Research Findings: XP Methodology Suitability

Date 13 August 2015
Researcher Sean Young

---

1. Background

We are starting development next week and need further research as to what is involved during development using the extreme programming (XP) practice. We need to determine the suitability of XP development for our project.

---

2. Objectives

Research the XP practice, its benefits during development and what is documented during development using XP

---

3. Approach

Internet research.

---

4. Findings

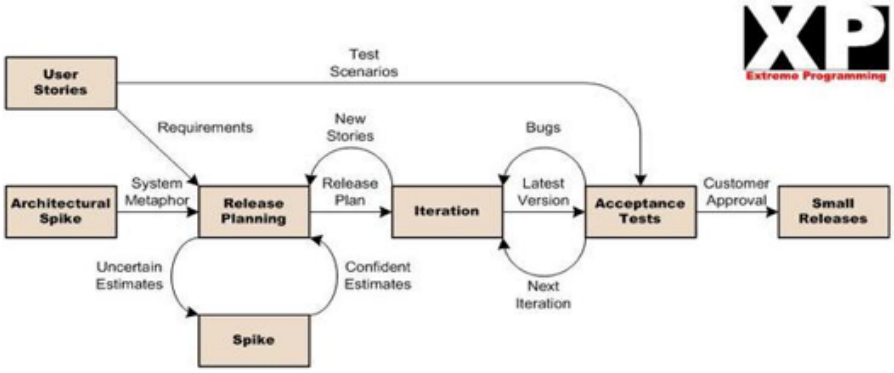
4.1. XP lifecycle:

4.1.1. Exploration: At this stage the client and developers come to a mutual understanding of what the systems vision and mission is. We have done this through the last meeting with the client when we gathered and documented requirements.

4.1.2. Planning: This involves both the client and developers coming to an agreement on what the user stories are and in what order are they going to be delivered in such a way that makes both business and technical sense. User stories should be given highest to lowest priority and should be strictly developed in that order during the projects development.

4.1.3. Iterations to First Release: Iterations last from 1 to 4 weeks long. Each iteration should aim to deliver the most value to the client as the system continues to grow over time. The first iteration should focus on the components required in the system that provide a solid base to work from in future iterations. At the end of an iteration the client/users should run the system by the acceptance tests.

4.1.4. Productionizing: When the client is happy with the product it can be deployed into the client environment where it can undergo system or application tuning. The goal here is to continue working on the functional growth of the system but to also stabilise the system



4.2. Benefits:

4.2.1. For the developers more focus is put on coding to avoid unnecessary documenting and meetings. It creates a better social dynamic and there are more opportunities to experience and learn more skills. This also provides more work value and satisfaction for the developer

4.2.2. Developers use peer programming to code in pairs or small groups at the same time so the code is more consistent and more readable. Having more coders working on a function rather than just one also provides more efficiency in a given time frame.

4.2.3. You can have a small release of tested software at the end of an iteration to show to the client. This involves the client and makes them feel a part of the project and also gives an opportunity for feedback from the client to highlight any flaws in early stages of development.

4.2.4. Very adaptable as during sprints, features within that sprint are able to be changed or swapped as long as development of that feature has not been started yet.

4.2.5. Test driven development allows you to write a test before the code is written to test for bugs and errors before new features are added to the system code.

4.3. Documentation

4.3.1. During XP there is no specification of potential documents that are to be created by the project team. It is suggested that you work together with your stakeholders in an environment of rapid feedback where you discuss what the stakeholder needs in terms documentation.

4.3.2. XP uses a concept of “travelling light”. This means that the project team creates just enough models and documentation that are actually needed. Too little or much can put the team at risk of things such as wasting time documenting unnecessary things or lack of evidence for development such as not documenting user stories effectively.

4.3.3. When documenting it is important to understand that it is business decision and not a technical one. As Ron Jeffries quoted “If there is a need for a document, the customer should request the document in the same way that she would request a feature: with a story card. The team will estimate the cost of the document, and the customer may schedule it in any iteration he/she wishes.”

4.3.4. The need for documentation during development is reduced by its practices. For example Test Driven Development (TDD) uses the acceptance tests as documentation to track and show that the systems code performs correctly and fulfils the requirements that the team has gathered. Refactoring of code also reduces the need for documentation as if the code is very clean and easily readable, why would you spend more time investing in documenting/commenting code to help understand code that is already easy to read.

4.3.5. Documents that are not required but may need to be produced as follows

4.3.5.1. System Documentation: For developers and provides an overview of the system to help people understand the system.

4.3.5.2. User Documentation: reference manual for users on how to use and work with the system.

5. Further Investigation

5.1. None.

6. Recommendations

6.1. To manage development documentation and represent the timeline of the project team's development progress, I suggest we use a form of digital storyboard to track progress effectively and easily .A site I have used in my past experience called trello.com will enable the project team to track the product development and user progress while giving the entire team a high level view of the project so we know what is being developed right now and what needs to be developed in the future.

6.2. Crafting user stories should be the next step along with acceptance tests as after this the team can then plan the first iteration and begin development. A confirmation from the client that the user stories are in correct priority is required

6.3. A definition of done (DoD) may need to be created to act as a standard during development and also an architectural diagram to give a high level view of the system for other stakeholders such as AUT . This should be discussed between the team

6.4. We have approximately 10 weeks from the start of week 5 and I suggest our iterations to be up to 2 weeks long making a total of 5 iterations to work with. This gives us room to fit lots of small tasks into the iteration blocks and maximise productivity whilst allowing for changes and feedback from the client

7. References

Agile Modeling. (2012). Agile Modeling and eXtreme Programming (XP).Retrieved from http://agilemodeling.com/essays/agileModelingXP.htm

Agile Modeling. (2012). Throughout the XP Lifecycle .Retrieved from http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm

Dip. (2009). The top 10 benefits of Extreme Programming. Retrieved fromhttp://enterpriseblog.net/a/top-10-benefits-of-extreme-programming/

Stewart Baird. (2002). XP: A Project Managers Primer. Retrieved from http://www.informit.com/articles/article.aspx?p=26060&seqNum=6