
Software Engineering

Software Process: Agile/Extreme Programming

Chapter Objectives

- To introduce software process model for **eXtreme Programming (XP)**
- To describe the **12 practices** of XP
- To introduce the **Story Card** template describing user requirements

eXtreme Programming XP

- New approach to development based on the development and delivery of **very small increments of functionality**
- Relies on:
 - Constant code improvement
 - User involvement in the development team
 - See www.extremeprogramming.org

eXtreme Programming XP (cont.)

- Extreme Programming is the most famous of the agile methods
- Short inspect-and-adapt cycles and frequent, short feedback loops
 - working software is the primary measure of progress
 - Embrace changing requirements
- Based on the recognition that separating design, evaluation, and documentation activities in software development is a futile exercise

Extreme Programming XP

- New approach to development based on the development and delivery of **very small increments of functionality**
- Relies on:
 - Constant code improvement
 - User involvement in the development team
 - See **www.extremeprogramming.org**

Extreme Programming XP (cont.)

- Extreme Programming is the most famous of the agile methods
- Short inspect-and-adapt cycles and frequent, short feedback loops
 - working software is the primary measure of progress
 - Embrace changing requirements
- Based on the recognition that separating design, evaluation, and documentation activities in software development is a futile exercise

eXtreme Programming

XP 12 practices

1. Planning Game
2. **Small Releases**
3. Testing
4. **Pair Programming**
5. Refactoring
6. **Continuous Integration**
7. **Simple Design**
8. Collective Code Ownership
9. **On-site Customer**
10. **Coding Standards**
11. Sustainable Pace
12. Metaphor

Extreme Programming

XP 12 practices (cont.)

1. Planning Game

- At the beginning of each release, the stakeholders negotiate the feature to realize.
- The business people decide how important a feature is, and the developers decide how much that feature will cost “effort in person-hour) to implement (user stories).
- The techniques for gathering requirements in XP are a radical departure from that of more traditional software methodologies.
- First, customer requirements are written on "**User Story**" cards, similar to use cases.

Extreme Programming

XP 12 practices (cont.)

- Software developers write **time estimates** (Effort) and customers assign **priorities** to each card.
- Together, developers and customers play the “Planning Game” in which the customer chooses those User Stories that comprise the most important content for a short, incremental deliverable of about 2-4 weeks.
- When completed by developers, each short implementation increment is accepted and tried by the customer.
- Then, the remaining User Stories are re-examined for possible requirement and/or priority changes and the Planning Game is re-played for the next implementation increment.

XP Customer Story Card

CUSTOMER STORY CARD

Story Card No:

Story Card Name:

Customer:

System:

Date:

Customer Effort writing SC (hrs):

Type of activity (New Story/Fix Defect/Enhance New Feature):

Failure Risk Impact (High/Medium/Low):

Priority (High/Medium/Low):

Acceptance 'Functional' Test Scenario No:

Story Description

Pre-conditions

1.

...

Story Description

...

Post-conditions

1.

...

Notes:

Extreme Programming

XP 12 practices (cont.)

2. Small Releases

- Small increments that result in a deliverable, working application
- XP heightens the pace of spiral development by having short releases of 2-4 weeks.
- At the end of each release, the customer reviews the interim product, identify defects, and adjust future requirements.

Extreme Programming

XP 12 practices (cont.)

3. Testing

- **Test first** “Unit testing”
- Extensive, **automated white box** test cases are written **before** production code is produced.
- Codes for automated tests are added to the **code base**.
- Before a programmer can integrate their code into the code base, they must pass 100 % of their own test cases and 100% of every test that was ever written on the code base.
- This ensures that the new code implements the new functionality without breaking anyone else’s code.

Extreme Programming

XP 12 practices (cont.)

- Functional Test.
 - Black-box test cases are written before the corresponding code. This approach gives confidence to make modifications
 - Traditionally, project management techniques have been based on a developer's own assessment of how much of their task has been completed.
 - Alternately, XP promotes the use of functional test case tracking for calculating project completeness.
 - XP terms this assessment "Project Velocity."
 - Functional test cases are based on customer scenarios. When a functional test case is successfully passed, it can be considered that a specified functionality has been implemented properly.
 - Project completeness is based on the percentage of functional test cases that have been passed. Team members can unequivocally compute this measure.

Extreme Programming

XP 12 practices (cont.)

