CS3213: Foundations of Software Engineering

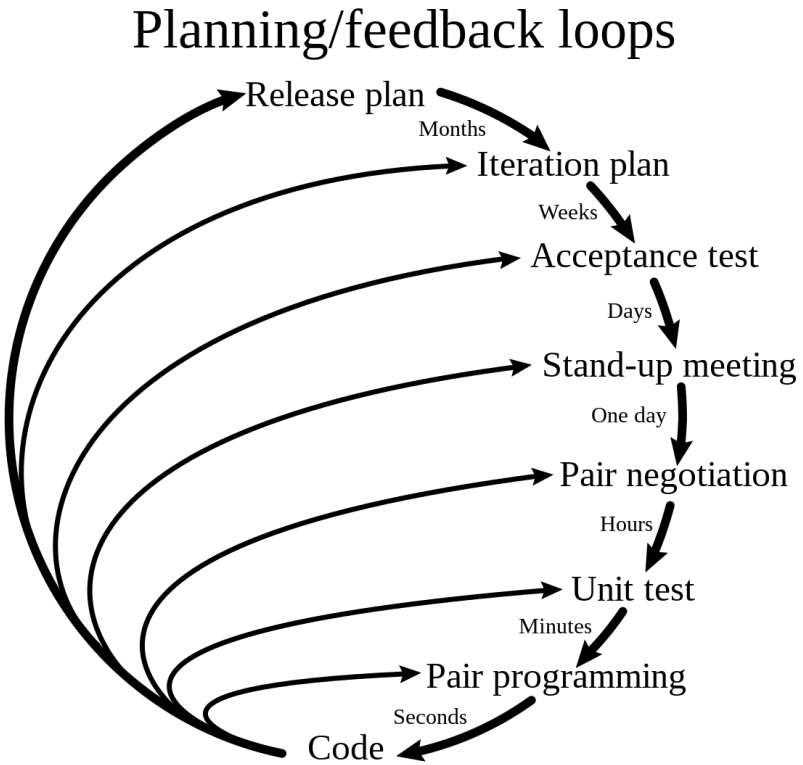
Extreme Programming (XP)

Figure by ChatGPT, voice by https://elevenlabs.io/

Extreme Programming (XP)

Figure by ChatGPT

Extreme Programming



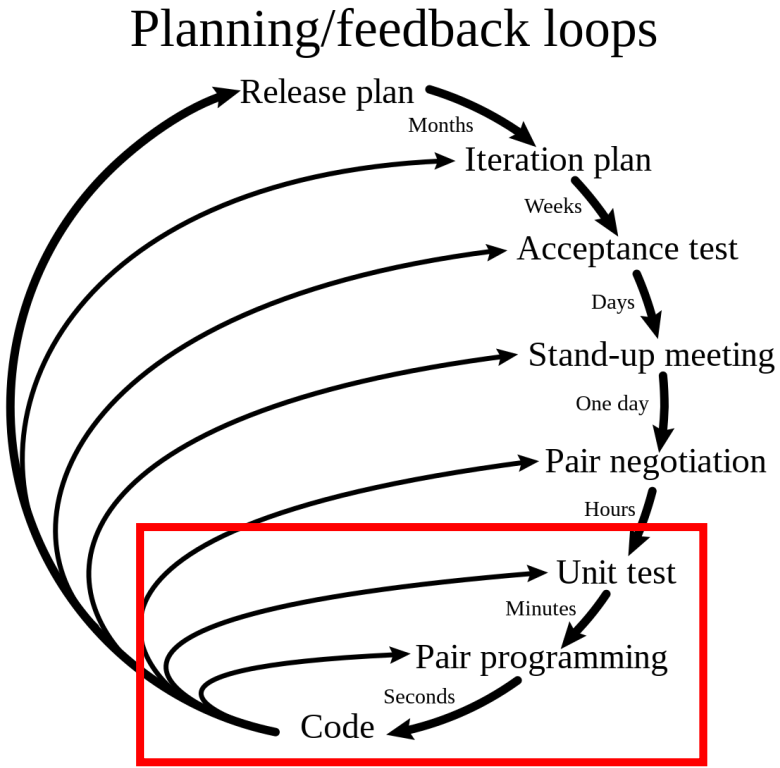
Extreme Programming (XP) takes an "extreme" approach to **iterative development and best practices**.

