Course: Software Engineering

Presentation on-ExtremeProgramming

Submitted By

Shafiuddin Siyam
IT-21010
3rd year 2nd semester

Submitted To

Mr. Ziaur Rahman
Assistant Professor
Dept. of ICT
Mawlana Bhashani Science and Technology University

Index:

Introduction

Extreme Programming (XP) is an agile software development methodology aimed at improving software quality and responsiveness to changing customer requirements. Introduced in the late 1990s by Kent Beck, XP emphasizes customer satisfaction, continuous delivery, and a flexible approach to development. The key principles of XP include frequent releases in short development cycles (iterations), constant communication among team members, pair programming, rigorous testing (test-driven development), and accommodating changing requirements. This methodology empowers teams to reduce risks, increase productivity, and produce high-quality software by embracing continuous feedback and improving collaboration.

What is Extreme Programming (XP)?

Extreme Programming (XP) is an agile software development methodology designed to improve software quality and responsiveness to changing customer requirements. It emphasizes teamwork, frequent releases, and customer satisfaction, incorporating practices that promote efficiency, communication, and adaptability.

Key Principles of Extreme Programming

1. **Communication:** Encourages collaboration between developers, customers, and team members through constant feedback and open discussions.
2. **Simplicity:** Focuses on delivering the simplest solution that works, avoiding unnecessary complexity.
3. **Feedback:** Gathers feedback early and often from both customers and the development process to guide the project.
4. **Courage:** Encourages taking bold actions, like refactoring code or changing designs when needed, without fear of failure.
5. **Respect:** Values the contributions of all team members and fosters a supportive work environment.

Good Practices in Extreme Programming

•**Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.

•**Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.

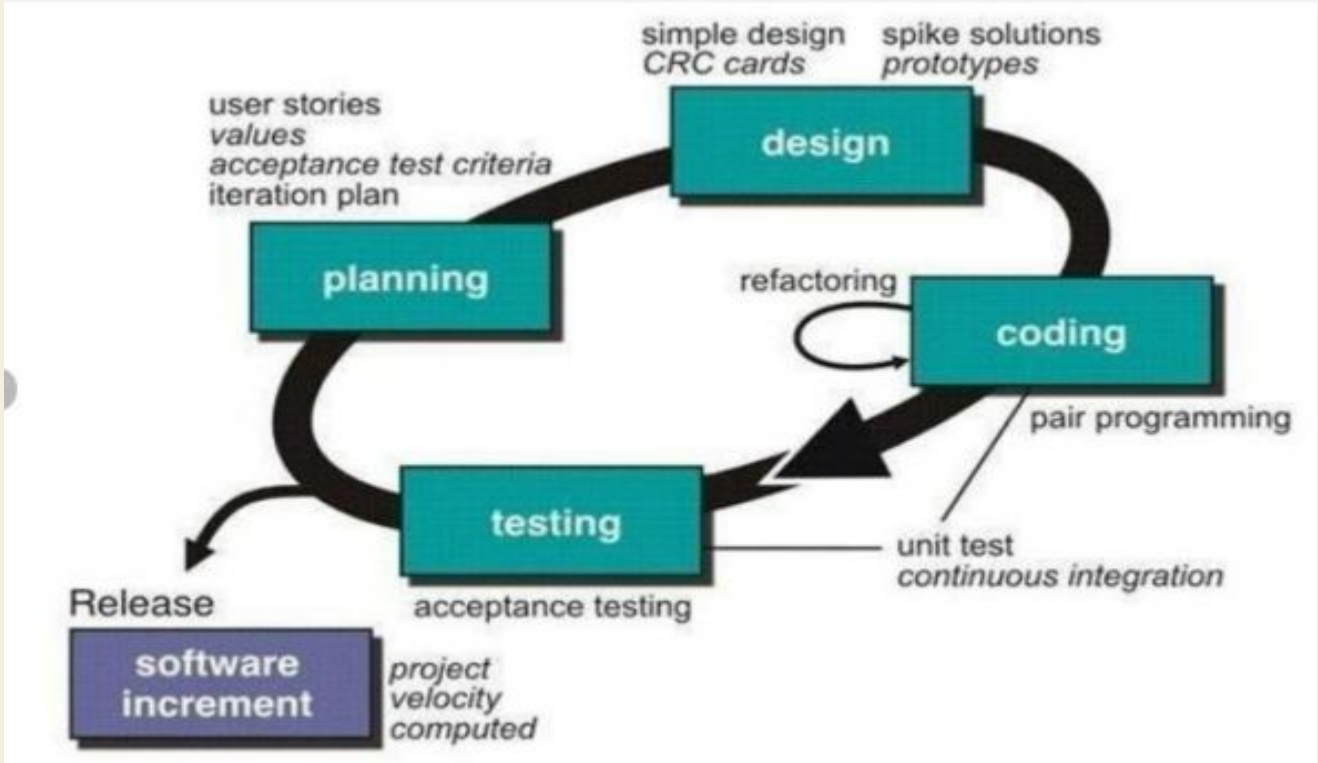
•**Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.

