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 time required for tasks within a programming project, which tries to eliminate certain biases of the participants in order to achieve greater accuracy of the estimates[2]. 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

Within agile development, a lot of factors come in to play when managing a project. One major factor is the use of user stories for planning the project. Within agile development it has become more and more important to have accurate time estimates. There has been some studies on the accuracy of estimates, where planning poker has been found to possibly increase the accuracy of estimates[6].

Planning Poker isn't rated as one of the most important aspects of agile development[8], and we ask ourselves why? Could developers not see the, perhaps hidden, beneficial aspects of Planning Poker, or is it simply so that developers dislike using Planning Poker? We will therefore try to explore the possibly hidden benefits of using Planning Poker.

PURPOSE

This study will examine Planning Poker as a method to estimate time required for tasks within a programming project. It is to be investigated especially how the test participants experience the method, but also how the estimates compare to other methods.

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 <insert other method here>

LIMITATIONS

The study will only involve two different methods of time estimation.

The study only involves one group of programmers.

BACKGROUND AND RELATED WORKS

In order to explore the relationship between the use of planning poker and code quality, we first need to define both concepts.

Planning Poker

Planning Poker was first introduced as a time estimation technique by James Grenning in [year] [2]. Grenning suggests that using less precise estimates and actively involve all programmers 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[5, 3].

Planning poker has been suggested to have other benefits apart from accurate estimations, such as increased knowledge sharing and programmer enjoyment[7]. Use of Planning Poker has also been speculated to increase the quality of code produced[6].

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[4], 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[8], and we will try to evaluate how we can incorporate that into our testing[1].

METHOD

We followed a team of X programmers during X weeks of development on a [briefly describe project]. During this time we acted as participating observers, taking part in group meetings and development.

The development period was divided in to week-long sprints. [describe sprints and tasks?] Task estimation and completion data was recorded for each sprint. We have participated as active [observants] in a group of developers during a period of X weeks. Every week has been it's own development sprint, with task estimation and completion data recorded for each week.

Half of the sprints were initiated with a Planning Poker session. Remaining sprints utilised the team's pre-existing estimation technique [expert estimates?].

Each week was also concluded with a code review session, where we tried to measure code quality using the aspects described above.

We opted to alternate between the two methods every other week, in order to minimise the effects of [X bias]. This ensures that we would not be biased towards thinking that either method is better, only as a result of it being used earlier or later in the development process. In order to measure actual time spent on tasks, each team member was asked to log his/her time spent using [time logging method]. This was then compared to the estimates, to produce the accuracy of estimates.

At the end of each development sprint each team member answered a survey, where they could rate 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. The results from the survey were anValysed to see if there was any trends among the team members.

CONCLUSION

Our conclusion is that...

ACKNOWLEDGEMENTS

Thanks to the team at Briteback for allowing us to use them as Guinea Pigs.

REFERENCES

1. Ala-Mutka, K. M. A survey of automated assessment approaches for programming assignments. *Computer science education 15*, 2 (2005), 83–102.
2. Grenning, J. Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting 3* (2002).
3. Haugen, N. An empirical study of using planning poker for user story estimation. In *Agile Conference, 2006* (July 2006), 9 pp.–34.
4. ISO, I. Iso/iec 25010". 2011. Tech. rep., 2011.
5. Mahnič, V., and Hovelja, T. On using planning poker for estimating user stories. *Journal of Systems and Software 85*, 9 (2012), 2086 – 2095. Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).
6. Molokken-Ostvold, K., and Haugen, N. C. Combining estimates with planning poker–an empirical study. In *Software Engineering Conference, 2007. ASWEC 2007. 18th Australian*, IEEE (2007), 349–358.
7. Moløkken-Østvold, K., Haugen, N. C., and Benestad, H. C. Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software 81*, 12 (2008), 2106–2117.
8. Williams, L. What agile teams think of agile principles. *Commun. ACM 55*, 4 (Apr. 2012), 71–76.