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

Extreme Programming (XP)

- A very influential agile method, developed in the late 1990s, that introduced a range of agile development techniques
- Software-development focused, in contrast to Scrum and Kanban

Extreme Programming



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

Pair Programming

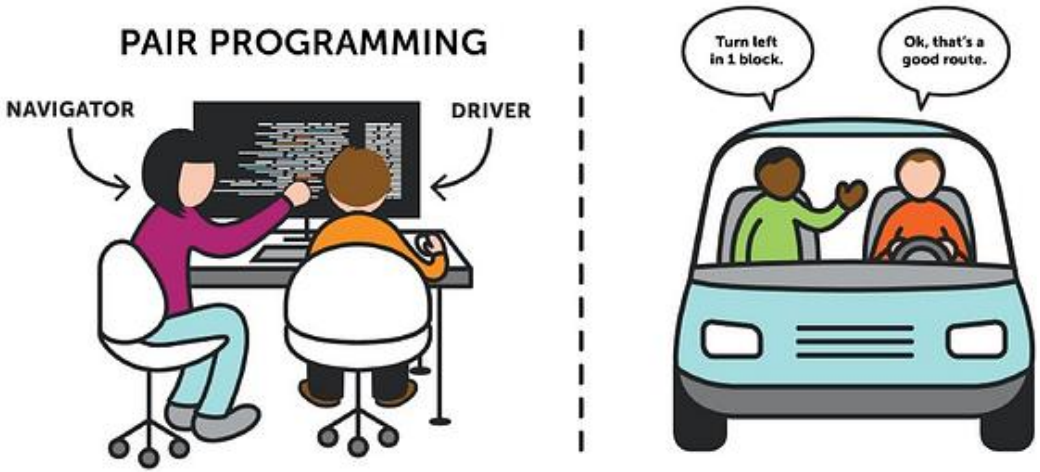
All code is produced by **two developers working together**

https://tuple.app/pair-programming-guide/

Pair Programming at Atlassian

https://www.youtube.com/watch?v=fQ-x-T34z9w

Extreme Programming (XP)



The Social Dynamics of Pair Programming,
https://ieeexplore.ieee.org/document/4222597

https://miro.medium.com/v2/resize:fit:4800/format:webp/1*U7QCSsoFvZs8lU1myrFpJw.png

Pair Programming at Microsoft (2008)

- 22% of engineers use(d) pair programming

Table 1: Pair Programming benefits

1. Fewer Bugs	66
2. Spreads Code Understanding	42
3. Higher Quality Code	48
4. Can Learn from Partner	42
5. Better Design	30
6. Constant Code Reviews	22
6. Two Heads are Better than One	22
8. Creativity and Brainstorming	17
9. Better Testing and Debugging	14
10. Improved Morale	13

Pair programming: what's in it for me?
https://dl.acm.org/doi/abs/10.1145/1414004.1414026

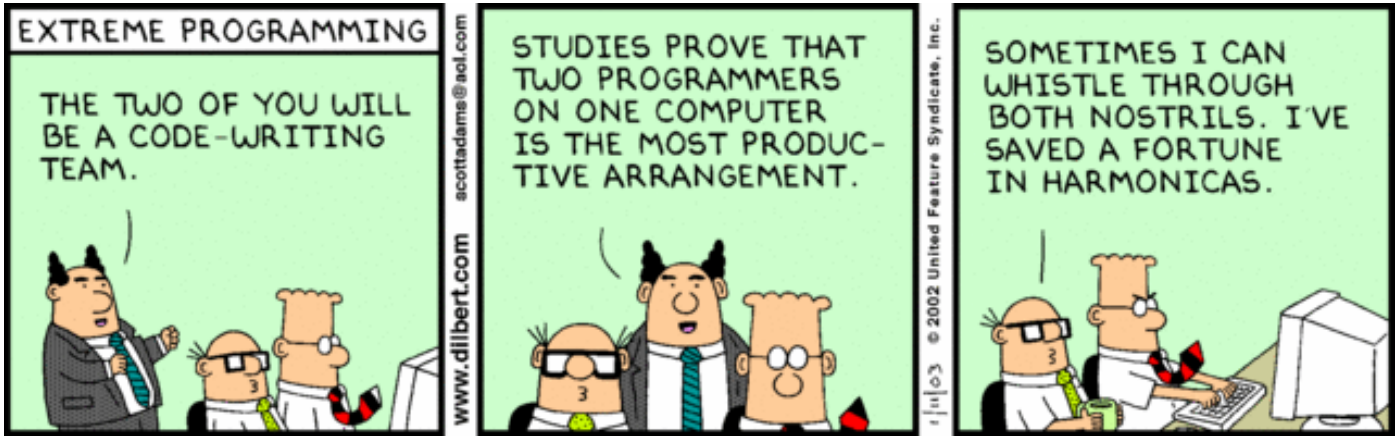
Pair Programming at Microsoft (2008)

Table 2: Pair programming problems

1. Cost efficiency	79
2. Scheduling	31
3. Personality clash	25
4. Disagreements	24
5. Skill differences	22
6. Programming style differences	13
7. Hard to find a partner	12
8. Personal style differences	11
9. Distractions	10
10. Misanthropy	9
10. Bad Communication	9
10. Metrics/Hard to Reward Talent	9

Pair programming: what's in it for me?
https://dl.acm.org/doi/abs/10.1145/1414004.1414026

