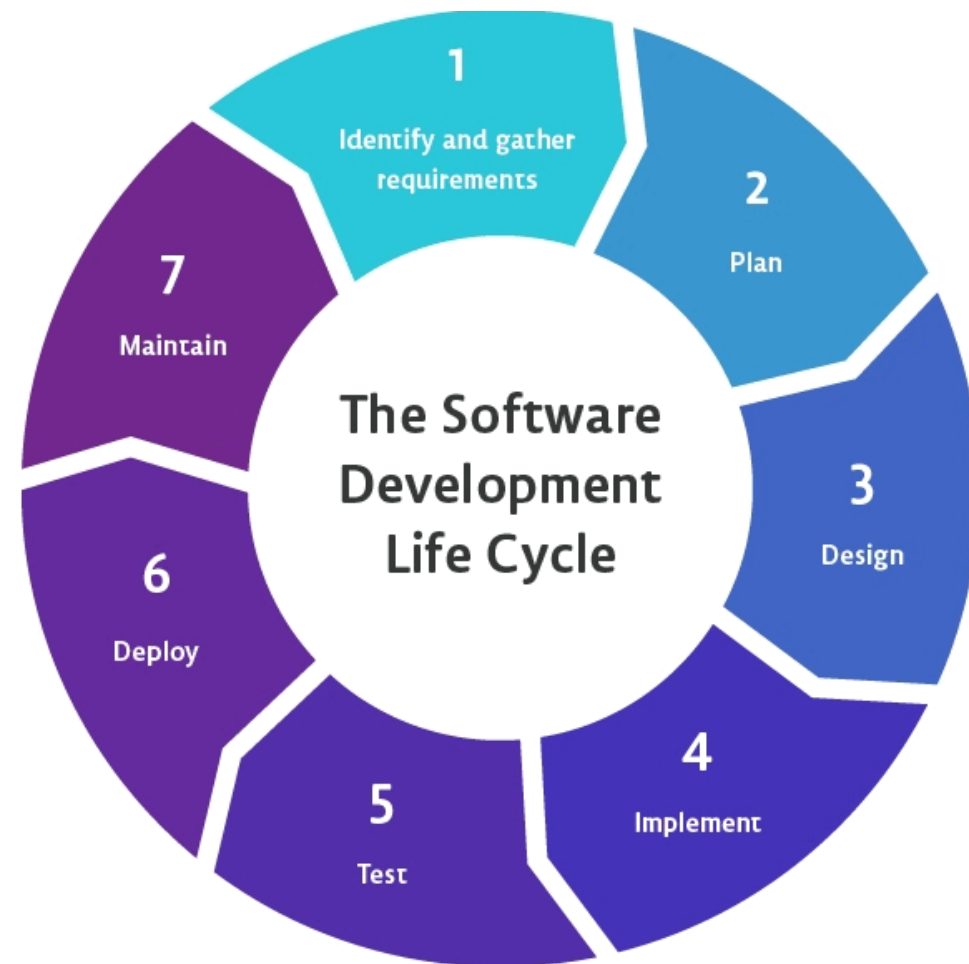
7 PHASES IN SOFTWARE LIFECYCLE

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



SOFTWARE PROCESS

Software process (软件过程) is a set of related phases/activities that take place **in certain order**, and leads to the production of a software product.





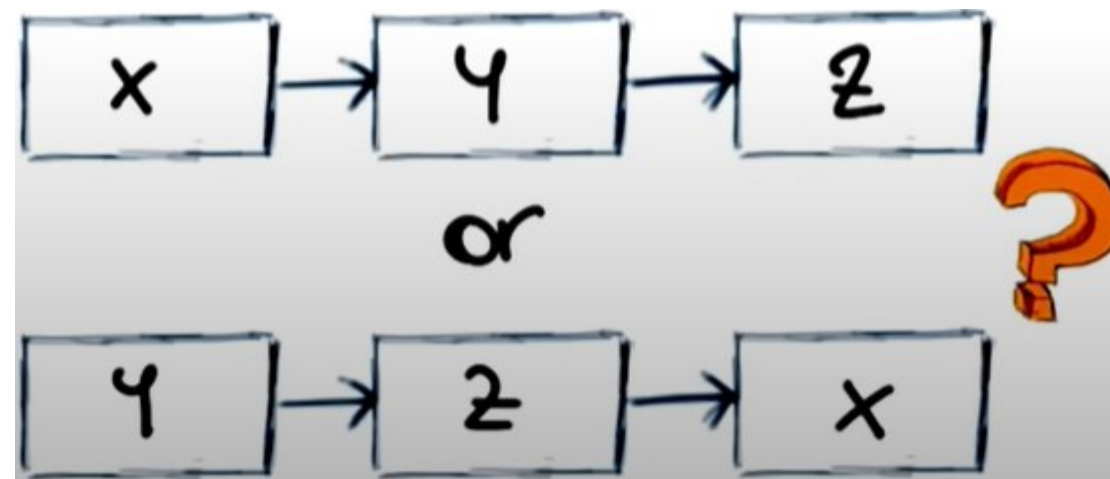
WHY SOFTWARE PROCESS?

- Organizations want a **well-defined, well-understood, repeatable** software process
- Benefits
 - New team members know what to do
 - Know what's been done
 - Estimate time to completion, costs, etc.
 - Follow and repeat good practices

SOFTWARE PROCESS MODELS

Software process models (or software lifecycle model) were originally proposed to bring order to the chaos of software development, it determines

- The **order** of these phases
- The **criteria for transition** of each phase





SOFTWARE PROCESS MODELS

Software process models fall on a **continuum**

Planning is incremental and it is easier to change the process to reflect changing customer requirements.

All of the process activities are planned in advance and progress is measured against this plan.

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.

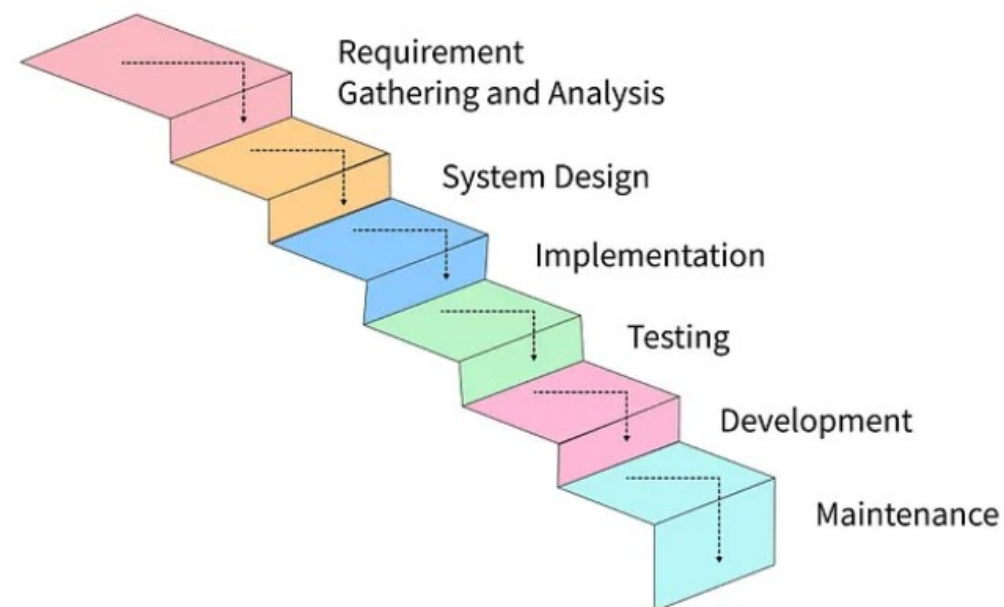


LECTURE 2

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

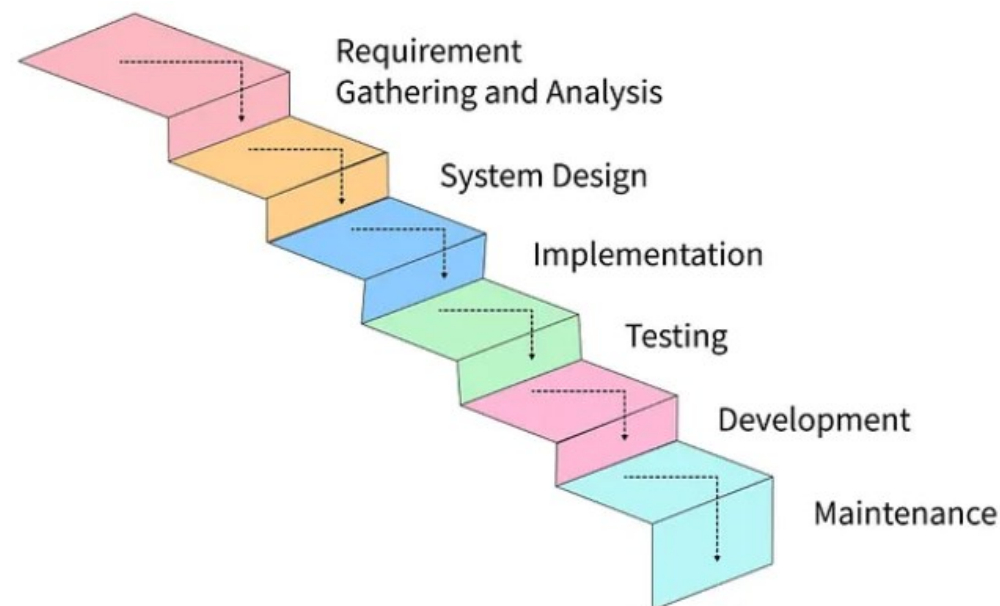
THE WATERFALL MODEL

The waterfall model is a **sequential** and **linear** approach for software development process



THE WATERFALL MODEL

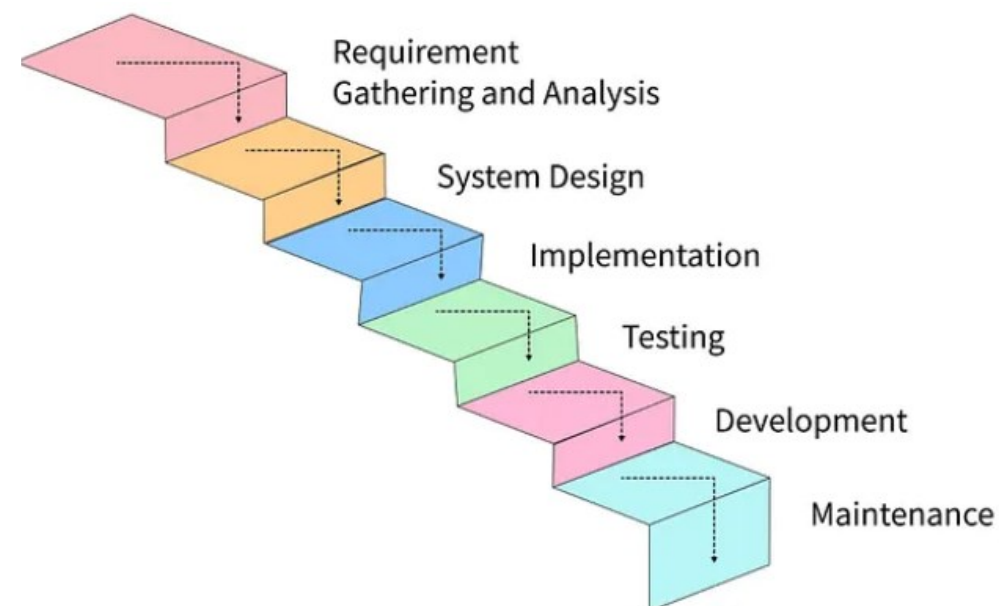
- In principle, the result of each phase is one or more documents that are approved
- A phase should not start until the previous phase has finished. No overlap between phases.



Sequential phases, flow downwards in one direction, hence the name “waterfall”



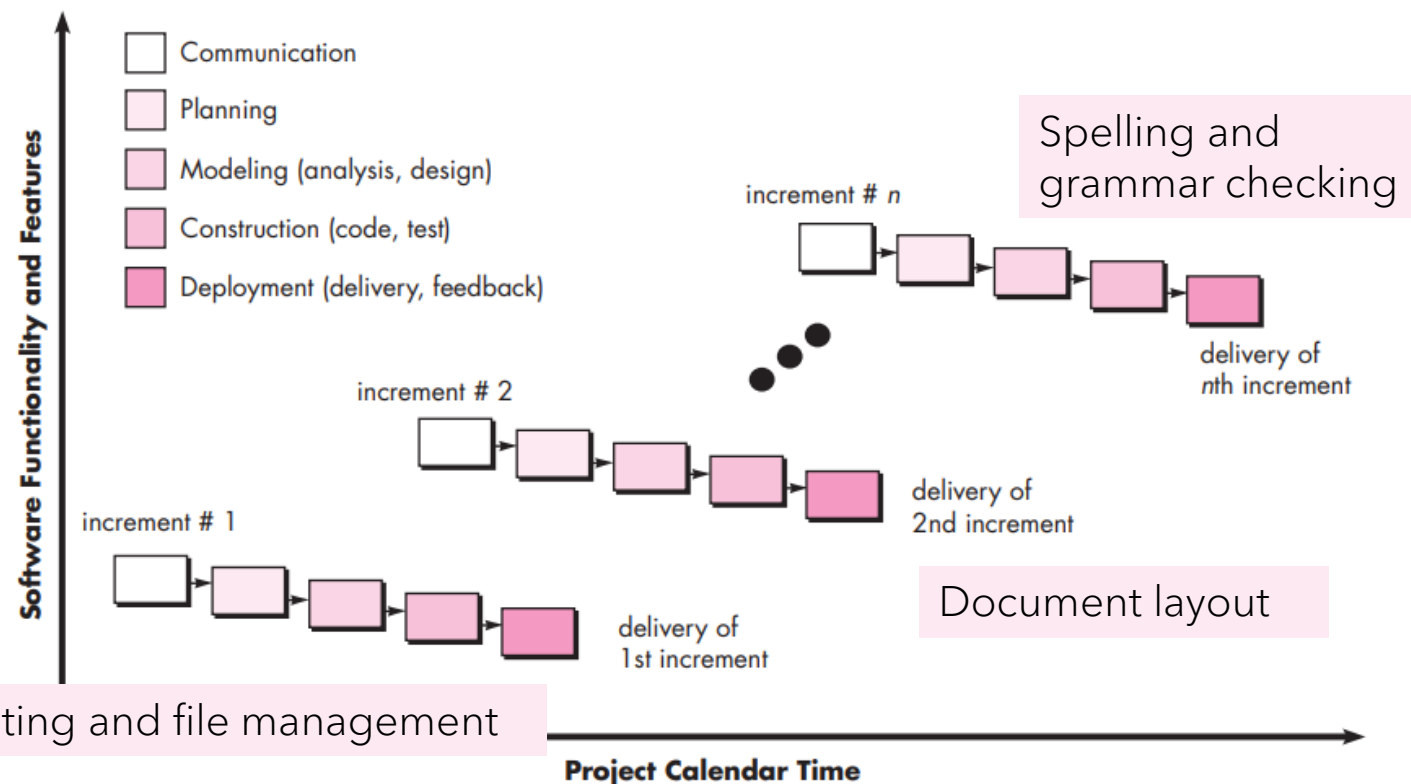
PROBLEMS?



