Introduction to Agile Processes and Extreme Programming

James Newkirk
ThoughtWorks
Chicago, IL, USA
http://www.thoughtworks.com
jnewkirk@thoughtworks.com

ABSTRACT

Extreme Programming is one of the most discussed subjects in the software development community. But what makes XP extreme? And how does it fit into the New World of agile methodologies? This tutorial will establish the underpinnings of agile methodology and explain why you might want to try one. Then we will see how XP uses a set of practices to build an effective software development team that produces quality software in a predictable and repeatable manner.

1. INTRODUCTION

Agile methods are the current rave amongst many software developers. But, what are agile methods? One of the most popular and also one of the most prescriptive methods is extreme programming. Extreme programming (XP) was developed to address the needs of small teams who are confronted with vague and changing requirements.

2. THE VALUES

When evaluating certain activities in the software development process one must have a reference to understand if we are making progress. XP has four core values that are used to guide the practices that are employed. These values are:

Communication – Most project failures can be traced to problems with communication. Communication overrides almost all of the other values. This includes communication between all team members, customers, programmers and managers.

Simplicity – "What is the simplest thing that could possibly work?" [1] The message is very clear. Given the requirements for today design and write your software. Do not try to anticipate the future, let the future unfold. It will often do this in very unpredictable ways making anticipation a cost that is often too expensive to afford.

Feedback – XP practices are designed to illicit feedback early and often. The practices such as short releases, continuous integration, testing provide very clear feedback. This feedback enables the project to move forward on a very sound foundation.

Courage – This is an important value. Often times many methodologies are based on being defensive. XP challenges this notion by asking the participants to play to win. It takes courage to say I have done enough design for now and I'll let the future happen.

ICSE'02, May 19-25, 2002, Orlando, Florida, USA.

3. THE PRACTICES

Extreme Programming is implemented with 12 practices. In [2] these are called the circle of life. They are the lifeblood of extreme programming. Each practice in and of itself can be described in terms of their adherence to the 4 values of XP. They also work together to form a whole that is much greater than the sum of the parts.

Planning – Determine the scope of the next iteration by working with customers who provide business priorities and with programmers who provide technical estimates.

Small Releases – Get the system into production quickly. This is a key factor in getting feedback on the actual software.

Metaphor – Understand how the whole system works. It is just as important for the customer to understand the metaphor along with the programmers.

Simple design – One of the key values is simplicity. The system should be designed for the features that are implemented today. Let the future dictate how the system evolves to that point, do not try and anticipate the future, you will probably be wrong.

Testing – Feedback is also one of the key values. Tests include unit tests, which programmers write and acceptance tests, which customers write. Tests are the indicator of completion.

Refactoring [4] – Programmers are responsible for improving the design of existing software without changing it's behavior. Refactoring is part of the programmer's everyday activities.

Pair Programming – Working with a partner is a requirement when writing production code.

Collective ownership – Anyone on the team can change any part of the system.

Continuous integration – Programmers integrate and build the software many times a day.

40-hour week – A better name would be to work until tired. However, be aware of the downsides of working too many hours many weeks in a row.

On-site customer – The customer is on the team, available to answer questions full-time. The customer is also responsible for writing acceptance tests.

Coding standards – Communication is a key value. Adopting coding standards improves communication because the code is consistent from class-to-class.

4. FURTHER INFORMATION

For a detailed case study of what it is like to adopt extreme programming see my book [5]. This book details a 3-week project providing various artifacts that are produced when working on an extreme programming project. These artifacts include, release plans, iteration plans, story cards, task cards, and code.

For additional information on the details of XP see the references below for more information.

5. ACKNOWLEDGMENTS

My efforts to describe extreme programming and agile methods depend heavily upon the works of other people. In particular I need to acknowledge Ward Cunningham and Kent Beck whose pioneering efforts in this area are well noted. I would also like to acknowledge the efforts of Martin Fowler, Ron Jeffries, Lowell Lindstrom, and Robert C. Martin for their work in teaching extreme programming at many XP Immersion classes over the past 2 years.

I must also thank countless people who have contributed to extreme programming through writing down their ideas and experiences on the Wiki (http://www.c2.com/) and on the extreme programming mailing list.

6. REFERENCES

[1] Beck, K. Extreme Programming Explained, Addison-Wesley, Reading MA, 2000

[2] Jeffries, R, Hendrickson, C., Anderson, A., Extreme Programming Installed, Addison-Wesley, 2001

[3] Cockburn, A., Agile Software Development, Addison-Wesley, 2002

[4] Fowler, M. Refactoring: Improving the Design of Existing Code, Addison-Wesley, Reading MA, 1997

[5] Newkirk, J. Martin, R. C., Extreme Programming in Practice, Addison-Wesley, Reading MA, 2001