4. Pair Programming

- Significant coding is done in pairs of programmers (working at the same workstation). An extra set of eyes has been shown to reduce the defect rate of software.

5. Refactoring

- Performing a number of small, and frequently applied, transformations, which improves structure of code without affecting its behavior (not improving functionality)

Extreme Programming

XP 12 practices (cont.)

6. Continuous Integration

- Describes the immediate and ongoing integration of completed tasks into the system
- Coding assignments are broken up into **small tasks**, preferably of no more than one day.
- When each task is completed, it is integrated into the collective **code base**.
- There are many product builds each day.

Extreme Programming

XP 12 practices (cont.)

7. Simple Design

- Consider the simplest thing that could possibly work and do that or the next best thing don't predict what is coming next because it probably isn't
- XP strives for supremely simple designs.
- Programmers should not try to predict future needs and to design accordingly.
 - » “You aren't gonna need it.” (or YAGNI)
 - » and “Do the simplest thing that could possibly work.”

Extreme Programming

XP 12 practices (cont.)

8. Collective Code Ownership

- Distribute skills and knowledge as much as necessary or possible (to avoid the “truck” factor)

9. On-site Customer

- Customer specify requirements using user stories and are present to answer questions regarding them
- Customer serve as an information source
- Customer is always readily available and accessible to the developers for the purpose of clarifying and validating requirements throughout the implementation process.

Extreme Programming

XP 12 practices (cont.)

10. Coding Standards

- Use of uniform, consistent coding standards simplifies working on the source code
- Offer recommendations to developers with respect to:
 - how to lay out the code, e.g. tabbing, braces, comments
 - where source code files should be located in the file system
 - which settings should be used for the compiler and linker
 - which naming conventions should be followed for files, classes, methods, attributes, parameters, and local variables.

Extreme Programming

XP 12 practices (cont.)

11. Sustainable Pace

- Avoid overworking (keep to 40 hour weeks)

12. Metaphor

- A metaphor is “a rhetorical figure, which expresses what was meant by using an imagination (mostly a picture), which stems from a completely different domain and which has no concrete relation to what was meant”
- Metaphors allow developers and customers to communicate understanding through verbal pictures

XP Metaphor Example- Payroll system is like an assembly line

- A large XP project was a payroll system for Chrysler.
- The metaphor for this project was that the payroll system was **like an assembly line** where hour parts were converted to dollar parts, all parts were assembled and a paycheck was produced

XP Metaphor Example: Test-First Programming is like a Math Book

- Test-first, or test-driven development, is one of XP's fundamental design techniques.
- It works like this:
 1. Write a test, which probably can't even compile because you haven't written any **real code yet**.
 2. Implement **a stub** of the feature, so the test runs but fails.
 3. Complete the feature so the test passes.

XP Metaphor Example: Test-First Programming is like a Math Book

- Many math books have the answers to odd-numbered problems in the back of the book. They do this because, paradoxically, you don't solve math book problems to get the answer. Rather, you solve them to learn the techniques for solving similar problems.
- **Test-first is like this: first you have the right answer - the test. Your challenge is to write the code that provides the answer. Usually, if your code is wrong, your test will detect that.**
- The fun part is, you act as both textbook author and student. The author chooses a problem at just the right of level of difficulty, just as you must do for your test.
- So, test-first is like a math book.

Adherence to XP:

Shodan Input Metric Survey

- The Shodan Adherence Survey is a subjective means of gathering adherence information from team members.
- The survey, answered anonymously via a web-based survey, is composed of 15 questions on the extent to which each individual on a team uses XP practices (testing has been split to three categories and stand up meetings were added to the practices).
- A survey respondent self-reports the extent to which he or she used the practice, on a scale from 0% (never) to 100% (always). An overall score for the survey is computed via a weighted average of each response.

Shodan survey questions

For each question, the respondents were asked to use the following scale:

- 10 Fanatic (100%)
- 9 Always (90%)
- 8 Regular (80%)
- 7 Often (70%)
- 6 Usually (60%)
- 5 Half 'n Half (50%)
- 4 Common (40%)
- 3 Sometimes (30%)
- 2 Rarely (20%)
- 1 Hardly ever (10%)
- 0 Disagree with using this practice

Shodan Input Metric Survey

- See the following link for details of how each XP practice is evaluated:
- [RC_XP_Adherence_Metrics_Shodan.doc](#)