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

INCREMENTAL PROCESS MODELS

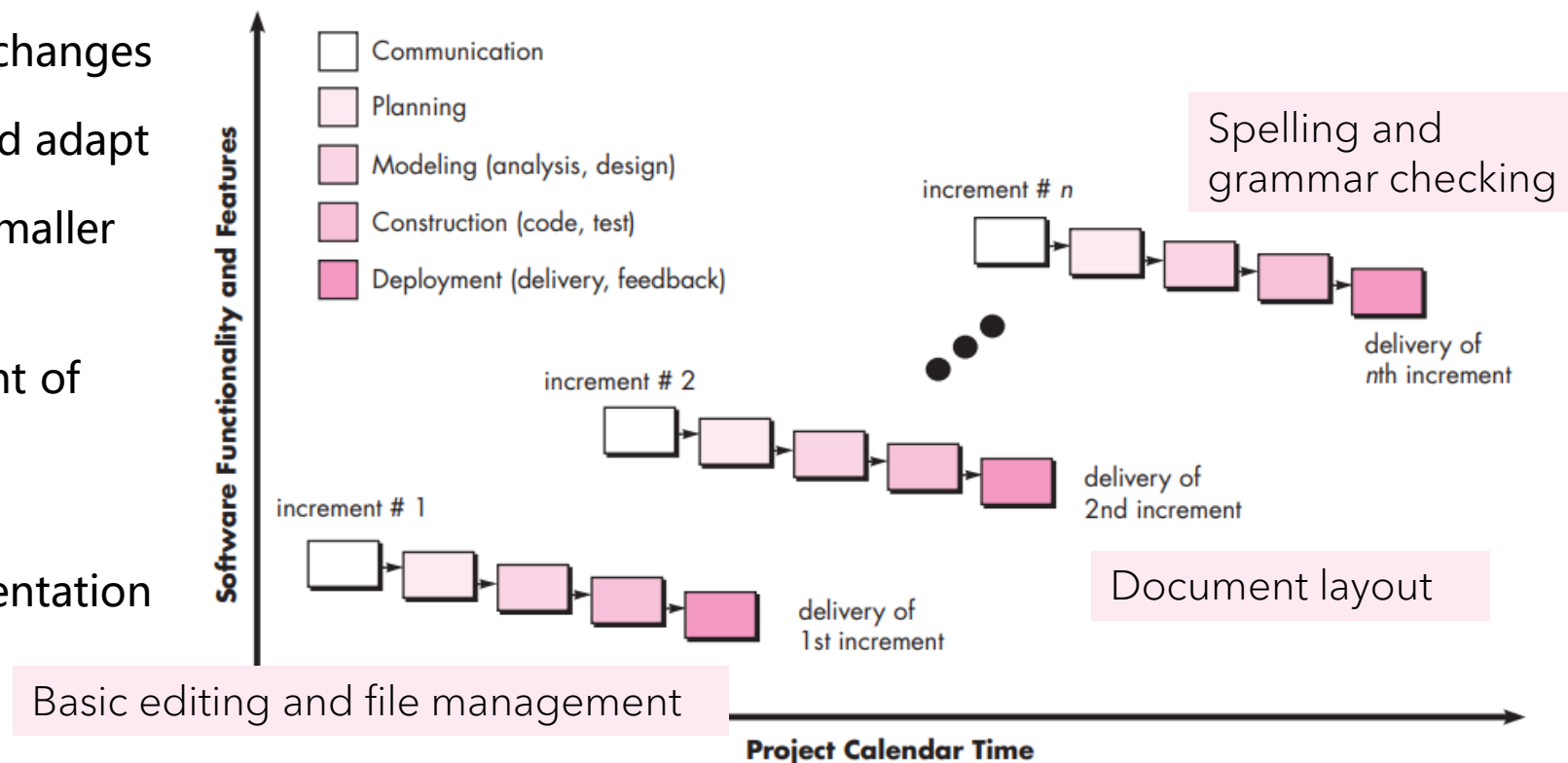
- The incremental model (增量模型) combines elements of linear and parallel process flows
- Each linear sequence produces deliverable “increments” of the software
- The first increment is often a core product with the basic requirements addressed. Supplementary features are developed in subsequent increments.





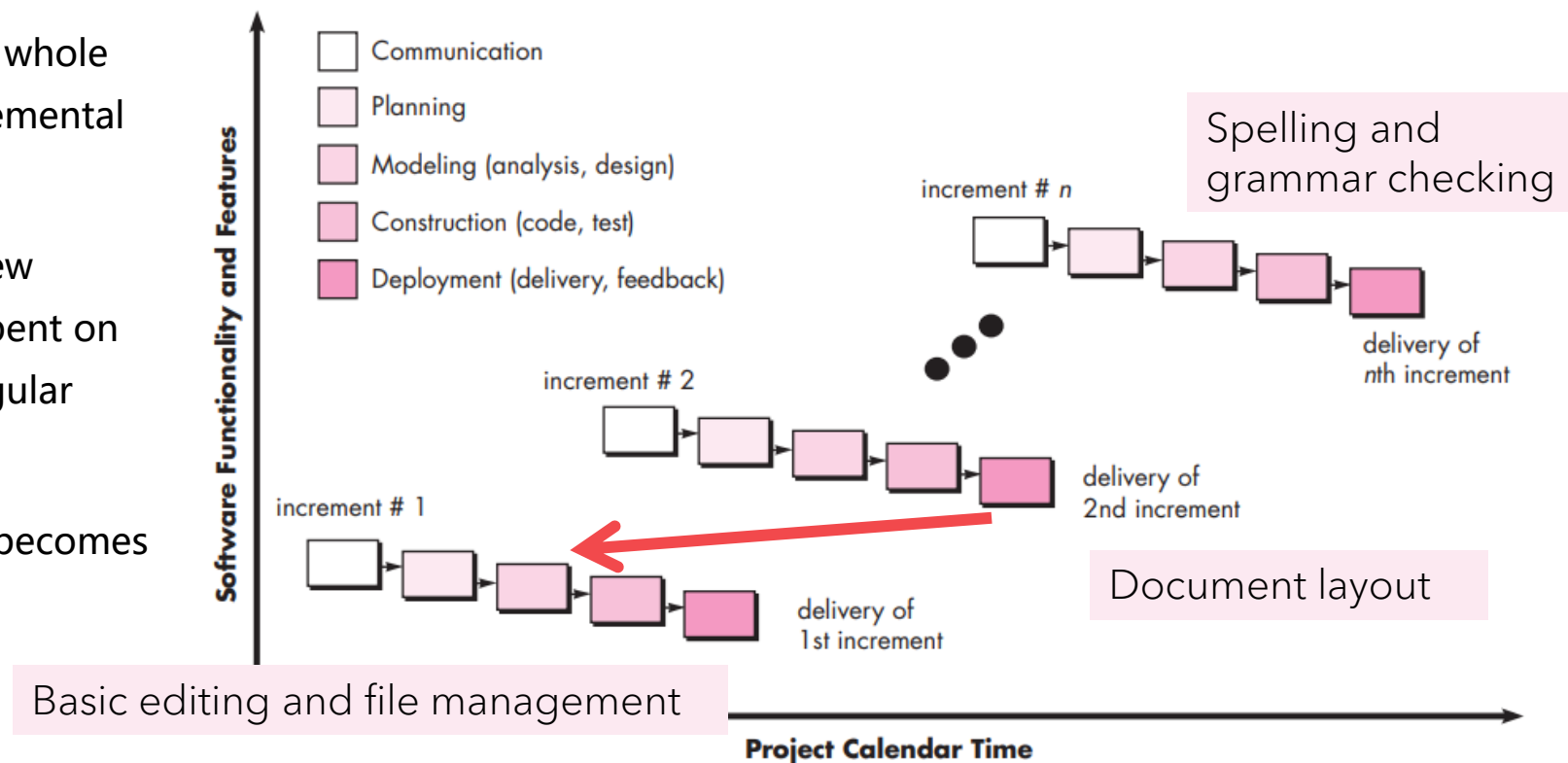
INCREMENTAL PROCESS MODELS PROS

- 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



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.





IS SOFTWARE DEVELOPMENT A LINEAR PROCESS?

