The effects of using Planning Poker on Code Quality

Gustav Bylund
IDA, Linkoping University
gustav.bylund@liu.se

Filip Lindman Marko
IDA, Linkoping University
filip.lindman.marko@liu.se

ABSTRACT

Planning poker is a technique for estimating the effort required for tasks within a programming project. It aims to eliminate certain biases of the participants in order to achieve greater accuracy of estimates[4].

There is a lack of knowledge regarding how Planning Poker affects other aspects of development. This study aims to examine the effects of using Planning Poker within a project on the quality of code produced.

Author Keywords

Planning Poker; Code Quality; Agile

INTRODUCTION

In order to estimate the workload that a development group can take on, the use of a good effort estimation technique is required. Planning poker is a well-known practice within agile development [3]. There has been some studies on the accuracy of estimates, where planning poker has been found to possibly contribute to an increased accuracy of estimates [10, 9, 5].

Research has also shown that Planning Poker might also affect other aspects of development. Moløkken-Østvold suggested that the use of Planning Poker might impact Code Quality and programmer enjoyment [10].

PURPOSE

There are still unexplored possible benefits of using Planning Poker in projects. This study attempts to examine Planning Poker from a different perspective, evaluating the technique using other metrics than just estimate accuracy. An area of special interest is how the test participants experience the method.

Research questions

1. How is code quality affected by using Planning Poker within a project?
2. How do participants experience code quality changes from using Planning Poker?
3. How accurate is the estimates of Planning Poker compared to Expert Estimates?

LIMITATIONS

The study covers only two different effort estimation techniques. The study only involves one group of developers.

BACKGROUND

Briteback is medium-sized startup developing a web service for handling E-mail and internal company communication [1]. During our study, around X developers worked with the project. The primary development method used at Briteback is Scrum. Scrum is built upon using sprints and dividing the project tasks into user stories[7]. A sprint is a unit of time, during which a portion of the development is carried out. A user story is a more or less independent part of a project. In the start of each sprint, all user stories for that sprint are effort estimated and assigned to the responsible developer.

RELATED WORKS

In order to explore the relationship between the use of planning poker and code quality, we first need to define both concepts. This section aims to explain the concept of Planning Poker and Code Quality, as defined by others.

Planning Poker

Planning Poker was first introduced as a effort estimation technique by James Grenning in 2002 [4]. Grenning suggests that using less precise estimates and actively involve all developers would decrease time spent on estimation, and give more accurate estimates. The concept has since been explored in several other projects and has been found to contribute to an increased accuracy of estimates [9, 5].

Planning poker has been suggested to have other benefits apart from accurate estimations, such as increased knowledge sharing and programmer enjoyment. Use of Planning Poker has also been speculated to increase the quality of code produced [10].

Code Quality

Code Quality is a common name given to the measurement of software quality. In order to perform software quality measurement, we will need some sort of explored standard. The ISO/IEC 25010:2011 defines a quality in use model composed of eight characteristics[6], which we will use to analyse the quality of our code:

- functional suitability
- reliability
- performance efficiency
- usability

- security
- compatibility
- maintainability
- portability

Agile teams often find the use of automated tools for code quality assurance important [12], and the use of automated testing tools as a method of evaluating code quality will also be examined [2].

METHOD

We followed a team of X programmers during Y weeks of development on a medium-sized project. During this time we acted as participating observers, taking part in group meetings and product development.

The development period was divided into week-long sprints. Task estimates and completion data was recorded for each sprint. Half of the sprints began with a Planning Poker session. Remaining sprints utilised Expert Estimates instead. We opted to alternate between PP and EE every week, in order to minimise the progressive effects [8]. This reduces the risk that the results would be scewed towards portraying either method differently due to it being used earlier or later in the development process.

Estimate Accuracy

In order to measure actual effort spent on tasks, each team member was asked to log his/her time spent using Toggl [11]]. This was then compared to the estimates, to produce the accuracy of estimates.

Code Quality

In order to measure Code Quality, we opted to use code review sessions at every sprint end. During code reviews, developers were asked to note how many times they experienced confusion when reviewing code written by others, and the number of errors they found in the code. Developers did not review their own code. We also used an automated code analysis tool, JSLint, to show syntactical errors.

Other metrics

At the end of each development sprint each team member answered a survey, where they rated their experience of the week and the work produced. They were also asked to estimate how accurate the estimates had been, and how enjoyable they found the planning method. The team members were also encouraged to comment with any additional thoughts on the process.

REFERENCES

1. Briteback AB. Briteback. (????). Retrieved March 13, 2015 from **https://briteback.com/**.
2. Kirsti M Ala-Mutka. 2005. A survey of automated assessment approaches for programming assignments. *Computer science education* 15, 2 (2005), 83–102.
3. Mike Cohn. 2005. *Agile estimating and planning*. Pearson Education. 56–58 pages.
4. James Grenning. 2002. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting* 3 (2002).
5. N.C. Haugen. 2006. An empirical study of using planning poker for user story estimation. In *Agile Conference, 2006*. 9 pp.–34. DOI: **http://dx.doi.org/10.1109/AGILE.2006.16**
6. ISQS ISO. 2011. *ISO/IEC 25010". 2011*. Technical Report.
7. Pieter Jongerius. 2014. *Get Agile: Scrum for UX, design & development*. BIS Publishers.
8. Anders Kjellberg. 2011. *Experimentell metodik för beteendevetare*. Studentlitteratur, Lund. 236–254 pages.
9. Viljan Mahnič and Tomaž Hovelja. 2012. On using planning poker for estimating user stories. *Journal of Systems and Software* 85, 9 (2012), 2086 – 2095. DOI: **http://dx.doi.org/10.1016/j.jss.2012.04.005** Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).
10. Kjetil Moløkken-Østvold, Nils Christian Haugen, and Hans Christian Benestad. 2008. Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software* 81, 12 (2008), 2106–2117.
11. Toggl OÜ. Toggl - Free Time Tracking. (????). Retrieved March 13, 2015 from **https://toggl.com/about**.
12. Laurie Williams. 2012. What Agile Teams Think of Agile Principles. *Commun. ACM* 55, 4 (April 2012), 71–76. DOI: **http://dx.doi.org/10.1145/2133806.2133823**