Nanyang Technological University

CZ3002 Advanced Software Engineering

Assignment 1

Extreme Programming (XP)

Adrian Goh Jun Wei

U1721134D

What is Extreme Programming?

Extreme Programming (XP) is a lightweight agile software development framework that aims to achieve:

- Higher software quality and responsiveness to changing customer requirements
- Higher quality of life and developer experience (DX) for the engineering team

Extreme Programming has some values, principles and common practices that guide and ensure that it is being implemented and run correctly and its effectiveness is maximised.

Values

1. **Communication**: Everyone on a team works jointly at every stage of the project.
2. **Simplicity**: Developers strive to write simple code bringing more value to a product.
3. **Feedback**: Software is delivered frequently to receive feedback for further improvements.
4. **Respect**: Every person assigned to a project contributes to a common goal.
5. **Courage**: Results are objectively reviewed without making excuses

Principles

1. **Rapid feedback**: Team members understand the given feedback and react to it right away.
2. **Assumed simplicity**: Developers need to focus on the job that is important at the moment and follow YAGNI (You Ain't Gonna Need It) and DRY (Don't Repeat Yourself) principles
3. **Incremental changes**: Making many small changes constantly instead of one large change at one go.
4. **Embracing change**: Developers are to support the customer's decision for requirement changes.
5. **Quality work**: A team that works well, makes a valuable product and feels proud of it.

Practices

There are many XP practices, but listed are some of the key and most common practices in the industry.

1. **Test-Driven Development (TDD)** is the process of designing and writing tests first, and then writing code around it until all the test cases pass. Some TDD frameworks are PyUnit (Python) and RSpec (Ruby).
2. **Pair Programming** is the process of two developers programming at the same machine concurrently.
3. **Weekly Cycle** is the process of the team coming together at the start of the week to review the past week, and then have the customer prioritize the tasks to be completed in the week. Finally, the team deploys an upgraded version of the software by the end of the week.
4. **Continuous Integration (CI)** is a practice where code changes are immediately tested automatically when they are added to a larger codebase. Some CI are Jenkins, CircleCI and GitHub Actions.

Advantages (and why you should follow XP)

Shipping software of higher quality and stability with greater efficiency

- TDD ensures that the code written is correct, to begin with. Also, one can pinpoint errors and solve them quicker. Besides, TDD helps in designing better software by ensuring each unit of code follows the Single Responsibility Principle of SOLID
- Automated CI sounds out integration errors as soon as new code is being added, and can thus be looked into quickly, instead of waiting for these errors to go unnoticed or snowball.

- Pair programming produces code that is more readable, maintainable and has fewer bugs as the code is being reviewed as it is being written, thus removing any ambiguities and errors.

Reduced project cost

Often in traditional methodologies, there are elements which could be eliminated to free up resources.

- Making the customer a part of the team ensures constant communication and clarification, thus ensuring that time and resources are not wasted on misaligned requirements.
- XP is code-centric and removes some activities e.g. needless paperwork and meetings so that developers can focus on coding and completing the requirements.

In software development, staff turnover is very costly due to the loss in efficiency from finding and onboarding a new staff into the project. Contingency recruitment fee for recruiting the replacement is also very expensive.

- A weekly cycle reduces staff turnover due to burnout and poor DX, as it creates a more positive work environment by giving the team a reason to celebrate each week.
- Tight communication and interactions within the team helps to build up teamwork and camaraderie, which makes work more enjoyable and improve

Disadvantages (and why you should NOT follow XP)

Infeasibility due to constraints regarding customers

- As customers must participate in the process, it's challenging if they are busy or located remotely

Infeasibility due to constraints regarding team/management

- Team's proficiencies will affect XP's effectiveness as a less experienced team will result in cases such as writing inadequate tests or tightly-coupled code that cannot be changed easily. This might be due to the lack of budget for the hiring of calibre developers.
- XP is very technical etc. full TDD is very challenging. Thus, to convince developers to accept this tough practice is not always easy as it requires discipline in the team and devotion of your customers.
- As communication is key for XP, having a remote team will limit its effectiveness, especially so if the team is working from places with different timezone and working hours.
- Justifying some decisions, such as writing more tests instead of features or pair programming, to a management team without a technical background will be tough as benefits are not immediately visible.
- Some processes could be expensive if done incorrectly, such as setting up a poor infrastructure for testing and CI/CD or having poorly conducted meetings.

Infeasibility due to constraints regarding projects/systems

- Some systems, such as safety-critical systems, have changes that have to be managed very carefully to preserve safety.
- XP might not be ideal for legacy systems where the volume of code far outstrips the time available to maintain it.
- As XP requires tight communication and works best for a team size of up to an estimated 12 people, it is less ineffective for large projects which require a huge team due to communication overhead.
- Projects might be using technology that does not allow for automated unit and functional tests.