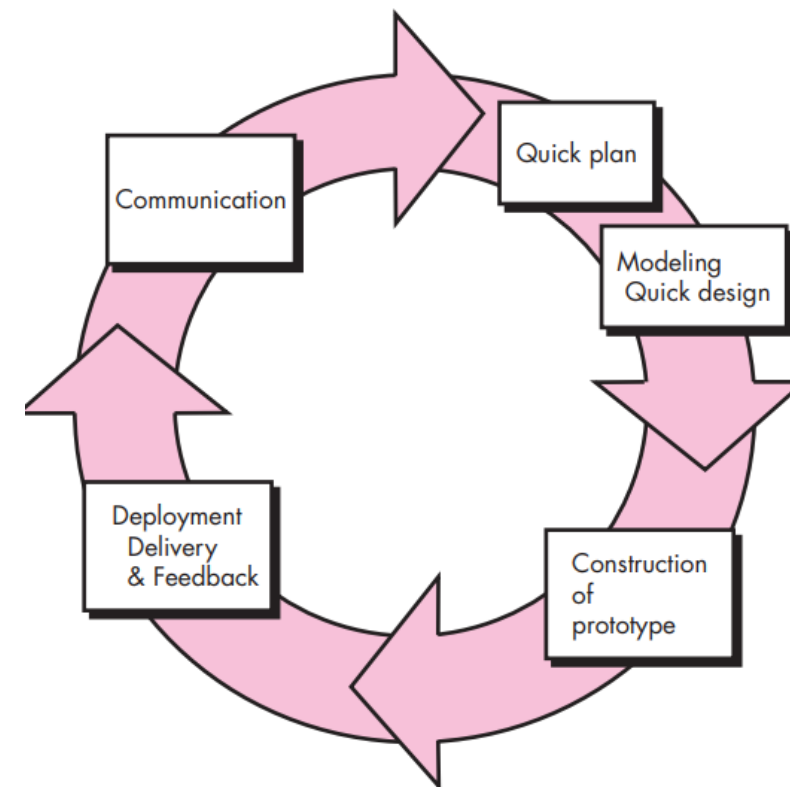


EVOLUTIONARY PROCESS MODELS

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

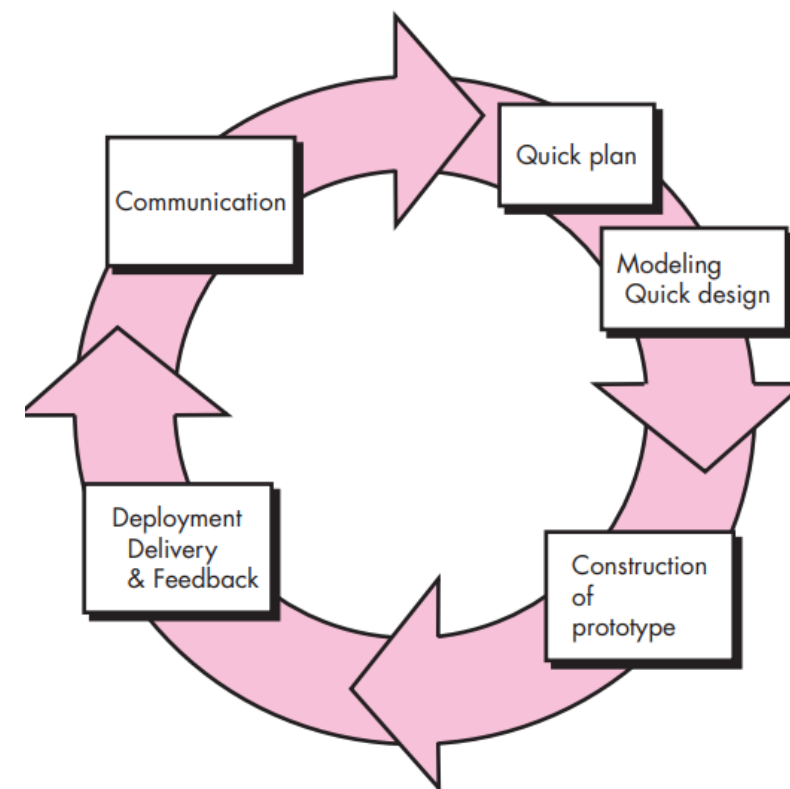
PROTOTYPING

- 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

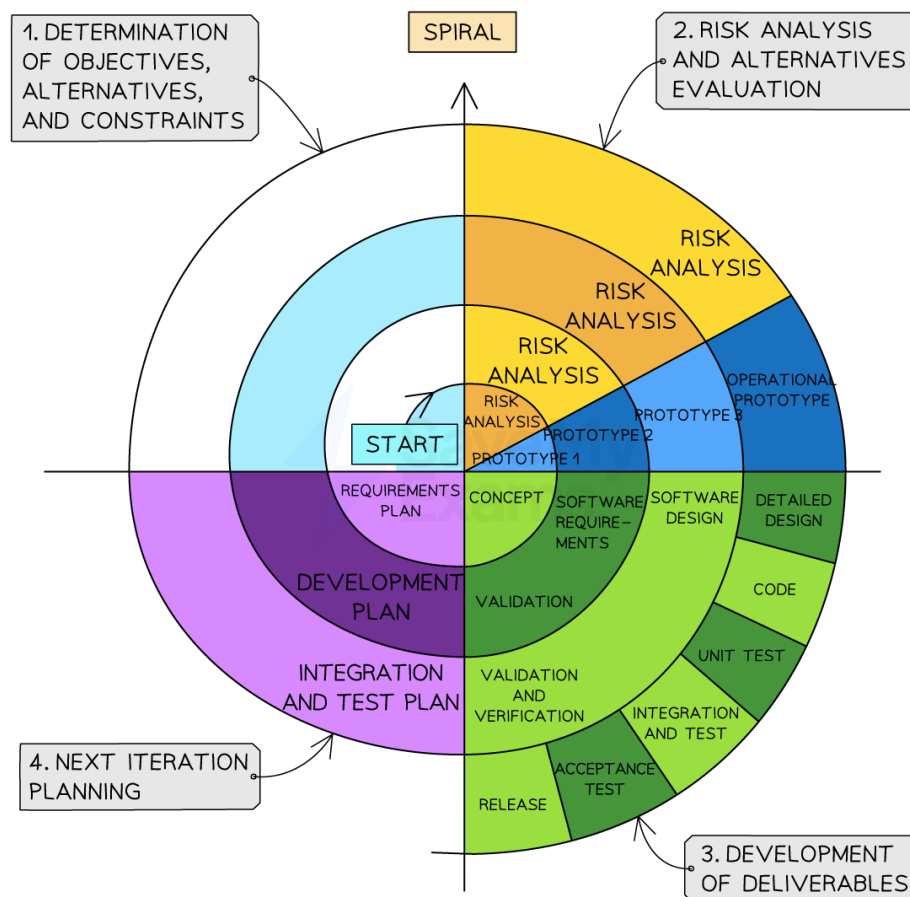


PROTOTYPING

- 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



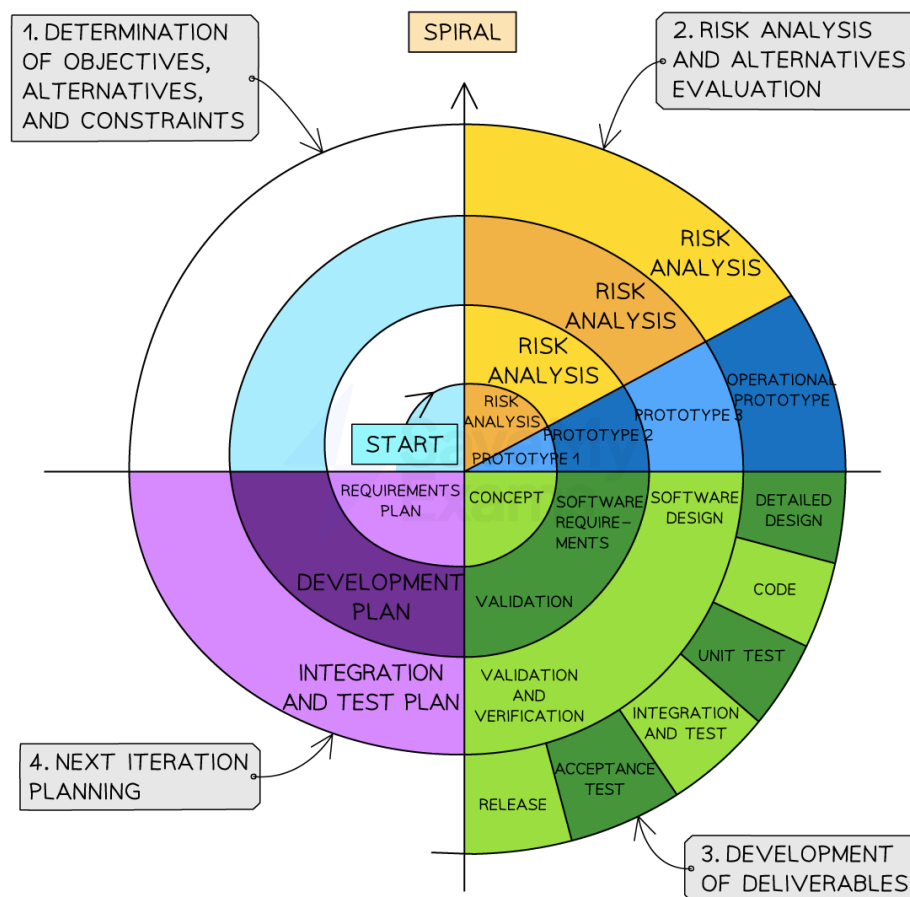
THE SPIRAL MODEL



Copyright © Save My Exams. All Rights Reserved

- The spiral model couples the iterative nature with the controlled and systematic aspects of **the waterfall model**
- 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

THE SPIRAL MODEL



Copyright © Save My Exams. All Rights Reserved

Pros

- Effectively reducing potential risks
- Features can be updated because of the iterative nature
- Better for high-risk project, e.g., banking, medical systems

Cons

- Risk analysis highly rely on experts
- Complex and costly



PLAN-DRIVEN SOFTWARE PROCESS MODELS

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

As consistency in action is a human weakness, high-discipline methodologies are **fragile**

Agile

Plan-driven

FRAGILE

Agile Manifesto Agile Values

We are uncovering **better ways of developing software by doing it and helping others do it.** Through this work we have come to value:



Individuals and interactions

over

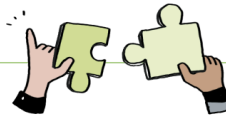
processes and tools



Working software

over

comprehensive documentation



Customer collaboration

over

contract negotiation



Responding to change

over

following a plan

That is, while there is **value in the items on the right**, we **value the items on the left more**.

<https://agilemanifesto.org/>

@SketchingSM
www.sketchingscrummaster.com



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

AGILE VALUES

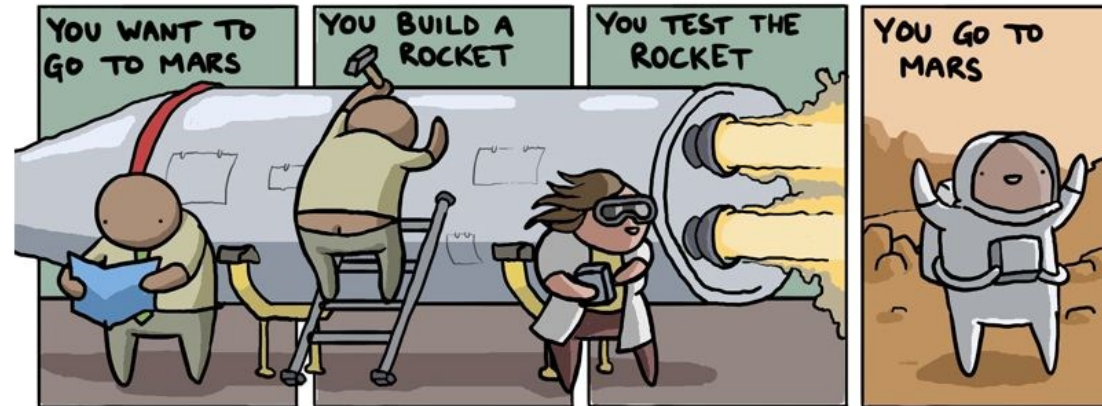
In 2001, Kent Beck and 16 other noted software developers, writers, and consultants signed the “Manifesto for Agile Software Development.”
(敏捷软件开发宣言)

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 VS. WATERFALL



THE WATERFALL METHOD



AGILE DEVELOPMENT



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

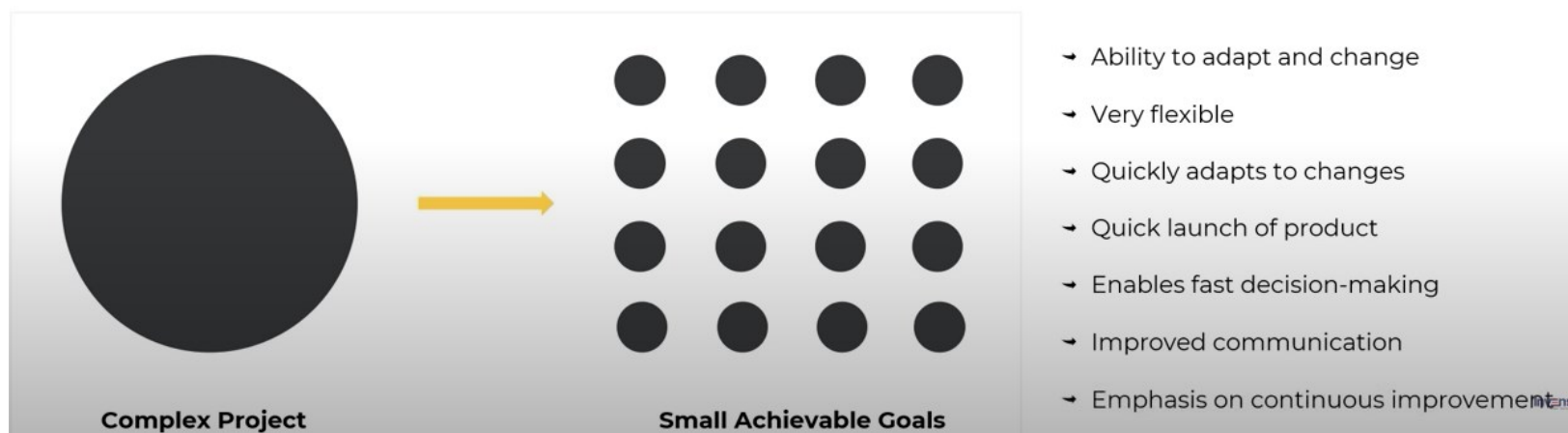


WHAT IS AGILE, EXACTLY?

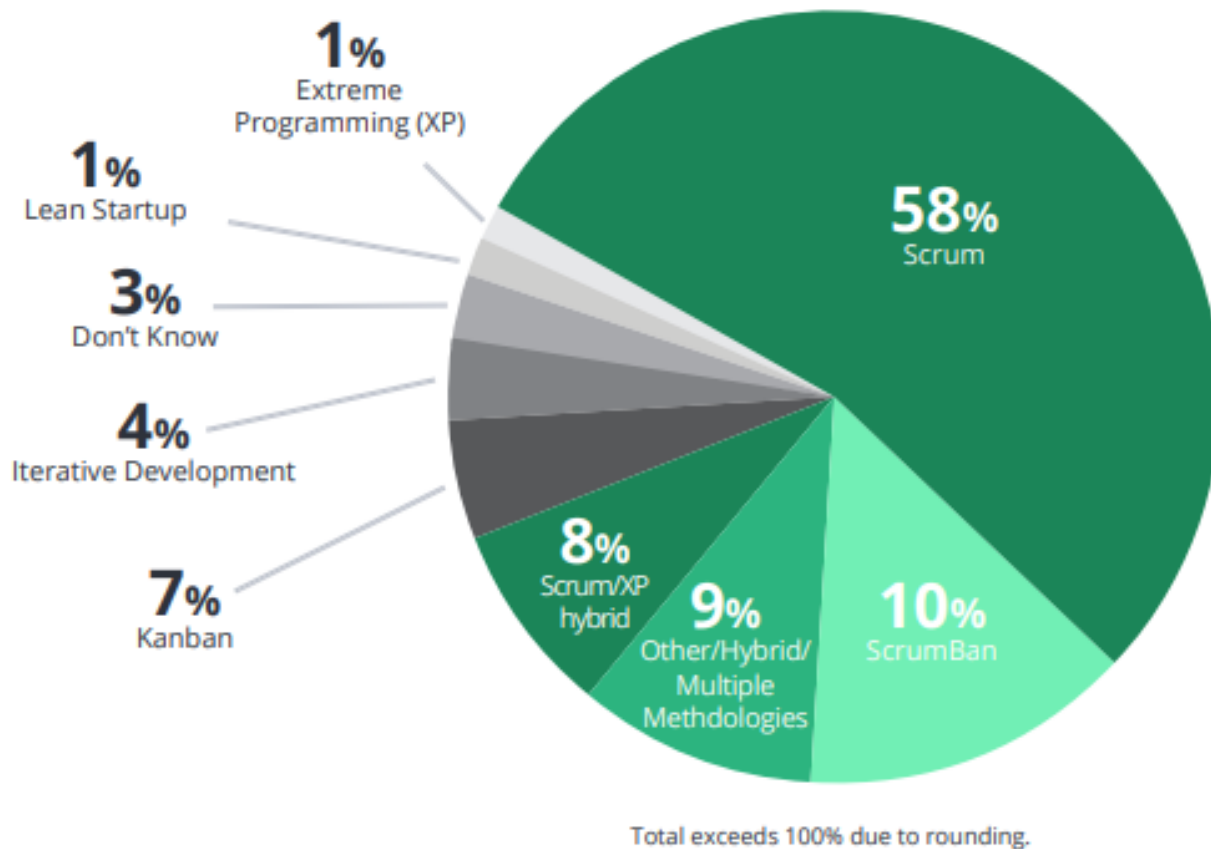
Agile (敏捷), **in general**, is the ability to create and respond to change.