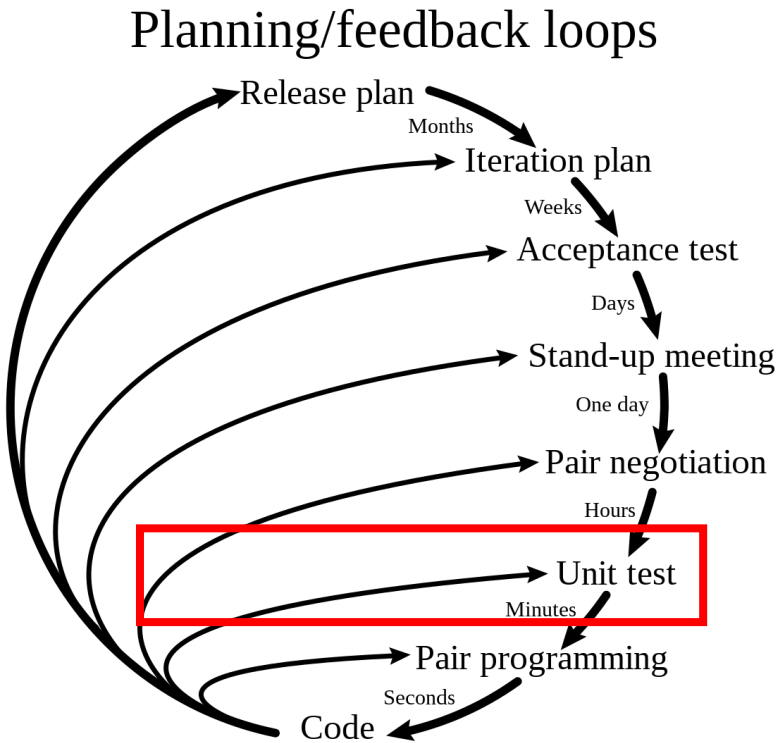
Pair Programming: Example of Challenge



Collective Code Ownership

- Any programmer can change any code in the project
- Everyone is responsible for all the code

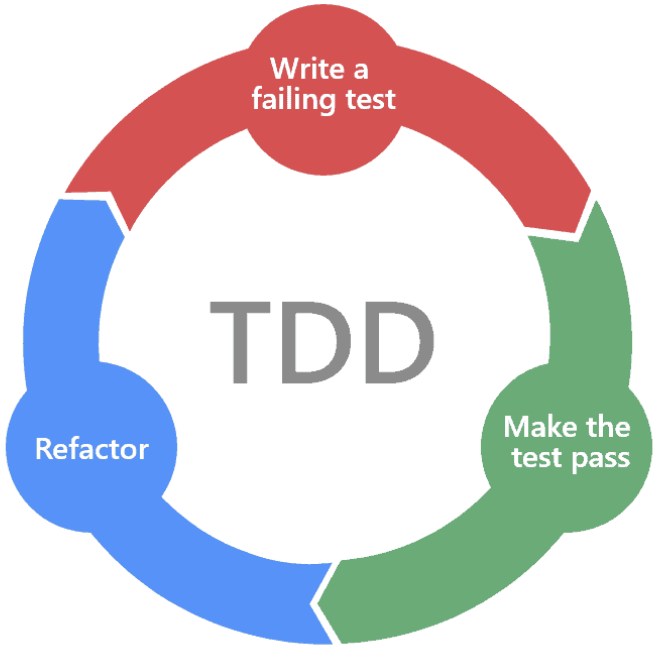
Extreme Programming



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

Unit Tests

- XP proposed *test-driven development*
- XP suggests to test for all potential cases where the code could fail
- Various advantages
  - Helps development, which can address a test case at a time
  - Facilitates other practices like Continuous Integration, Collective Code Ownership, Refactoring, …



https://marsner.com/blog/why-test-driven-development-tdd/

Acceptance Tests

- Acceptance tests are derived from the user requirements
- Written together with the customer (as *user stories*)

Continuous Integration

- Development team should always work on the latest version of the software
- Code should be integrated into the main branch of the project frequently
- Facilitated by unit tests
  - Today, often automatically run in frameworks such as GitHub Actions or Jenkins

Simple Design & Refactoring

- XP mandates a “simple is best” mentality
- Both in terms of system design and code that is written
- **Refactoring** as part of the lifecycle of a project to keep the project maintainable and easier to extend

Other XP Practices

- **On-site customer**: A representative of the end-user of the system (the customer) should be available full time for the use of the XP team.
- **Sustainable pace**: Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity

XP and Scrum

- Both XP and Scrum are incremental and feedback-driven
- Key difference is that Scrum focuses on **agile management mechanisms**, while XP focuses on **software engineering practices**
- Various practices of XP have been adopted in other agile, as well as plan-driven approaches (e.g., continuous integration, pair programming, test-driven development, …)

Summary and Key Points

- XP, unlike Scrum or Kanban, established agile software development practices
- Key practices include pair programming, collective code ownership, continuous integration, test driven development, as well as simple design and refactoring

YouTube

https://www.youtube.com/watch?v=Saaz6D1azlU