•**Simplicity:** Simplicity makes it easier to develop good-quality code as well as to test and debug it.

•**Design:** Good quality design is important to develop good quality software. So, everybody should design daily.

•**Integration testing:** Integration Testing helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Necessary diagrams on Extreme Programming



This diagram represents a simplified Agile Software Development Process, specifically aligned with Extreme Programming (XP) practices. Below is an explanation of the stages:

1. Planning

•**Purpose**: Defines what needs to be built and establishes priorities.
•**Key Inputs:**

- User Stories: Short descriptions of features from the user's perspective.
- Values: Principles like simplicity, feedback, and communication guide planning.
- Acceptance Test Criteria: Define what success looks like for a feature.
- Iteration Plan: A roadmap for what to develop in a given sprint or iteration

2. Design

•**Purpose:**Outlines how the software will function and organizes code structure.

•Key Practices:

• Simple Design: Keep designs minimal, avoiding over-engineering.

• CRC Cards: "Class-Responsibility-Collaborator" cards are used to model relationships between components.

• Spike Solutions: Quick explorations to resolve uncertainties or technical challenges.

• Prototypes: Basic models of features or interfaces to test feasibility.

3. Coding

•**Purpose**: Actual implementation of the planned functionality.

•**Key Practices:**

• Pair Programming: Two developers work together at one workstation to improve code quality.

• Refactoring: Simplifying and optimizing existing code without changing its behavior.

• Continuous Integration: Frequently merging and testing code changes to prevent integration issues.

• Unit Tests: Small, automated tests verify that individual components work correctly.

4. Testing
•**Purpose:** Ensures the product meets requirements and functions as intended.
•**Key Practices:**
• Acceptance Testing: Verifying that the software meets user needs.
• Continuous feedback from automated and manual tests ensures quality and guides adjustments.

5. Release
•**Purpose:** Deliver an increment of the software to users.
•**Key Outputs:**
• Software Increment: A functional version of the product that adds value.
• Project Velocity Computed: Measures how much work the team completed in the iteration to plan future sprints.

Applications of Extreme Programming (XP)

Some of the projects that are suitable to develop using the XP model are given below:

1.Small projects: The XP model is very useful in small projects consisting of small teams as faceto-face meeting is easier to achieve.
2.Projects involving new technology or Research projects: This type of project faces changing requirements rapidly and technical problems. So XP model is used to complete this type of project.
3.Web development projects: The XP model is well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
4.Collaborative projects: The XP model is useful for collaborative projects that require close collaboration between the development team and the customer.
5.Projects where quality is a high priority: The XP model places a strong emphasis on testing and quality assurance, making it a suitable approach for projects where quality is a high priority

Advantages of Extreme Programming (XP)

•**Slipped schedules:** Timely delivery is ensured through slipping timetables and doable development cycles.
•**Misunderstanding the business and/or domain** - Constant contact and explanations are ensured by including the client on the team.
•**Canceled projects:** Focusing on ongoing customer engagement guarantees open communication with the consumer and prompt problem-solving.
•**Staff turnover:** Teamwork that is focused on cooperation provides excitement and goodwill. Team spirit is fostered by multidisciplinary cohesion.
•**Costs incurred in changes:** Extensive and continuing testing ensures that the modifications do not impair the functioning of the system. A functioning system always guarantees that there is enough time to accommodate changes without impairing ongoing operations.
•**Business changes:** Changes are accepted at any moment since they are seen to be inevitable.
•**Production and post-delivery defects:** the unit tests to find and repair bugs as soon as possible.

Conclusion

Extreme Programming (XP) is a Software Development Methodology, known for its flexibility, collaboration and rapid feedback using techniques like continuous testing, frequent releases, and pair programming, in which two programmers collaborate on the same code. XP supports user involvement throughout the development process while prioritizing simplicity and communication. Overall, XP aims to deliver high-quality software quickly and adapt to changing requirements effectively

Thank You