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 METHODOLOGY



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 quickly adapt to unexpected, complicated situations
- Process of sport games = Process of software development





SCRUM ROLES

Product Owner

Define and adjust product features; Accept or reject work results

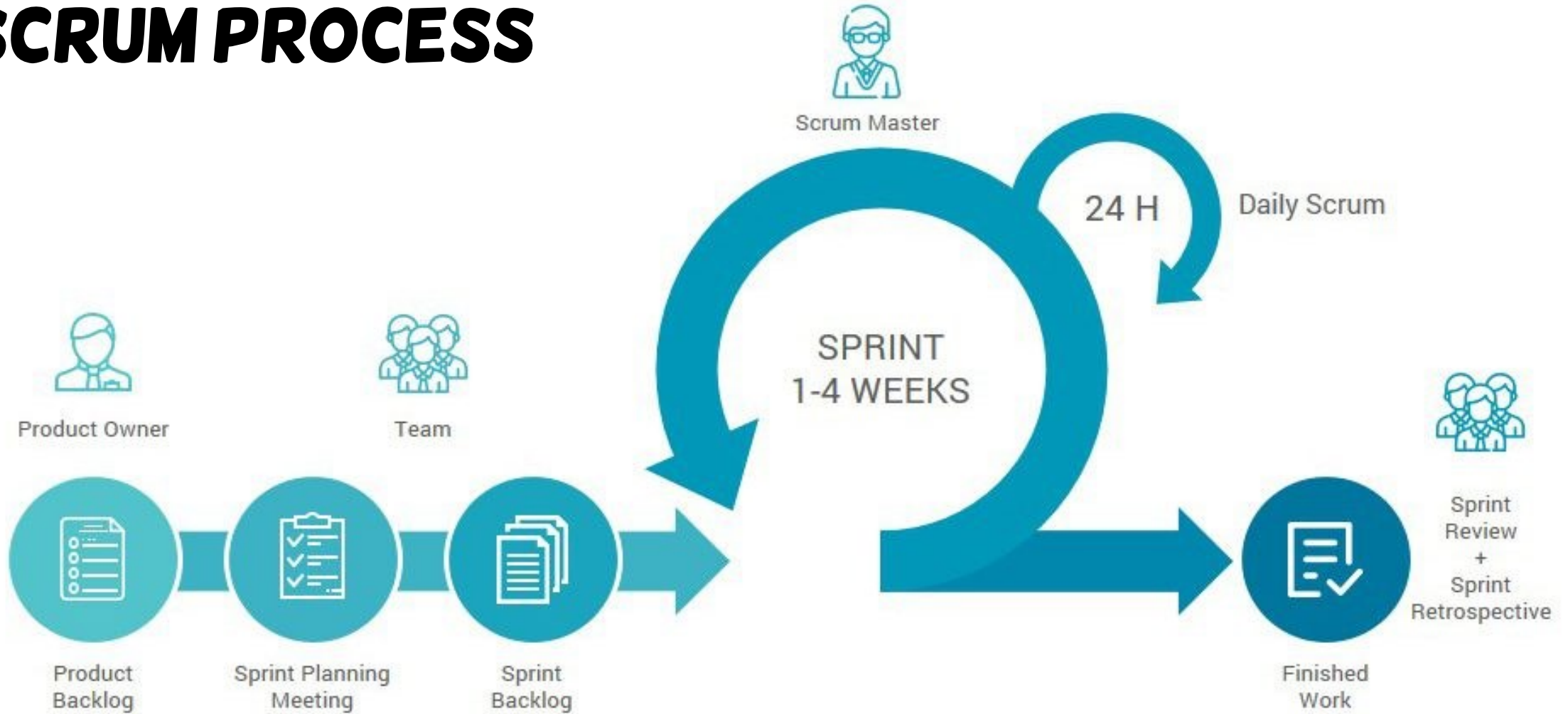
Scrum Master

Coach the team and enact Scrum values and principles

Scrum Team

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

SCRUM PROCESS



Sprint

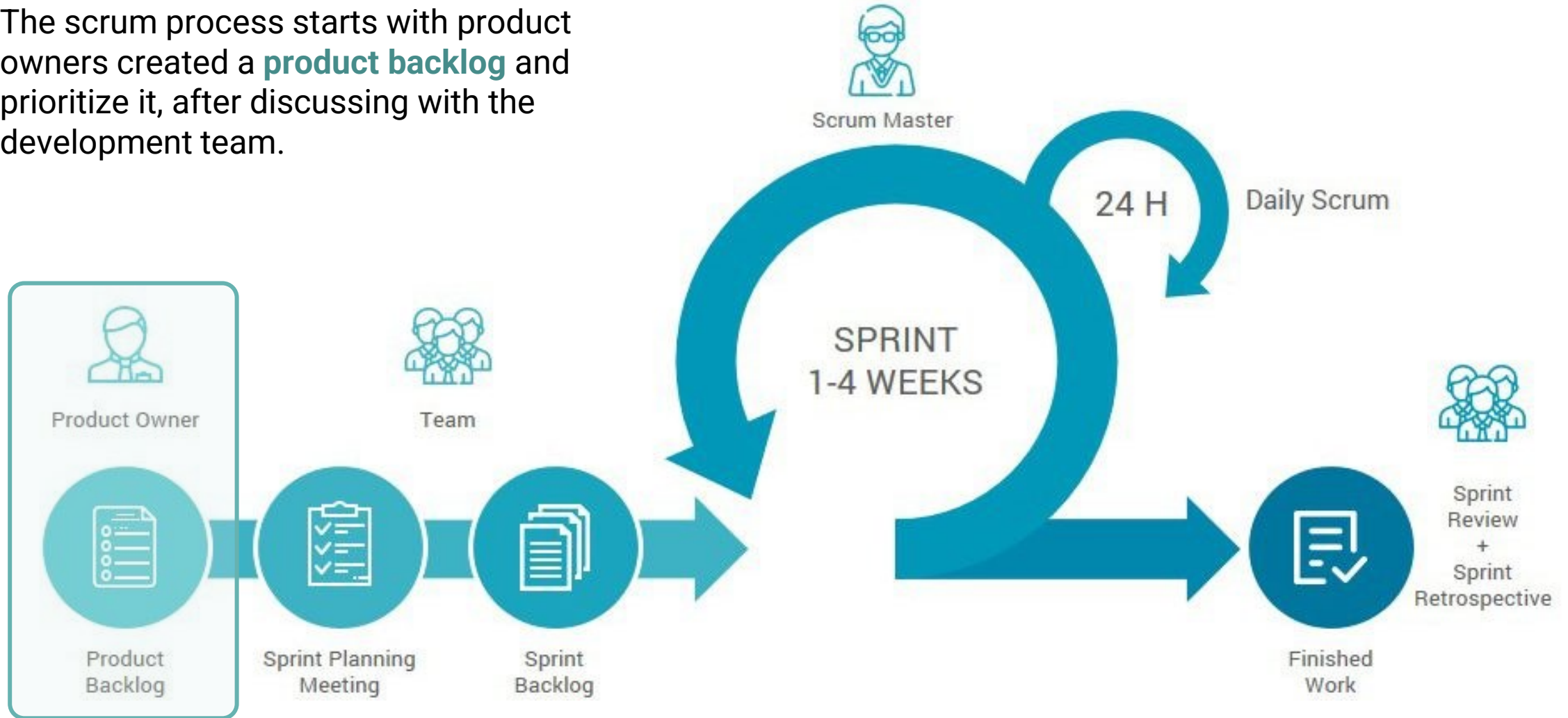
- A short, time-boxed period, typically 1-4 weeks
- The scrum team focuses on delivering a potentially shippable product increment during a sprint

How do people decide what to work on in a SPRINT?



Product Backlog

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





EXAMPLE OF PRODUCT BACKLOG

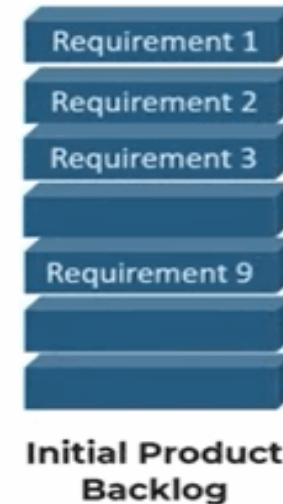
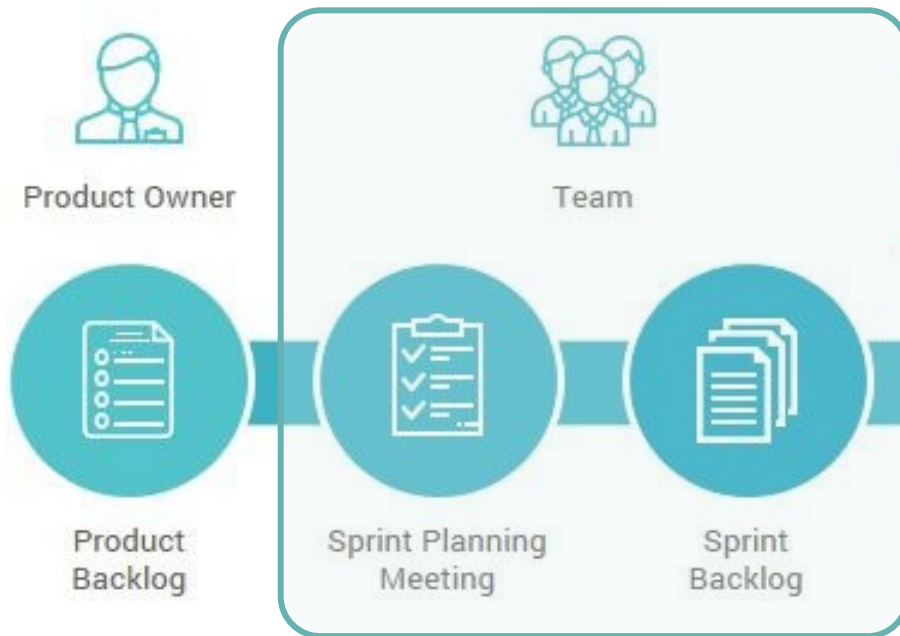
More details on week 3

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	

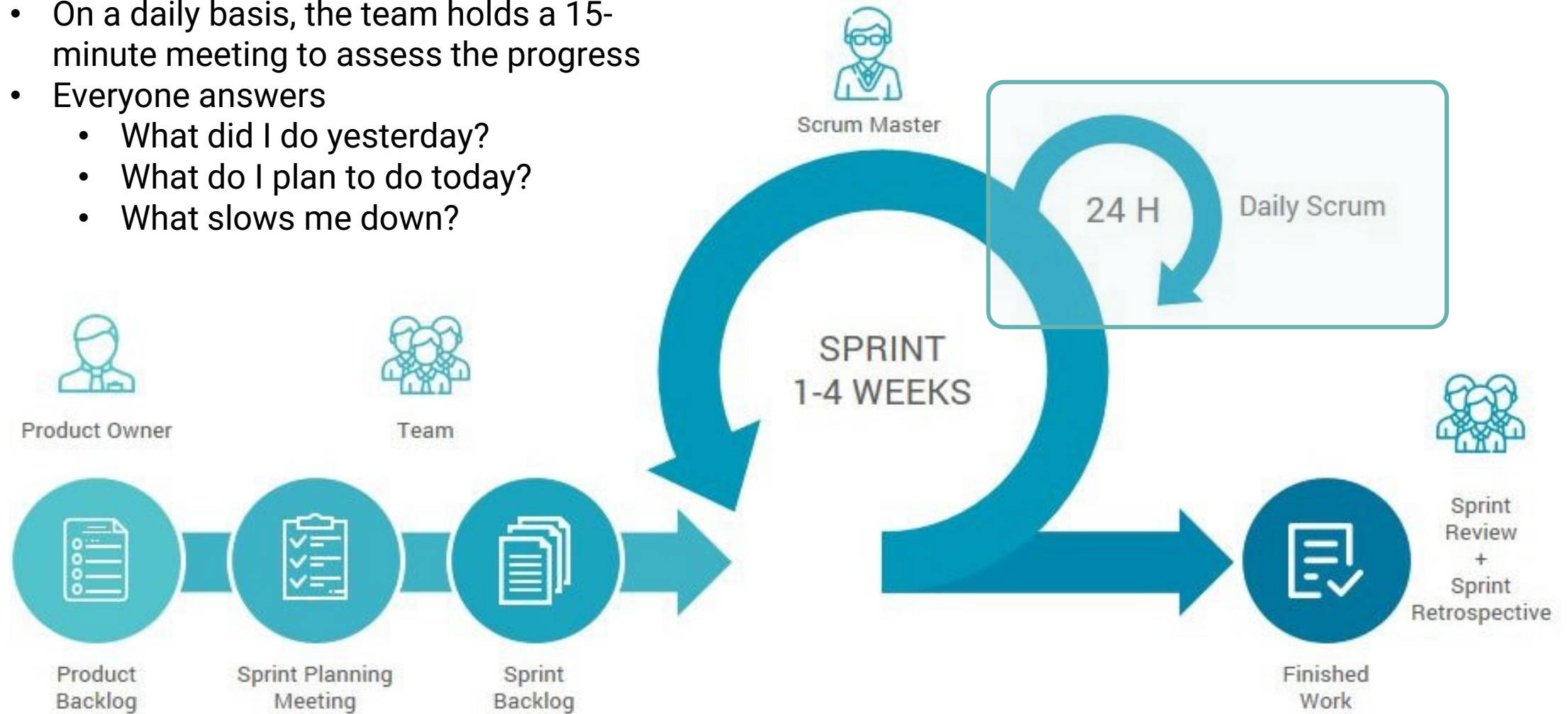
Sprint Backlog

Next, the whole team attend **the Sprint Planning Meeting** to select the items to deliver in the next sprint, which forms the **Sprint Backlog**.



