OSE - Agile week 2

Keywords:

Iterative; Incremental; Customer Collaboration; Cross-functional Teams; Adaptive Planning;

Continuous Feedback; Timeboxing; Emphasis on Working Software; Self-Organizing Teams;

Continuous Improvement

Agile development is a software development methodology that emphasizes **iterative** and **incremental** development, teamwork, and customer satisfaction. This approach to software development is based on the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change. Agile development is a flexible and adaptive approach that allows teams to deliver functional software faster and with higher quality.

The Agile methodology is characterized by short development cycles known as sprints, which typically last two to four weeks. The focus of each sprint is to deliver a working part of the software that can be reviewed by the customer and evaluated by the development team. The Agile methodology requires close collaboration between the development team and the customer, with regular feedback and communication throughout the development process.

Agile development relies on a number of practices and tools to support the development process. These include

- daily stand-up meetings,
- sprint planning sessions,
- sprint reviews and retrospectives,
- continuous integration and testing.

Agile development also emphasizes the importance of self-organizing teams, where individuals work together to achieve common goals and take ownership of their work.

Succinct Manifesto

Agile Manifesto

The four values of the Agile Manifesto are:

1. Individuals and interactions over processes and tools: This value emphasizes the importance of people in the development process, and the need for open communication and collaboration between team members. 个人和互动高于过程和工具。这一价值观强调了人在开发过程中的重要性，以及团队成员之间公开沟通和协作的必要性。
2. Working software over comprehensive documentation: This value emphasizes the importance of producing working software, rather than focusing on extensive documentation or planning. 有效的软件胜过全面的记录。这一价值观强调了生产工作软件的重要性，而不是专注于广泛的文档或计划。
3. Customer collaboration over contract negotiation: This value emphasizes the importance of working closely with customers or stakeholders to understand their needs and requirements, rather than relying on contracts or specifications. 客户合作高于合同谈判。这一价值观强调了与客户或利益相关者密切合作以了解他们的需求和要求的重要性，而不是依赖合同或规范。
4. Responding to change over following a plan: This value emphasizes the importance of being flexible and responsive to changing requirements or circumstances, rather than following a rigid plan. 应对变化而不是遵循计划。这一价值观强调了灵活应对不断变化的要求或环境的重要性，而不是遵循一个僵硬的计划。

The twelve principles of the Agile Manifesto are:

1. Customer satisfaction through early and continuous software delivery. 顾客满意度
2. Embrace changing requirements, even late in development. 积极应对随时改变的需求
3. Deliver working software frequently, with a preference for shorter timescales. 频繁提交软件

4. Collaborate closely with customers and stakeholders throughout the development process.在整个开发过程中与客户和利益相关者密切协作。
5. Build projects around motivated individuals and give them the support they need. 围绕积极的个人建立项目，并给予他们所需的支持。
6. Use face-to-face communication as much as possible. 大家面对面的沟通 stand-up meeting
7. Measure progress primarily through working software. 主要通过工作软件来衡量进展。
8. Maintain a sustainable pace of work for the development team. 为开发团队保持可持续的工作节奏。
9. Focus on technical excellence and good design. 注重技术的卓越性和良好的设计。
10. Keep things simple and minimize unnecessary complexity. 保持事情简单，尽量减少不必要的复杂性。
11. Allow self-organizing teams to make decisions about how they work. 允许自我组织的团队对他们的工作方式做出决定。
12. Reflect regularly on the development process and look for opportunities to improve. 定期对发展过程进行反思，寻找改进的机会。

Overall, the Agile Manifesto is a set of values and principles that prioritize collaboration, flexibility, and responsiveness in software development, and encourage development teams to focus on delivering working software that meets the needs of customers and stakeholders.

