Research Findings: XP Methodology

Date 20 March 2015
Researcher Joshua Son

---

1. Background

Our team has outlined a scope for our project. We require a development methodology which should follow an agile approach as our client's requirements are not concrete.

---

2. Objectives

To investigate different agile methodologies and select one which we can implement during our development phase.

---

3. Approach

Gain a general understanding of a few development methodologies. Delve further into one by looking into literary sources on AUT Library.

---

4. Findings

4.1. XP Development – Background

4.1.1. Originally designed for small teams working with uncertain and changing requirements. Other approaches were considered 'overkill' for small development teams. XP was developed for these teams without discarding most of the ideology. It focuses on the "timely delivery of software that meets users' requirements". (Hunt, J., 2006)

4.1.2. It is very lightweight meaning you can "take what you think you need from XP and create something which is neither XP nor particularly agile." (Pierce, D., 2002)

4.2. XP's values:

4.2.1. Communication – Communication is an obvious value within a team project however it seems to be difficult for members to get right. Poor communication leads to problems and defects in the implementation.

4.2.2. Simplicity – Keep things simple. Develop the simplest solution to better understand, implement and test the software and thus are easier to debug any errors in the code.

4.2.3. Feedback – Frequent feedback from customers, team, and stakeholders helps to identify problems as early on. This prevents unanticipated surprises which may hinder productivity further on.

4.2.4. Courage – Courage is required to adopt XP's methods.

4.3. XP's practices:

4.3.1. Planning

4.3.2. Small release

4.3.3. Simple design

4.3.4. Testing

4.3.5. Refactoring

4.3.6. Pair programming

4.3.7. Collective ownership

4.3.8. Continuous integration

4.3.9. On-site customer

4.3.10. Coding standards

4.3.11. 40-hour week

4.3.12. System metaphor

4.4. XP's phases:

4.4.1. Planning – Requirements elicitation and user stories. Time and costs are usually estimated before each iteration. An example is the critical path method.

4.4.2. Designing – Designing is based on the practices of XP. Simplicity is the main ideology during the designing phase of each iteration. Using agreed coding standards allows for a more fluid collaboration of work among different members.

4.4.3. Coding – Developing code on agreed standards to ensure integrity. Pair programming occurs at this phase, adopting the collective ownership policy. Working less hours to ensure optimisation of mentality of programmers and thus quality of code.

4.4.4. Testing – Testing code against unit tests eliminate bugs while acceptance tests ensures the intended functionality of the system has met the user requirements.

4.4.5. Listening – Customer involvement is a fundamental aspect of XP. Feedback from customers ensures that the customer is satisfied with the features and functionalities of the iteration.

4.5. Limitations and risks of XP:

4.5.1. Doesn't use the linear approach of planning, analysing and designing. Described like a jigsaw puzzle – small pieces which don't make sense by themselves but makes a complete package when joined.

4.5.2. Works well for smaller teams, not so well for larger teams.

4.5.3. Scope creep – never ending project unless maintained.

4.5.4. Possible failure to document.

4.6. Why we should use it?

4.6.1. Allows for uncertain and changing requirements.

4.6.2. Targets small development teams.

4.6.3. Adaptive – adjusts to the needs of the group.

4.6.4. Simple – clear and concise values and practices.

4.6.5. Small processes – planning then life cycle of designing, coding, testing, listening.

---

5. Further Investigation

5.1. None.

---

6. Recommendations

6.1. Use findings to discuss the applicability of XP methodology with our team plan, values, goals etc at the next meeting.

6.2. Possibly investigate other methodologies and their applicability to our team project.

6.3. Adapt the XP methodology to suit our team's needs.

---

7. References

Edwards, G. (2011).*Understanding the Extreme Programming Life Cycle Phases. (n.d.)*. Retrieved March 2015, from http://www.brighthubpm.com/methods-strategies/88996-the-extreme-programming-life-cycle/#imgn_0

Hunt, J. (2006). Agile software construction. London: Springer. dos 10.1007/1-84628-262-4_5

Duncan Pierce. Extreme Programming. The Computer Bulletin (2002) 44 (3): 28 doi: 10.1093/combul/44.3.28