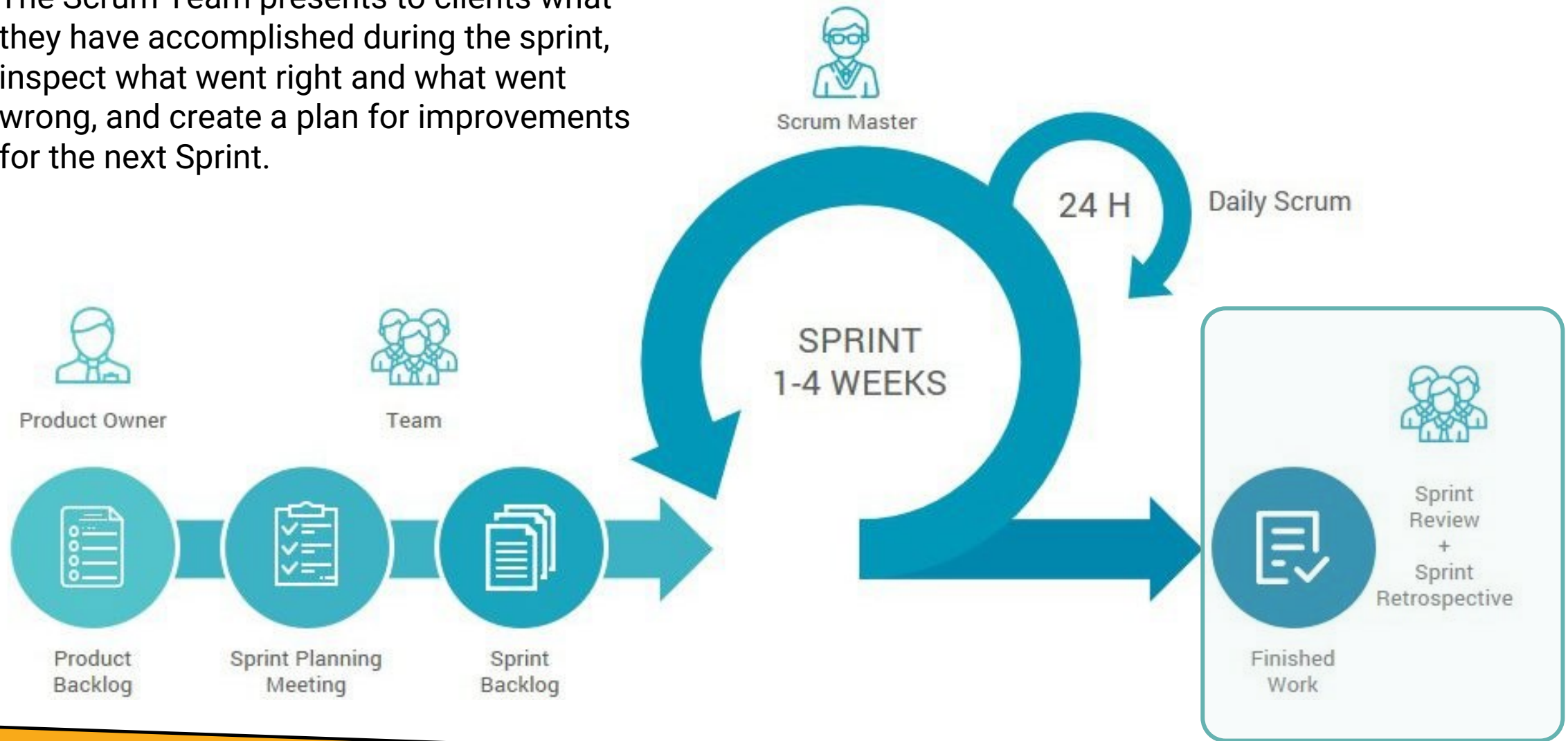
Daily Scrum

- On a daily basis, the team holds a 15-minute meeting to assess the progress
- Everyone answers
 - What did I do yesterday?
 - What do I plan to do today?
 - What slows me down?



Sprint Review & Retrospective

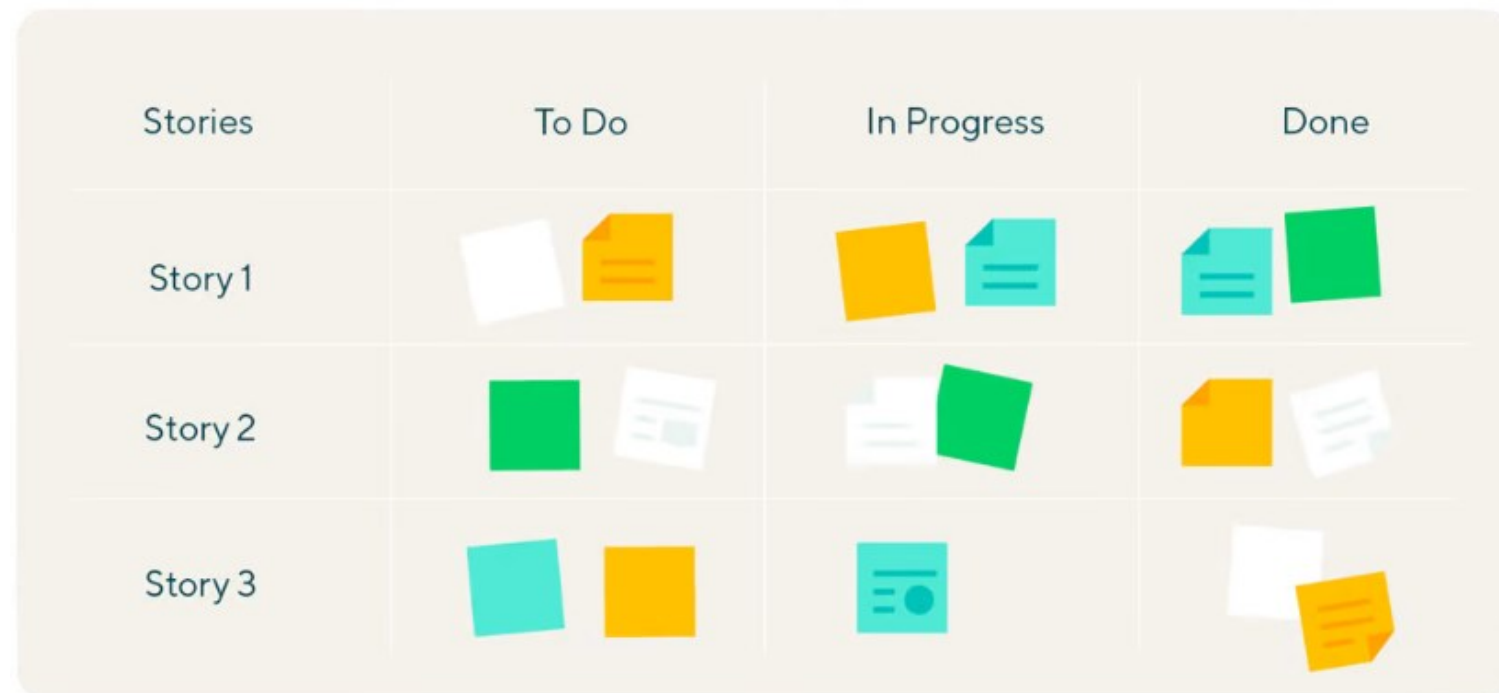
The Scrum Team presents to clients what they have accomplished during the sprint, inspect what went right and what went wrong, and create a plan for improvements for the next Sprint.



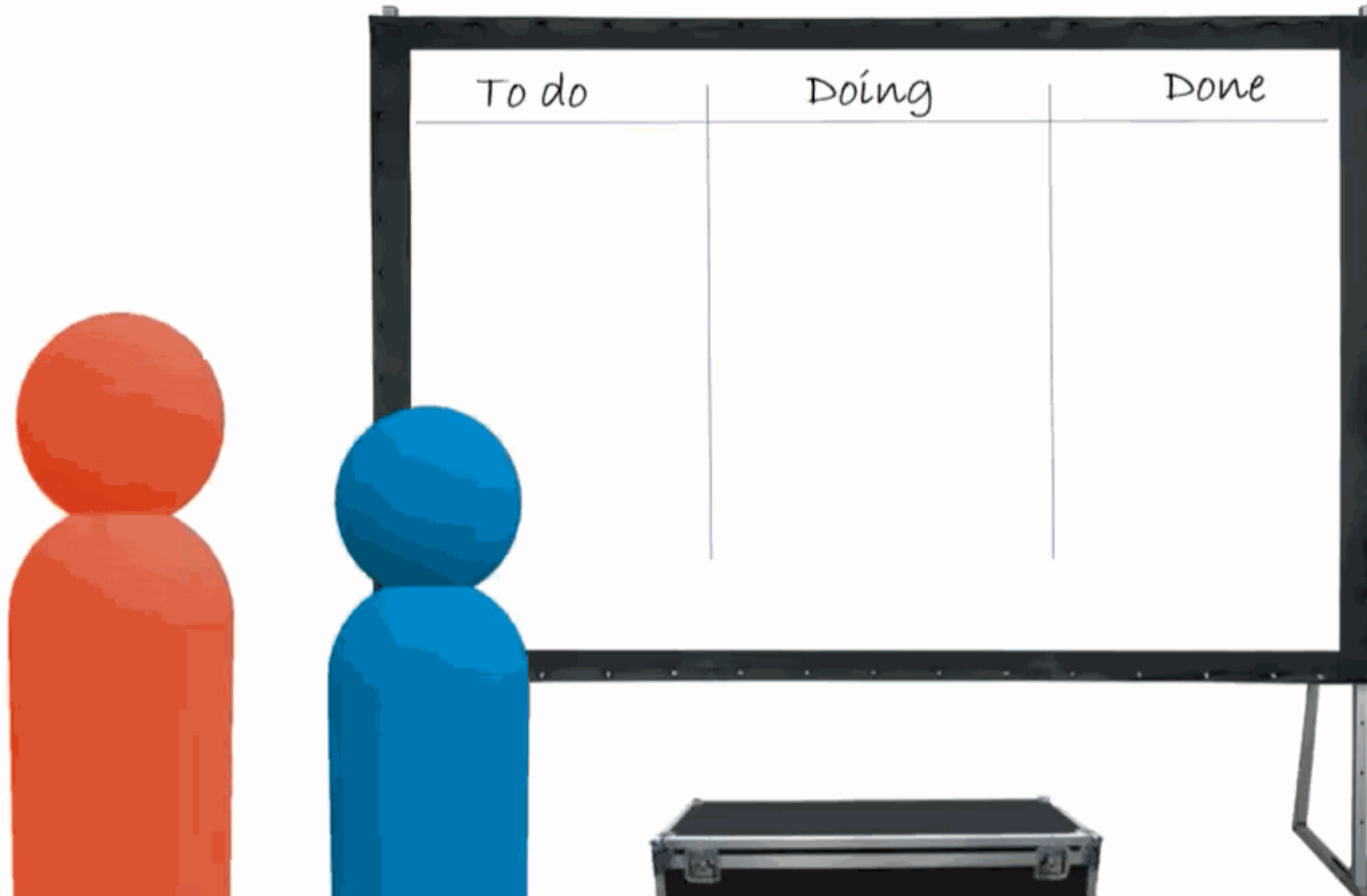


SCRUM BOARDS

- 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.