Twelve agile principles

Satisfy clients' needs	Satisfy coders' needs
The highest priority is satisfying the client (by delivering working software early and continuously)	Work at a steady, sustainable pace (no heroic efforts)
Embrace change (even late in the development cycle)	Rely on self-organising teams
Collaborate every day with the client	Teams reflect regularly on their performance
Use face to face communication	Progress is measured by the amount of working code produced
Deliver working software frequently	Continuous attention to technical excellence
	Minimise the amount of unnecessary work
	Build teams around motivated individuals

For clients:

- Agile development involves close collaboration between the development team and the client or customer throughout the development process
- Agile development focuses on delivering working software frequently, **allowing the client to provide feedback and make changes early and often**
- Agile development allows for flexibility and responsiveness to **changing client needs**, as requirements can be adapted and adjusted throughout the project
- Agile development places a strong emphasis on **delivering value to the client**, with a focus on the features and functionality that will have the greatest impact on their business
- Agile development encourages open communication and transparency between the development team and the client, promoting trust and a shared sense of ownership over the project

- Agile development can help to reduce risk and uncertainty for the client by providing early and frequent opportunities to test and validate the software
- Agile development can lead to greater client satisfaction by providing a more collaborative and responsive approach to software development
- Agile development can help to ensure that the final product meets the client's needs and expectations, as they are involved throughout the development process
- Agile development can also result in a **more efficient and cost-effective development** process, as changes can be made earlier and with less impact on the overall project

For coders:

1. Flexibility: Agile development allows coders to work in an iterative and incremental manner, which means they can adapt to changes in requirements and feedback from stakeholders more easily. This can help reduce frustration and stress that may arise from having to redo work that has already been completed.
2. Collaboration: Agile development places a strong emphasis on collaboration between team members, which can help create a supportive and positive work environment for coders. Collaboration can also help ensure that the code being developed is of high quality and meets the needs of stakeholders.
3. Continuous improvement: Agile development encourages continuous improvement and learning, which can help coders stay engaged and motivated. It can also help them stay up-to-date with the latest technologies and practices in their field.
4. Transparency: Agile development emphasizes transparency and communication, which can help coders understand the overall goals of the project and how their work fits into the bigger picture. This can help them feel more invested in the project and can help them make more informed decisions about their work.
5. Quick feedback: Agile development typically involves frequent reviews and feedback from stakeholders, which can help coders stay on track and ensure that their work meets the needs of the project. This can also help identify potential issues early on, which can save time and effort in the long run.

Overall, the Agile development model can provide coders with a supportive and collaborative work environment, opportunities for learning and growth, and a sense of ownership and pride in their work.

Agile methods:

1. Extreme Programming (XP): XP is a software development framework that aims to produce higher quality software through collaboration and frequent feedback. It emphasizes practices such as pair programming, continuous testing, continuous integration, and short development cycles.
2. Test-driven development (TDD): TDD is a development approach that emphasizes writing automated tests before writing the actual code. It helps ensure that the code is correct, maintainable, and extensible, and enables faster feedback and iteration.
3. Kanban: Kanban is a visual management tool that helps teams manage and optimize their workflow. It emphasizes continuous delivery, limiting work in progress, and optimizing the flow of work through the team's pipeline.
4. Scrum: Scrum is an Agile framework that emphasizes collaboration, self-organization, and iterative development. It involves a set of predefined roles, ceremonies (such as daily stand-ups and sprint retrospectives), and artifacts (such as backlogs and burndown charts) to help teams work effectively together.
5. Lean Software Development: An Agile methodology based on the principles of lean manufacturing, with a focus on eliminating waste, optimizing the whole system, and continuously improving the process.
6. Crystal: An Agile methodology that emphasizes communication, reflection, and frequent delivery of working software. It comes in several variants, with different levels of formality and process depending on the project size and complexity.
7. Feature-Driven Development (FDD): An Agile methodology that focuses on developing features in short, iterative cycles. It involves a strong emphasis on design and modeling, and includes practices such as feature teams and frequent code inspections.