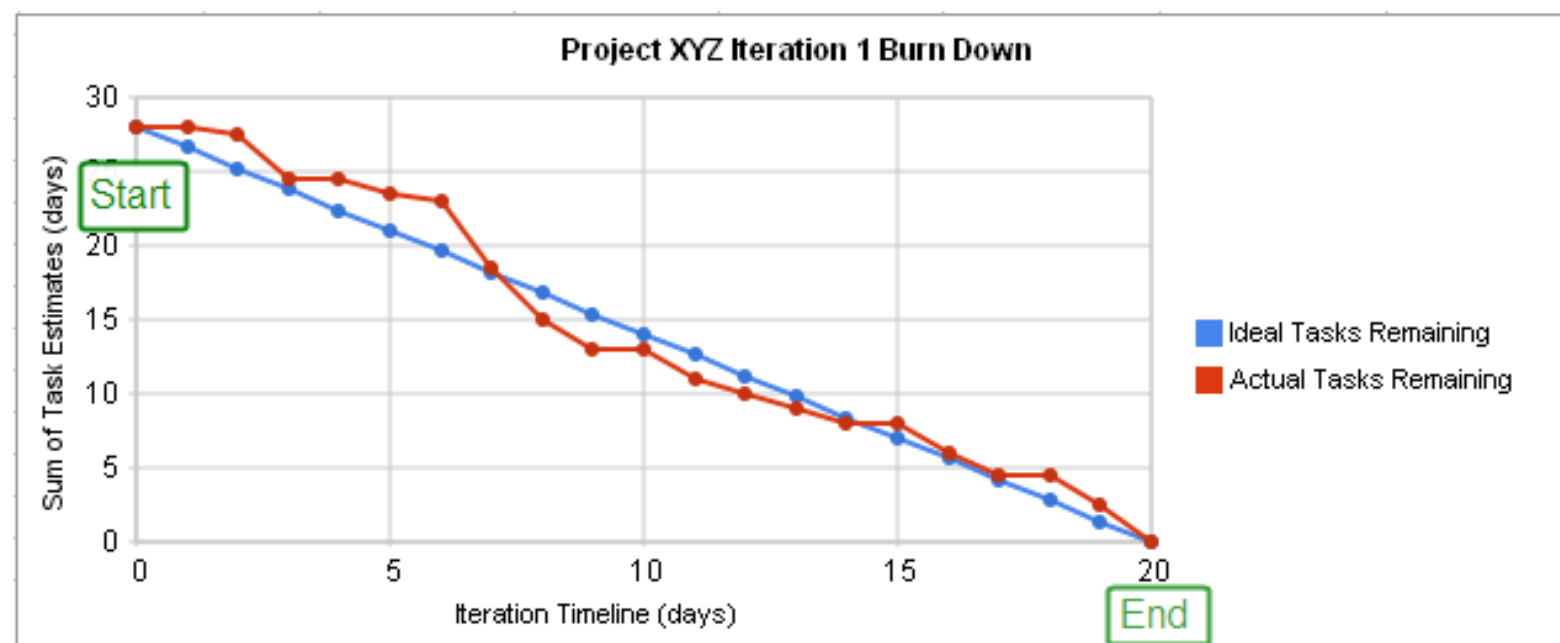


The scrum master



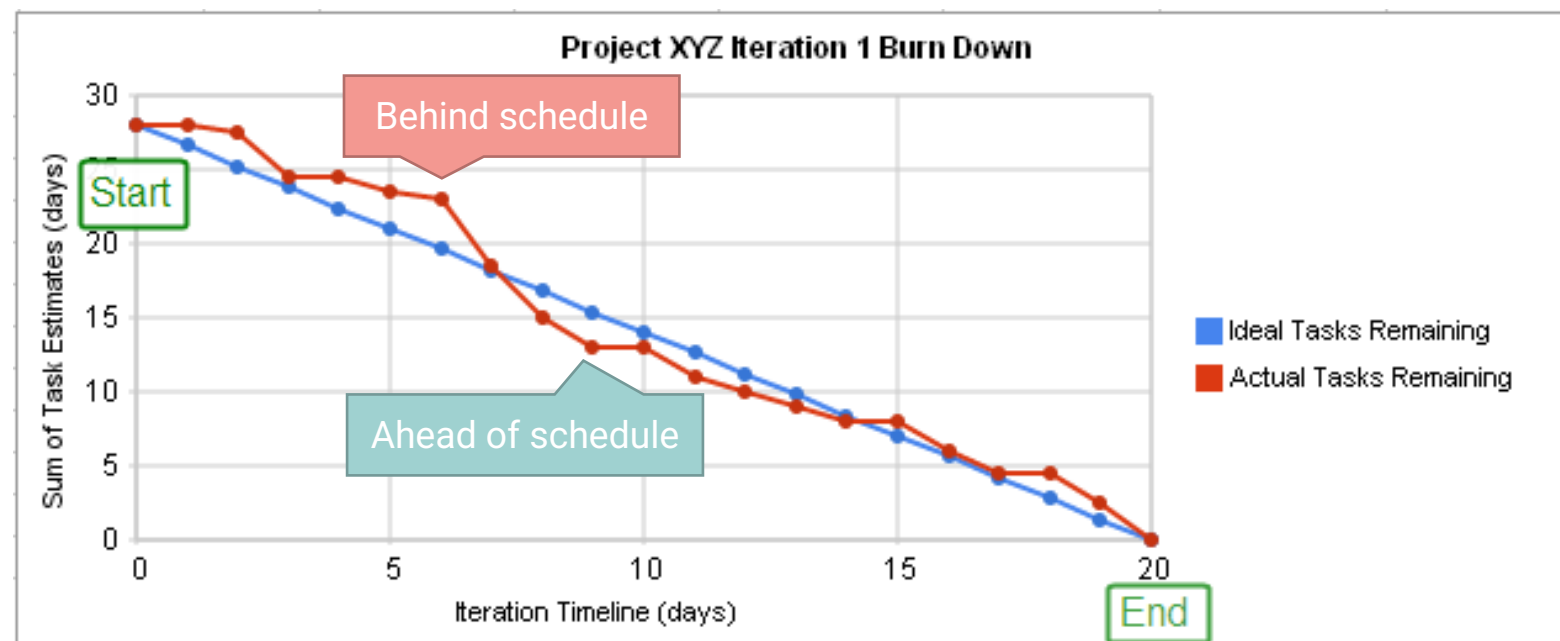
BURNDOWN CHARTS

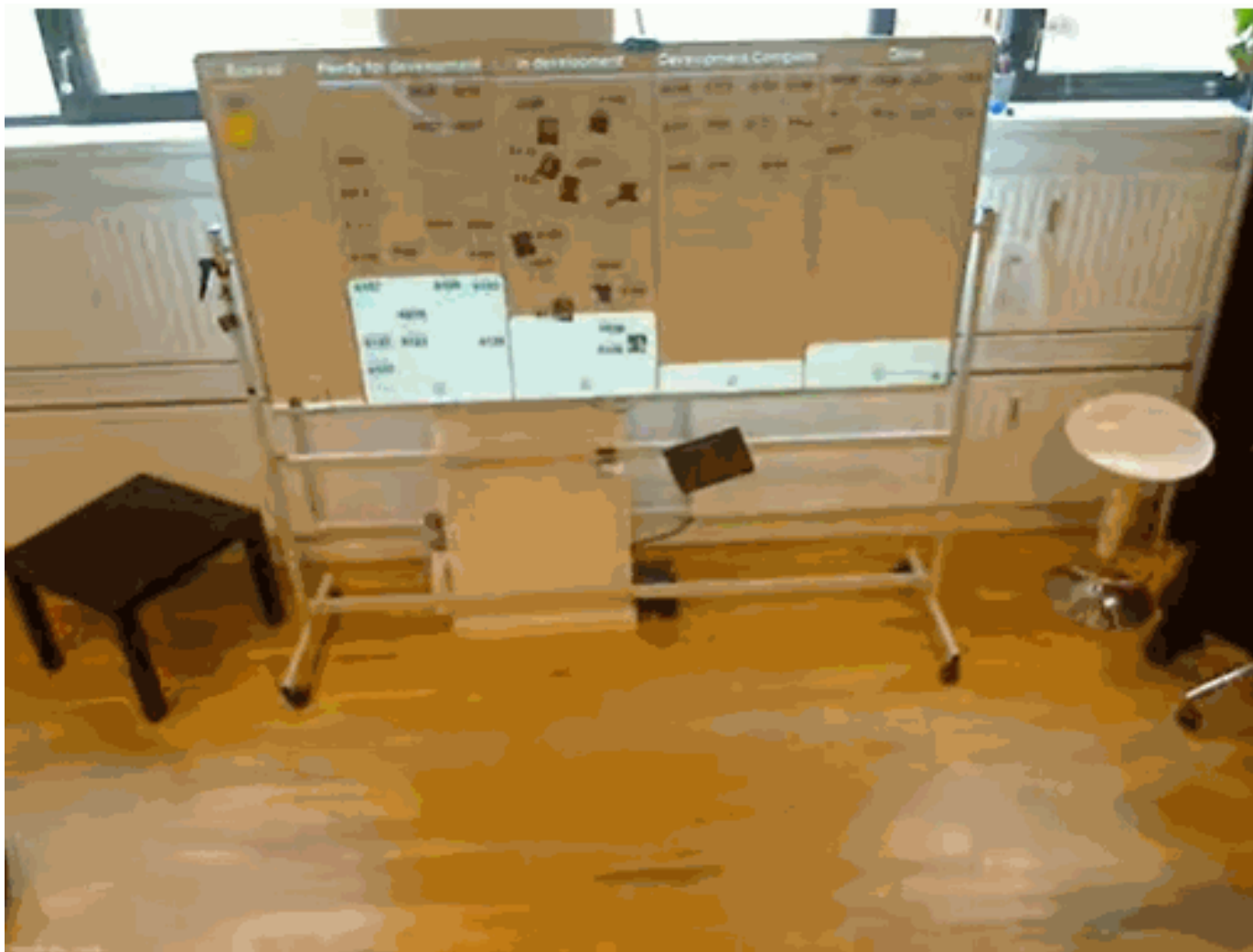
- A graphical representation of work left to do versus time.
- The backlog or effort remaining is often on the Y-axis, with time in the sprint along the X-axis
- Useful for predicting when all of the work will be completed



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

Product Owner
Scrum Master
Scrum Team

Ceremonies

Sprint Planning
Daily Scrum
Sprint Review
Sprint Retrospective

Artifacts

Product Backlog
Sprint Backlog
Burndown Charts

SCRUM CASE STUDY



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



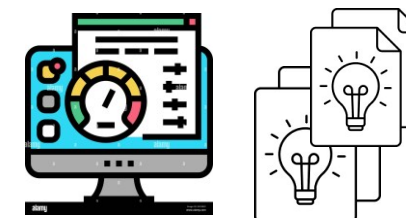
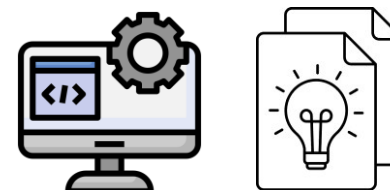
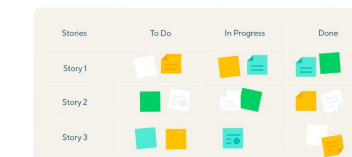
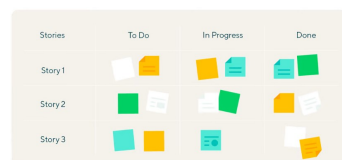
Scrum was adopted by Tesla for their autonomous driving project, which led to a **15%** acceleration of the development cycle and a **10%** decrease in errors.



Cisco switch from waterfall to agile for the billing platform, which led to **40%** decrease in defect rate and **14%** increase in defect removal.



APPLYING SCRUM IN OUR TEAM PROJECT



Week 1
Team up

Week 5
Proposal

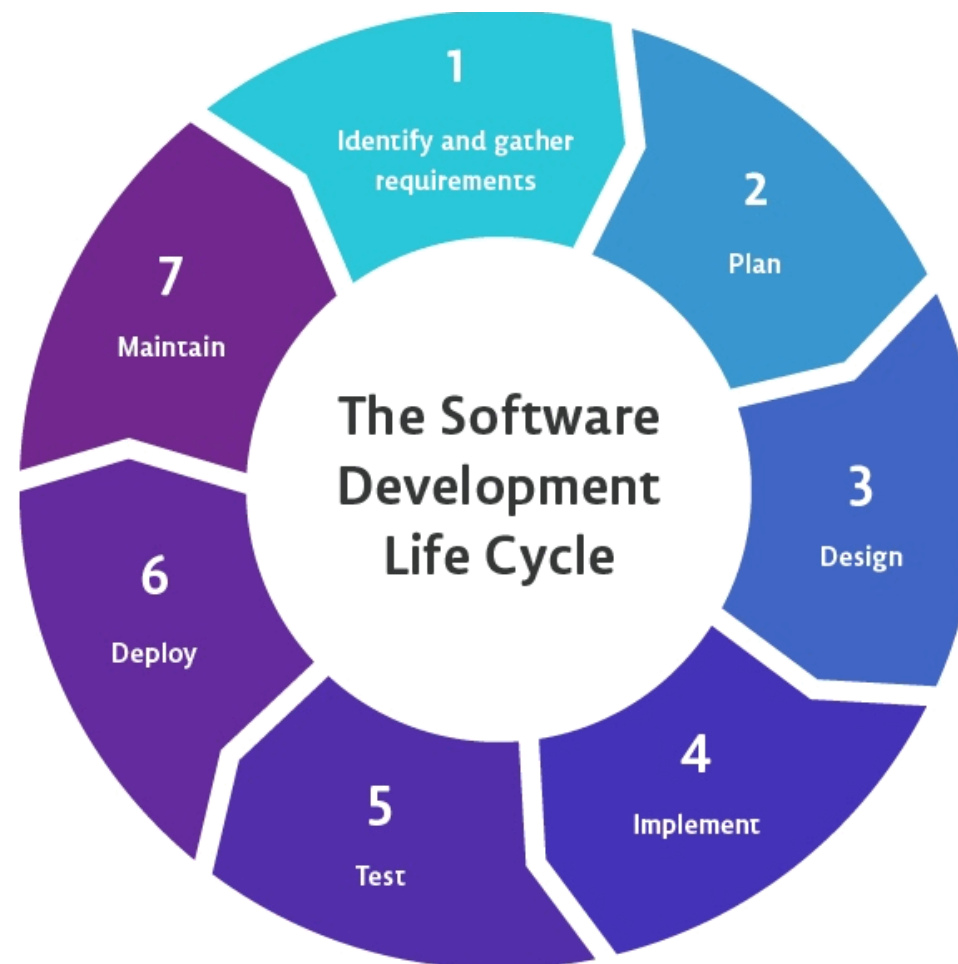
Week 9
Sprint 1

Week 16
Sprint 2



AGILE SEEMS GOOD.

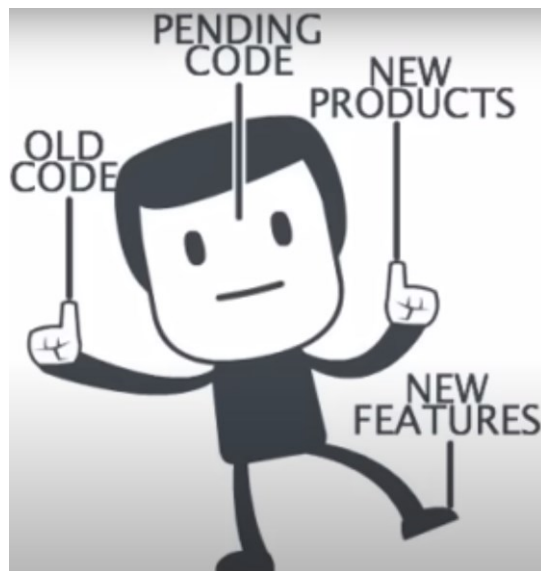
Anything missing?





LECTURE 2

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

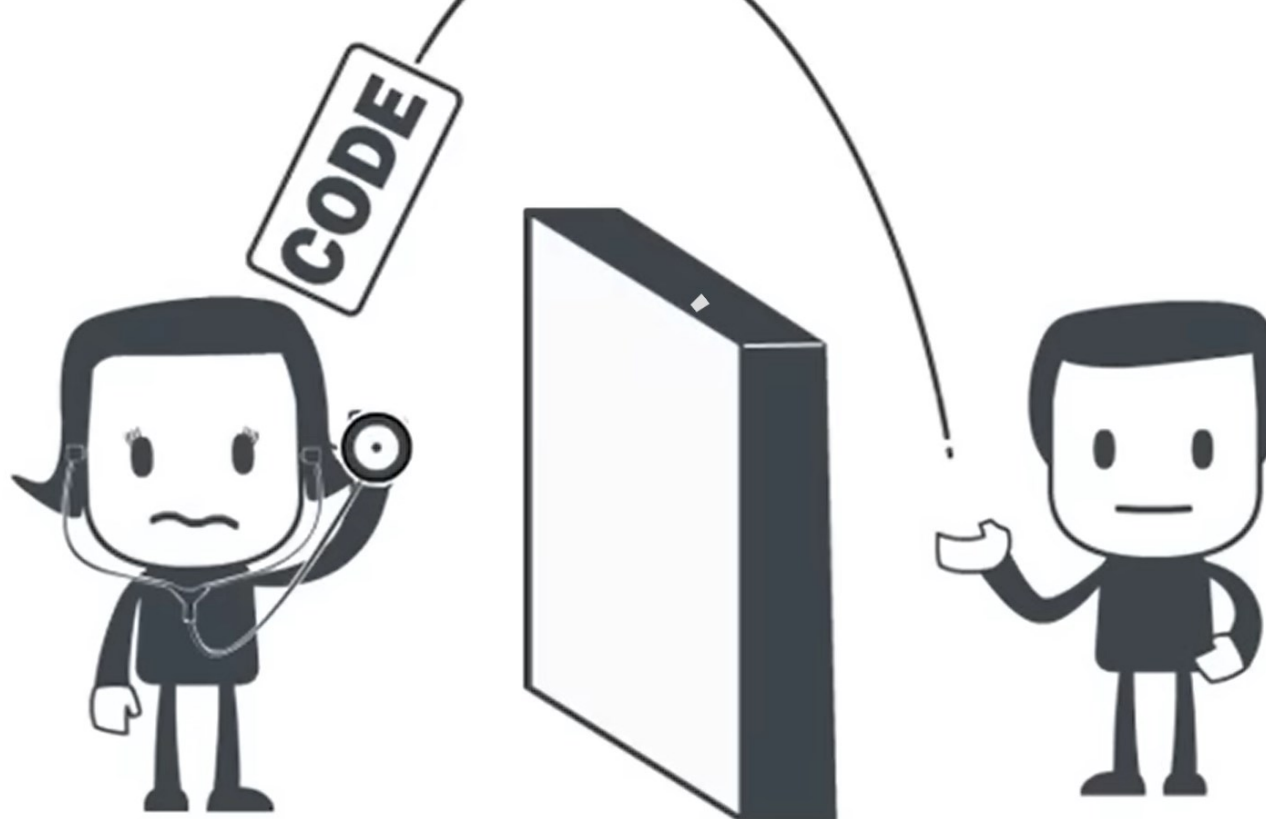


Dev (开发团队) responsibility:

- Design
- Develop
- Test
- Document

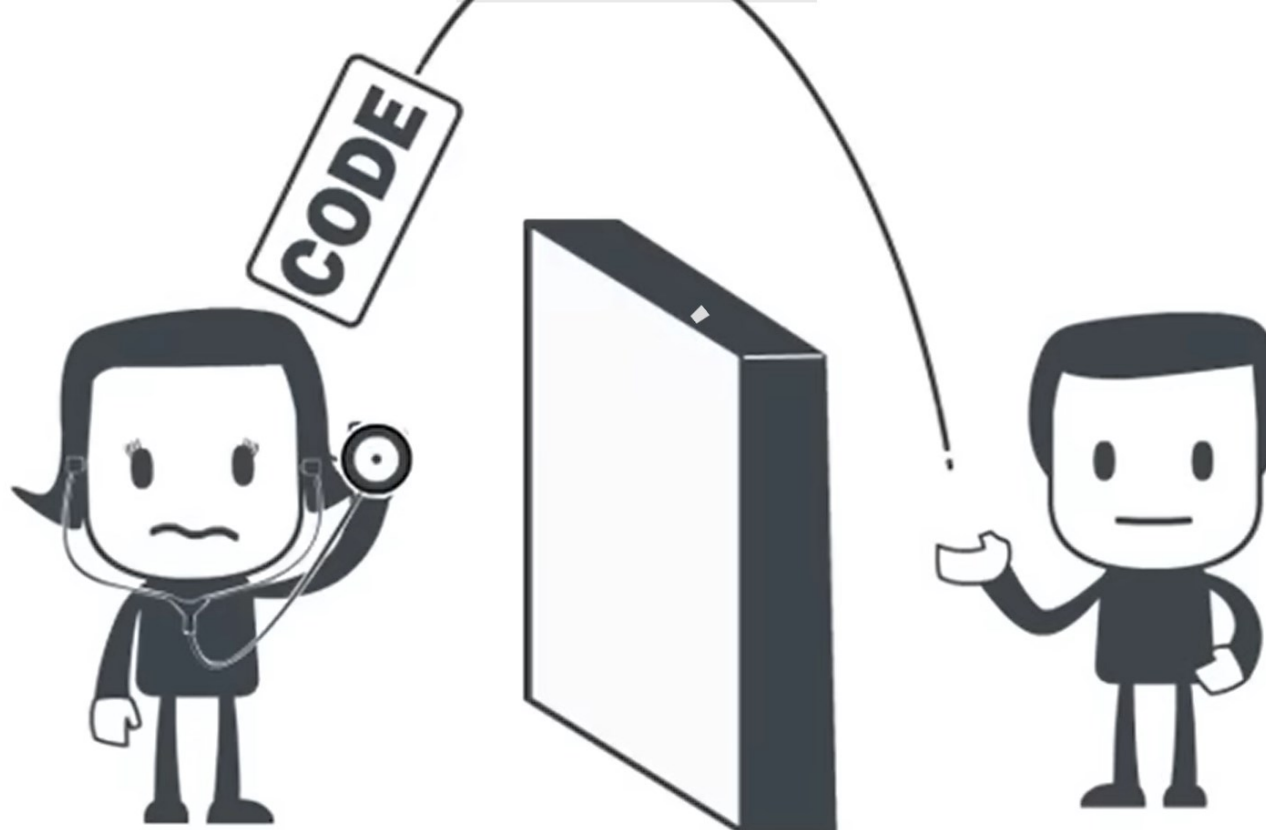
Ops (运维团队) responsibility:

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



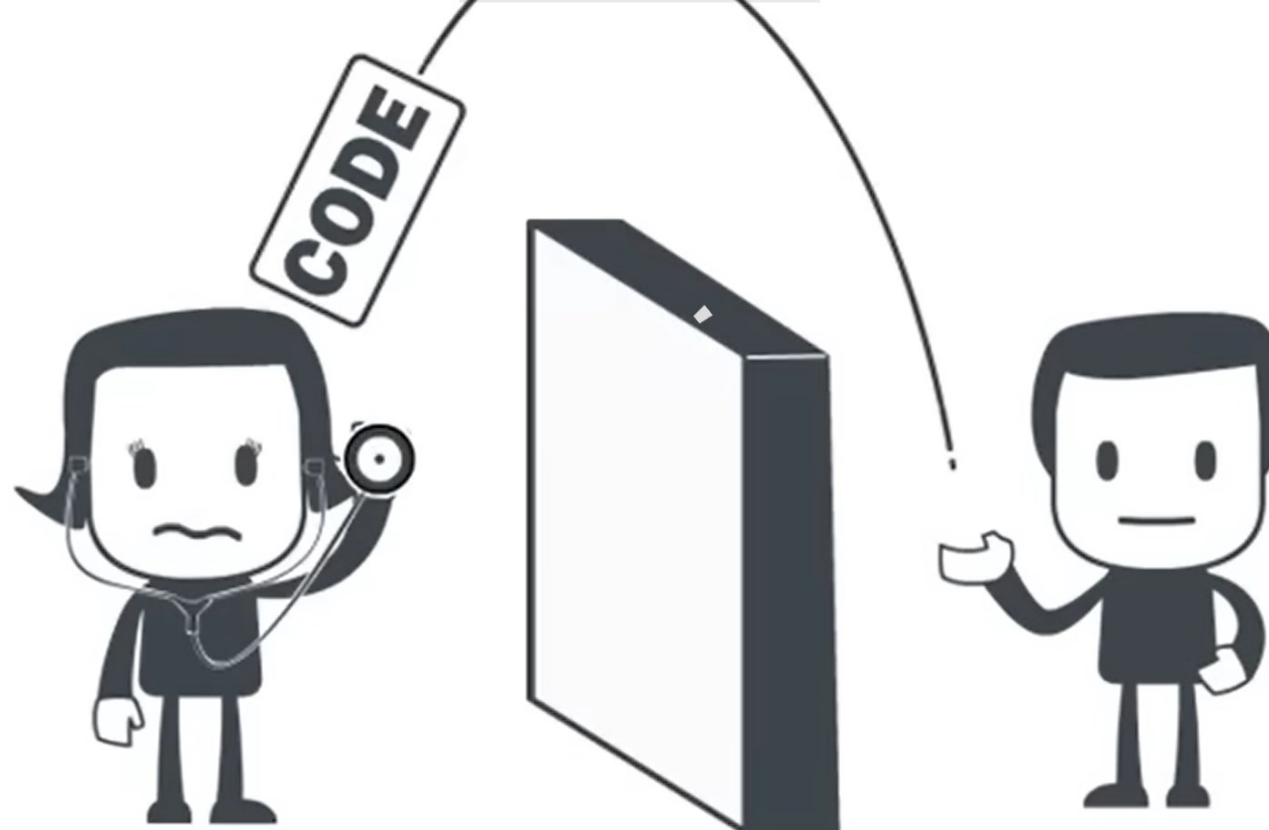
In traditional IT operations, Dev and Ops teams typically work **sequentially** and **individually**

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



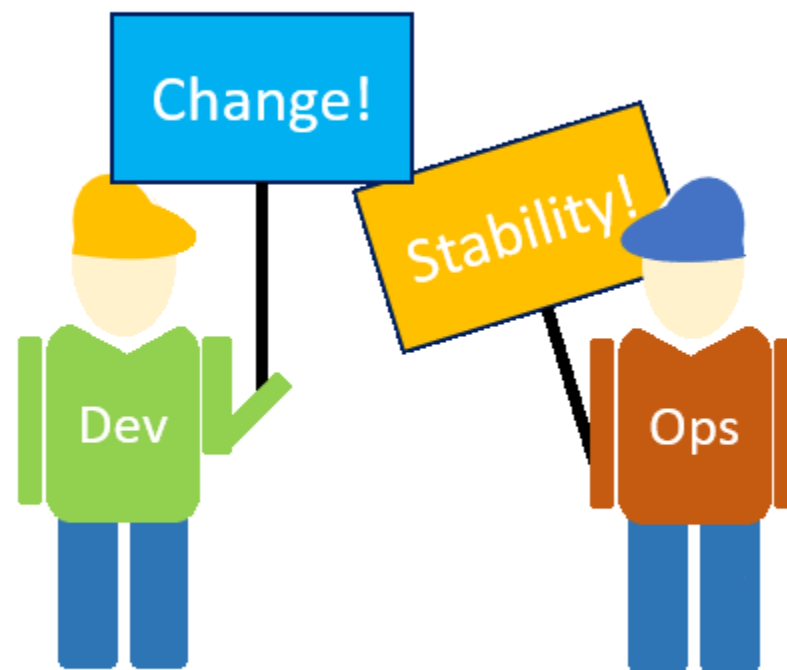
Problem 2: slow feedback, long time to ship

- 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

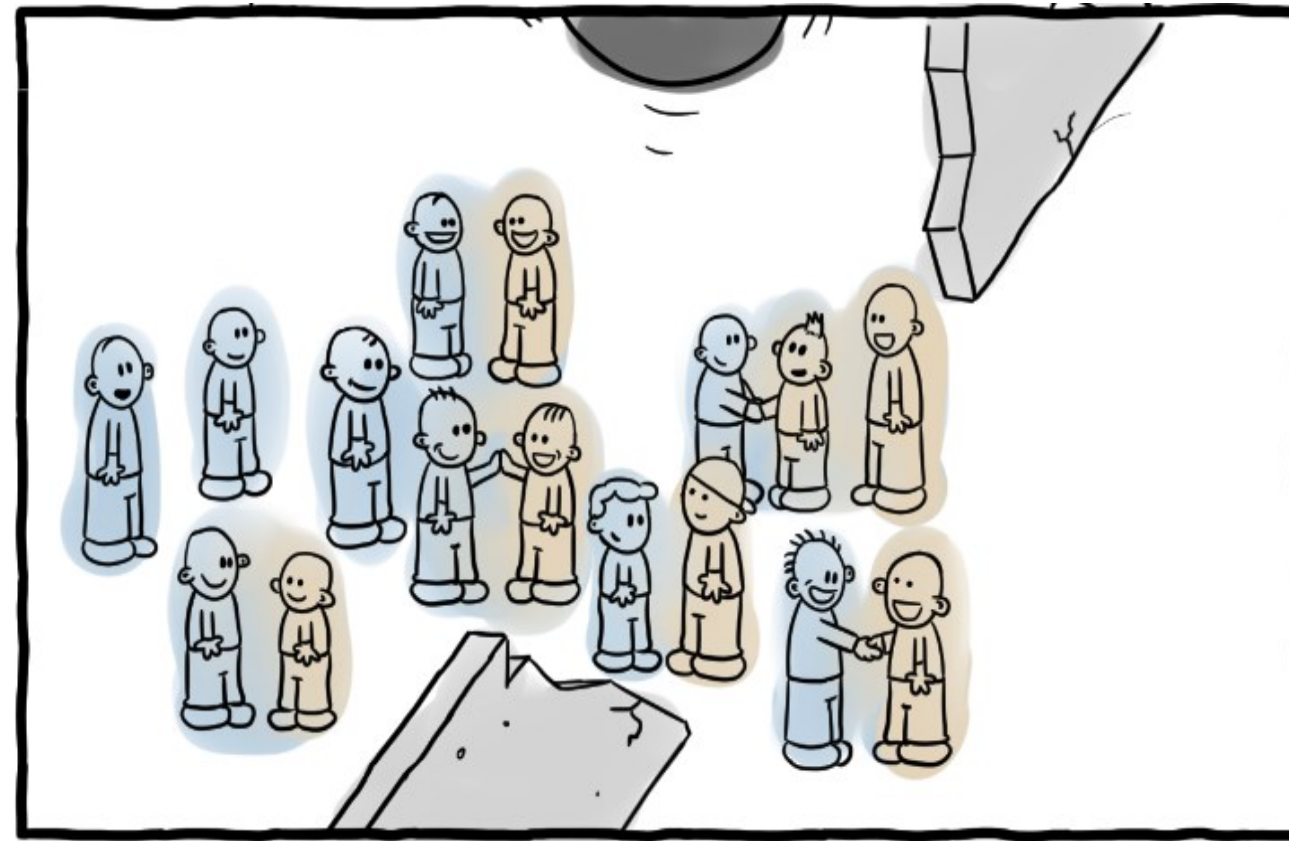
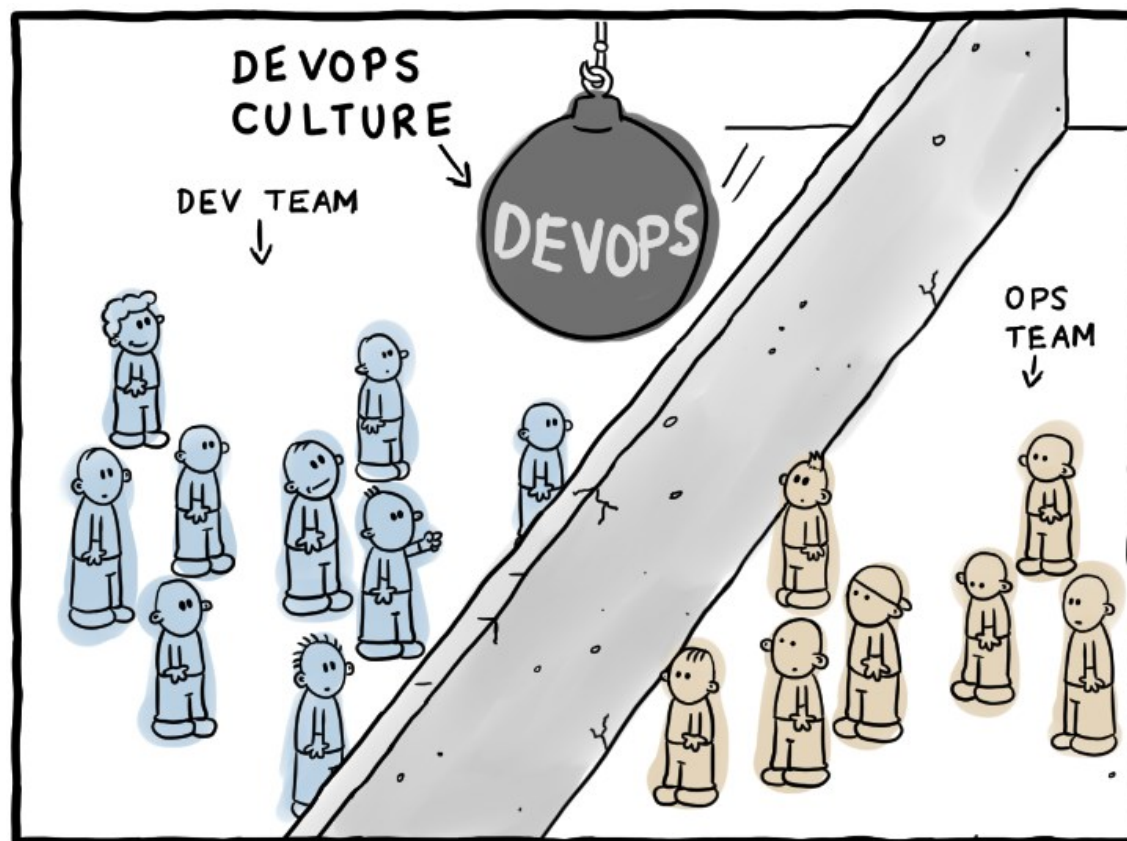
Image source: <https://ralfneubauer.info/devops-gifs-reactions/>