Extreme Programming

XP is characterized by a set of practices that promote quality, teamwork, and rapid feedback, including:

1. Pair programming: Two developers work together on a single task, with one person coding while the other provides feedback, suggestions, and catches errors.
2. Test-driven development: Developers write automated tests for each feature before writing the code for that feature. This ensures that the code works as intended and catches bugs early in the development process.
3. Continuous integration: Developers integrate their code changes into a shared repository frequently, which allows for rapid feedback and identification of conflicts or issues.
4. Refactoring(**重构**): Developers regularly review and update existing code to improve its quality and maintainability.
5. Small releases: The development team releases small updates frequently, often on a weekly or daily basis. This allows for rapid feedback and course correction, and helps ensure that the product is meeting the customer's needs.
6. On-site customer: A representative from the customer or product owner is present and actively involved in the development process, providing feedback and guidance to the development team.

By emphasizing close collaboration, continuous feedback, and a focus on quality, XP can help teams deliver high-quality software that meets the needs of the customer in a rapidly changing environment.

Extreme Programming ethos: (Team-oriented)

• Simple design: use the simplest way to implement feathers ;

• Sustainable pace: effort is constant and manageable;

• Coding standards: teams follow an agreed stye and format;

• Collective ownership: everyone owns al the code;

• Whole tram approach: everyone is included in everything

Pair programming:

Two programmers work together at one workstation.

Helm: One programmer, called the driver / helm (舵手), writes the code 写代码的

Tactician: the other programmer, called the navigator /tactician(tactician), reviews the code as it is being written. 观察的 ；The navigator provides feedback, suggests improvements, and helps catch errors, while the driver focuses on writing the code.

Pair programming has several benefits. It allows for better code quality because both programmers are constantly reviewing and improving the code. It also encourages knowledge sharing and learning, as each programmer brings their own skills and expertise to the table. Pair programming can also help to reduce the number of defects in the code, as issues are caught early on during the development process. Additionally, it can help to improve communication and collaboration between team members.

The impact of pair programming:

1. Improved code quality: With two people working on the code, errors and bugs are likely to be caught more quickly, leading to better code quality.
2. Knowledge sharing: Pair programming allows for knowledge and expertise to be shared between team members, leading to a more skilled and cohesive team.
3. Increased productivity: While it may seem counterintuitive, studies have shown that pair programming can actually lead to increased productivity, as the code is written more efficiently and with fewer errors.
4. Better communication: Pair programming requires constant communication between team members, which can improve overall communication skills and lead to better collaboration on projects.
5. Reduced developer burnout: Pair programming can reduce developer burnout by providing opportunities for social interaction and preventing developers from feeling isolated or overburdened.

Overall, pair programming can lead to better quality code, increased productivity, and a more cohesive team.

Test-driven development

Test-driven development (TDD) is a software development approach that emphasizes writing automated tests before writing the code itself. TDD involves a cycle of writing a failing test, writing just enough code to pass that test, and then refactoring the code to make it cleaner and more maintainable.

In TDD, tests are written first and serve as a **specification** for the code that is yet to be written. This helps to ensure that the code meets the desired behavior and requirements. Once the test is written, the developer writes code just enough to pass the test, ensuring that the test passes and the code works as expected. The developer then refactors the code to improve its design, readability, and maintainability.

The benefits of TDD include improved code quality, better test coverage, simplified debugging, system documentation and faster feedback loops. By writing tests first, developers can catch bugs earlier in the development process, making them easier and less expensive to fix. Additionally, TDD can help to reduce the risk of regressions and can make it easier to change and refactor code later on.

TDD is often used in conjunction with other agile development practices, such as continuous integration and continuous delivery, to ensure that the code is constantly being tested and improved.

Scrum

Scrum is a widely used Agile methodology that is commonly used in software development projects. It is an iterative and incremental approach that emphasizes teamwork, collaboration, and the delivery of a working product increment within a short period called a Sprint.

Scrum consists of several roles:

•**Product Owner, the Scrum Master** (responsible for prioritizing the product backlog, which is a list of all the features, requirements, and tasks that need to be completed. The Scrum Master is responsible for ensuring that the Scrum process is followed and for removing any obstacles that may be preventing the Development Team from delivering the product increment.)

•**Development Team**: (responsible for designing, coding, testing, and delivering the product increment at the end of each sprint.)

•**Product Owner** : the client

Scrum also has several ceremonies, including the

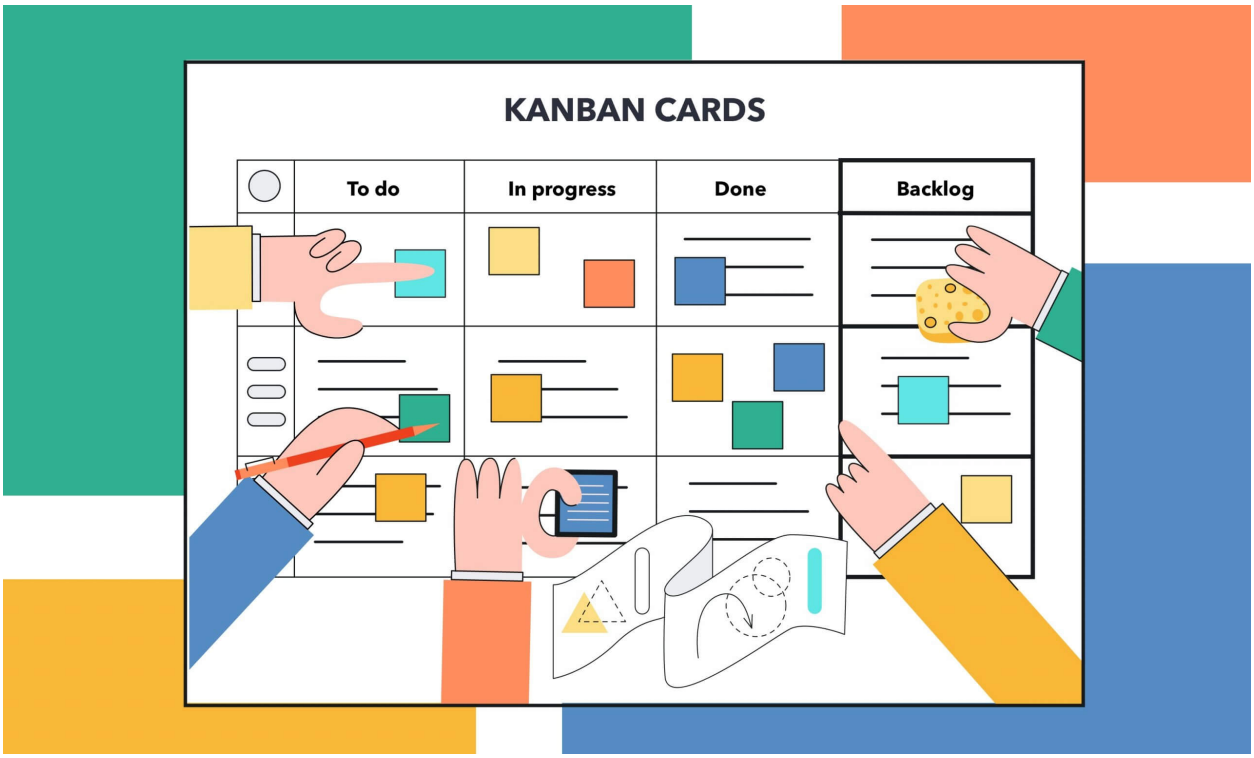
•Sprint Planning: a meeting at the beginning of each Sprint where the Development Team determines how much work they can complete during the Sprint. The Daily Scrum is a short

•the Daily Scrum: a short meeting held each day where the Development Team members share what they have done, what they are working on, and what obstacles they are facing.

•the Sprint Review: is a meeting held at the end of the Sprint where the Development Team demonstrates the product increment they have built, and the stakeholders provide feedback.

•the Sprint Retrospective(回顾): is a meeting held at the end of the Sprint where the team reflects on the previous Sprint and identifies ways to improve the process.

Kanban board



Kanban boards are visual project management tools used to track work in progress and to optimize workflow. The board consists of a series of columns that represent the various stages of a project or process, and cards or sticky notes are used to represent individual tasks or work items. As a work item progresses through each stage, it is moved to the next column on the board.