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

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

BEFORE/AFTER DEVOPS



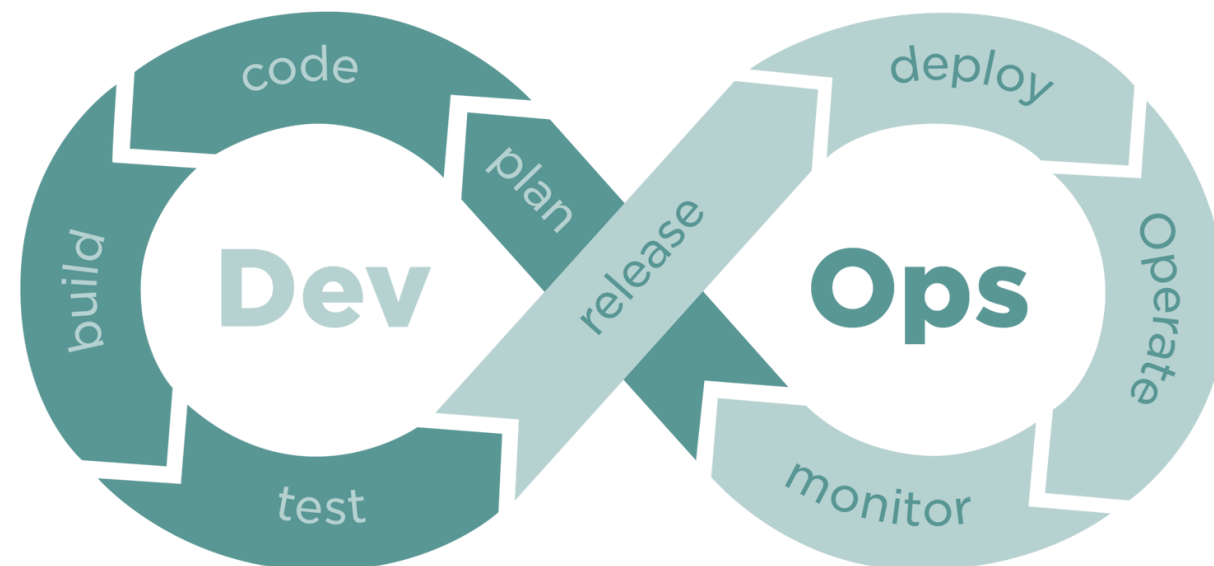
<https://turnoff.us/geek/devops-explained/>

DEVOPS

A culture, practice, and set of tools that integrate development (Dev) and operations (Ops).

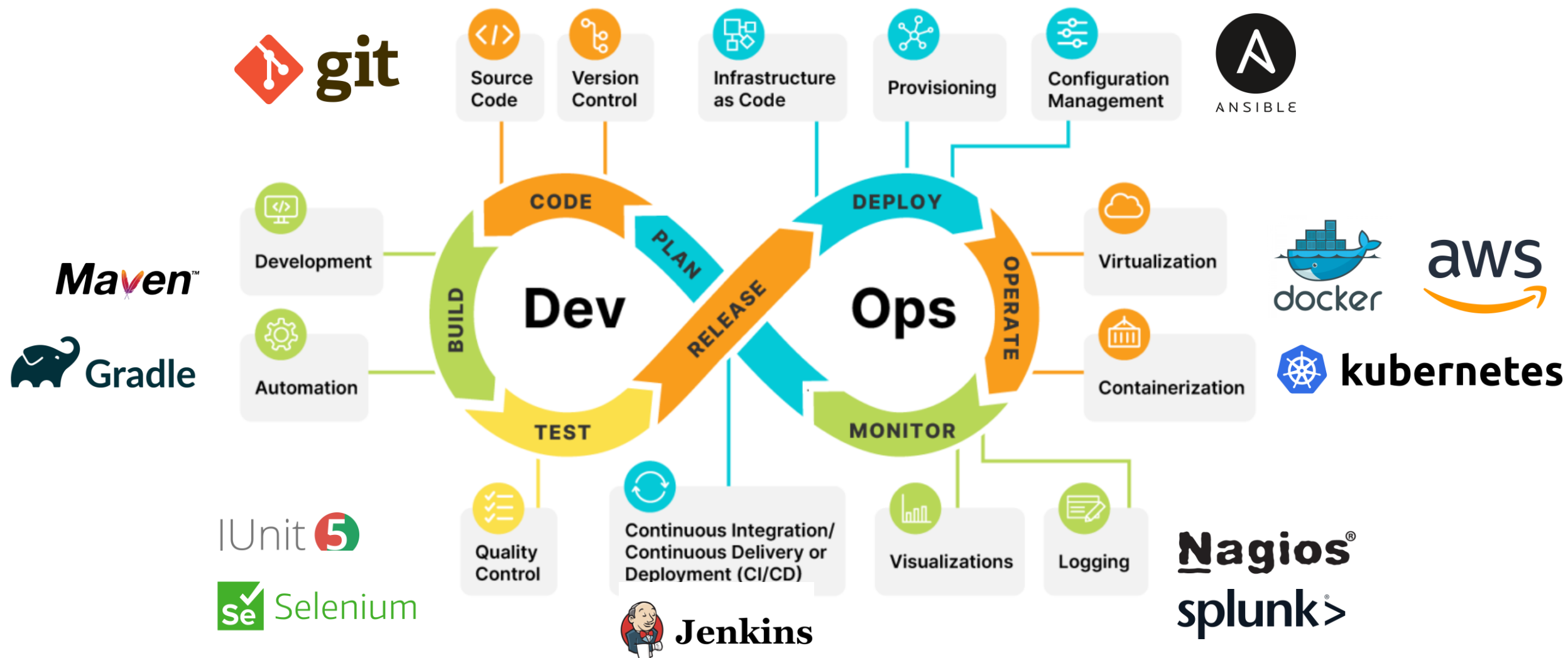
Key principles

- Collaboration: Bridges the gap between developers and IT operations.
- Automation: Automates builds, testing, and deployments.
- Monitoring & Feedback: Uses real-time monitoring to improve system performance and user experience.





DEVOPS PHASES & TOOLS

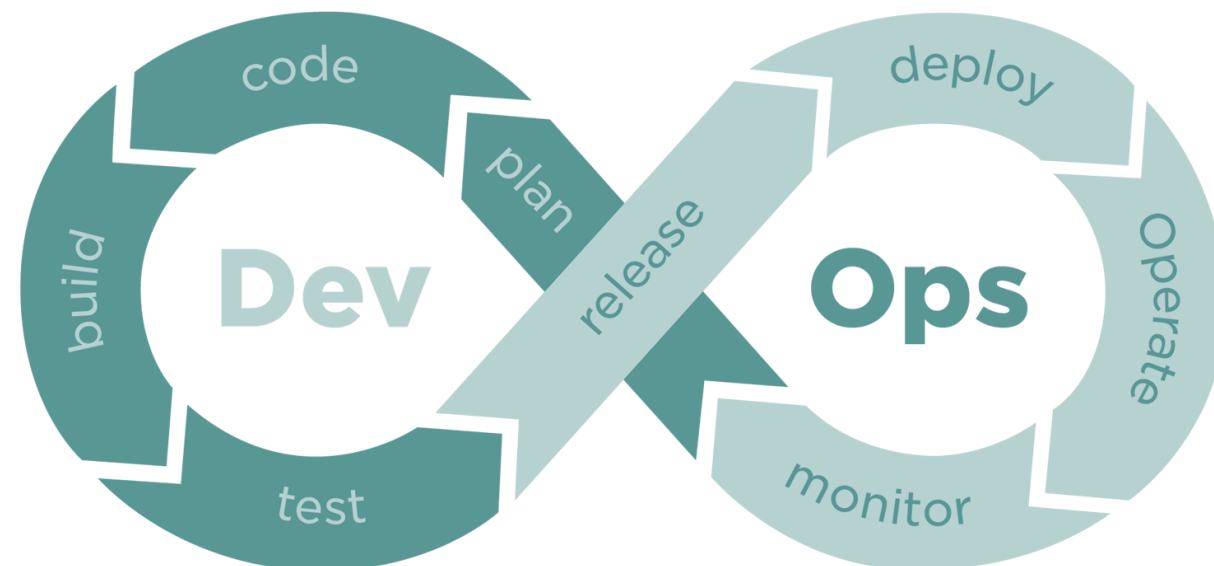


DEVOPS

A culture, practice, and set of tools that integrate development (Dev) and operations (Ops).

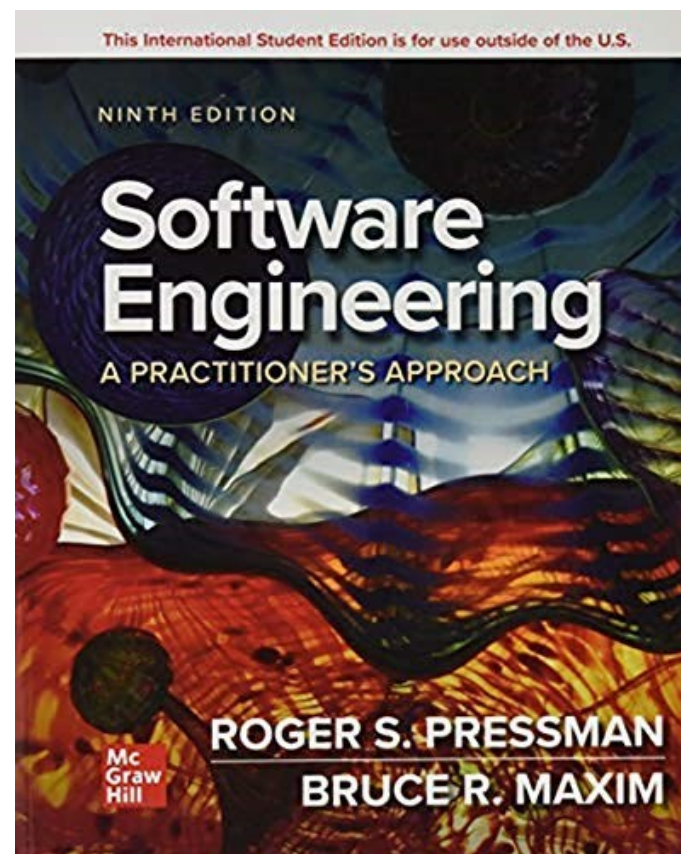
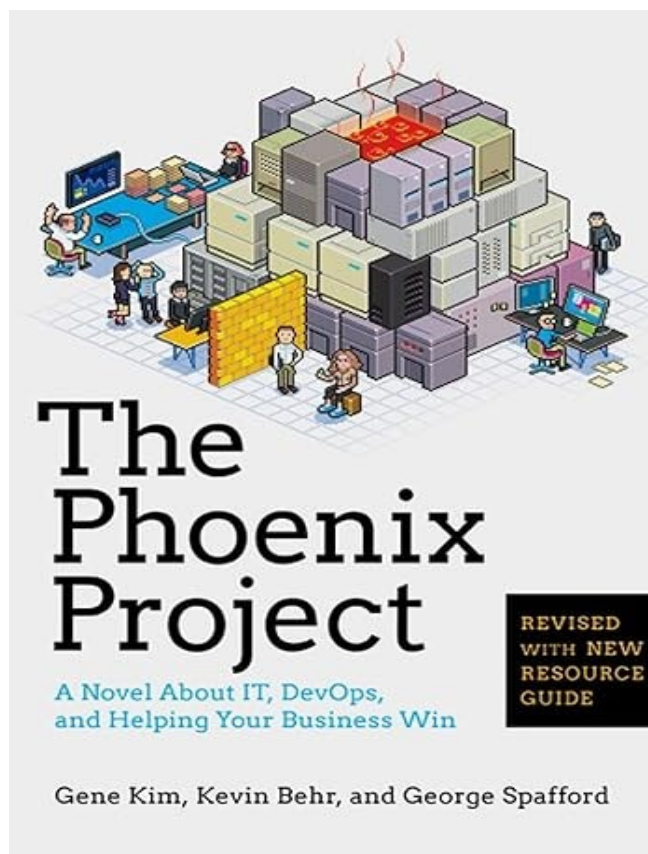
Benefits

- Faster software releases
- Faster feedback
- Improved system reliability and stability
- Reduced deployment failures and rollback times
- Increased efficiency and resource utilization





READINGS



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



NEXT

- Software Requirements