Kanban boards can be used to improve efficiency and productivity by providing a clear and visual representation of work in progress. They allow team members to quickly identify bottlenecks and prioritize work items based on their urgency and importance. Kanban boards also promote collaboration and communication among team members, as everyone has visibility into the progress of each work item.

Problems with Agile model:

- Lack of documentation :Agile development often prioritizes working software over documentation, which can lead to insufficient documentation or difficulty in maintaining documentation over time.
- Not effective for brownfield development : it is not so effective for brownfield development which involves improving and maintaining legacy systems.

- Require strong team work : Might not work well for large distributed development, and if the developer is absent or on leave it might be not that productive.
- Scope creep: Since Agile development is flexible and responsive to change, it can be easy for projects to expand beyond their original scope, leading to scope creep and delays.
- Time management: Agile development requires ongoing planning and prioritization to ensure that the team is working on the most important tasks at any given time. If time management is not handled effectively, it can lead to missed deadlines or incomplete work.
- Customer involvement: While Agile development emphasizes customer involvement, it can be challenging to keep customers engaged and informed throughout the process, especially if they are not able to commit significant time and resources to the project.
- Lack of predictability: Because Agile development is highly flexible and responsive to change, it can be difficult to predict timelines and deliverables, which can be problematic for stakeholders who need to plan around the project.

More information about why agile is not effective for brownfield development:

Agile is generally designed to work best for projects with a relatively clean slate, or greenfield development, where there are few or no constraints on the development process. In contrast, brownfield development involves working with an existing system or infrastructure, which can make it difficult to implement an agile approach. This is because:

1. Legacy systems: Brownfield development often involves working with legacy systems, which may be poorly documented and difficult to modify. This can make it hard to implement the continuous integration and testing that is a key part of agile.
2. Technical debt: Brownfield development can also involve dealing with technical debt, which is the accumulation of old, outdated code that needs to be cleaned up before new features can be added. This can slow down the development process and make it harder to deliver working software quickly.

3. Integration challenges: When working with an existing system, it can be difficult to integrate new features and changes without disrupting the existing system. This can make it hard to implement the collaborative, iterative approach that is central to agile development.

Overall, while it is still possible to use agile in brownfield development, it may require more careful planning and a more tailored approach to account for the challenges presented by working with an existing system.

Some quiz about agile development:

1. Which of the following is NOT one of the values of Agile development?
   a. Individuals and interactions over processes and tools
   b. Working software over comprehensive documentation
   c. Customer collaboration over contract negotiation
   d. Following a strict plan over adapting to change
2. Which Agile methodology places emphasis on continuous delivery and continuous deployment?
   a. Scrum
   b. Kanban
   c. Extreme Programming (XP)
   d. Lean software development
3. What is the purpose of the daily stand-up meeting in Scrum?
   a. To provide an opportunity for team members to report on their progress and identify any obstacles
   b. To review the sprint backlog and make any necessary adjustments
   c. To discuss the prioritization of user stories
   d. To review the completed work and identify any areas for improvement
4. In Test-Driven Development (TDD), what comes first?
   a. Writing the tests
   b. Writing the code
   c. Writing the user stories
   d. Reviewing the acceptance criteria

5. Which of the following is NOT a recommended practice for successful pair programming?
   a. Switching roles frequently
   b. Avoiding communication with other team members
   c. Taking breaks to prevent burnout
   d. Using a shared workstation

Answer:

1.d

2.c

Lean software development emphasizes delivering value to the customer through the elimination of waste and optimization of flow, but it does not specifically emphasize continuous delivery or deployment. Continuous delivery and deployment are primary focuses of Extreme Programming, where frequent releases are made to ensure that working software is consistently delivered to the customer.

3.a

4.a

Reviewing the acceptance criteria is an important step in the software development process, but it is not directly related to Test-Driven Development (TDD). In TDD, the process begins with writing the tests, which define the desired behavior of the code. The code is then written to satisfy the tests, and the process continues with refactoring and further iterations of test/code development.

5.b