Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

CASE STUDY: USING PAIR PROGRAMMING IN DEVELOPMENT OF A COMPLEX MODULE

KRAJCAR, M[irjana]

Abstract: *Theory and studies of pair programming tell that in projects where pair programming technique is used, quality of code is higher than in those where code is developed by single developers. To verify this hypothesis, we conducted an experiment in which a pair of programmers developed a complex module over a two-month period. The number of reported bugs in this module was compared with number of reported bugs in modules of similar complexity developed by single developers. Finally, long term commercial benefits of pair programming are discussed.*
Key words: *pair, programming, extreme, quality, productivity*

1. INTRODUCTION

In software development projects, it is not only important to create new application modules with great functionality, but it is also equally important to assure easy maintenance, which includes bug fixes and code refactoring.

Company Perpetuum Mobile, whose main areas of expertise are Internet technologies and portals, conducts a long-term project of developing software for web content management. Many of the company's developers participated in this project. Most complex modules were developed entirely by one person. Maintaining these modules has proved to be difficult, because the person that developed a certain module was most of the time engaged in developing a new module or maintain another existing one. In order to avoid situations like these, we decided to carry out an experiment of using pair programming during development of one complex module. Afterwards, we would analyze results and decide about possible changes in the development process.

Pair programming is one of several extreme programming techniques. This technique requires that two developers work together on a single computer (Roodyn, 2004). One of them is writing a code, and he is called the driver. Another one, called observer or navigator, is just sitting aside, examining the work of the driver, watching for errors and constantly reviewing the code written by the driver. Driver and navigator periodically change their roles. It is really important that the navigator is not just passively watching the process. Instead he should be actively involved in the process of creating code by thinking and communicating with the driver about what could be done differently (McConnell, 2004).

Pair programming enables the company to avoid situations where only one person is familiar with the code of some module. That was the main reason that Perpetuum Mobile decided to experiment with pair programming. If two people work together on one task, it is easier for the team to compensate when an expert who developed the code cannot work on it any longer. This benefit can be even greater in bigger teams where it is possible to change pairs several times during the project, during the development of certain task or even during the day regardless whether the task is finished or not. This enables the distribution of project-specific knowledge among the team. Apart from project-related knowledge, developers naturally share other technical knowledge adopted in their free time, because they work more closely and communicate more compared to the situations when they develop code individually.

2. PAIR PROGRAMMING ANALYSIS

Among all extreme programming techniques, pair programming raises most concerns. Working in pairs can be seen as a time loss, especially in situations when it is highly important to finish functionality as soon as possible, no matter if code quality is lower. If no studies are analyzed and no experiment has been done, the expected presumption is that pair programming requires almost twice as much time then when developers are working alone. This can be true if we compare only code writing, but if we compare time spent on analysis, design and programming itself, it is possible for pairs to finish the job in 40-50% of the time needed by two single developers (Williams et al., 2000).

This can be explained by the fact that group IQ is bigger than the sum of IQs of individuals in the group (Williams & Sternberg, 1988). Developers working in a pair usually design solutions far better than those of individual developers, due to their combined experience and creativity. People who have tried pair programming testify that when working in pairs, they feel much greater confidence in their solutions and they enjoy the development process much more. Confidence and joy are two very good motivational factors in a creative job such as programming.

Continuous code reviewing results in quality code. During the process of coding, bugs still appear, but are found and fixed early on in the development process.

The following conditions need to be met prior to applying pair programming technique:

- Pair members have to agree with pair programming concepts
- They have to be willing to work with a particular partner and the communication between them has to be good
- Technology knowledge of developers in a pair has to be comparable. If one experienced and one inexperienced member work together in a pair, they can establish a good mentoring relationship, but their different levels of experience are not conducive to productive pair programming.

3. CASE STUDY DESCRIPTION

3.1 Case study goals

We wanted to use this case study of complex module development to gain practical experience with pair programming. Based on that experience we wanted to evaluate the usability of pair programming for the current long term project or other future company projects. The evaluation should be done for all specific project activities: analysis, application design, UI design, coding, testing, refactoring and bug fixing.

3.2 Module functionality

In this case study we developed a membership module with applied policies and rights. Both users and groups can be members of an unlimited number of groups. For each group and user, policies define which actions are generally allowed, denied or not set. Besides policies, it is possible to set rights to perform actions on specific objects in the application. Just as policies, rights can be allowed, denied or not set. If a policy or right is not defined for the specific object, it is inherited from the parent object. If a policy or right is not defined for a user or group, it is inherited from the parent group.

Membership module was supposed to work fast enough in order not to disturb normal working of the application.

Technology used in the whole project, including this module was: MS VB.NET & MS SQL server.

3.3 Acceptance of pair programming

Participants in this experiment were two developers that have been involved in the project from its beginnings, but did not have any experience in pair programming, neither in this project, nor any other.

One of them was keen to experience pair programming while the other did not see any advantages at the beginning of the experiment. It can be explained by the fact that the two developers were familiar with the technology used to a different extent. Still, no one refused to participate in the experiment, because it was a group decision to try it and it has been defined as an experiment with limited duration – until the membership module is finished.

3.4 Working environment

Most of the time, pair was sitting alone in the room. Occasionally, the team leader worked in the same room, but didn’t interfere.

3.5 Duration

Experiment was conducted during the period of two calendar months. During this period, no other tasks were taken by these individuals – no bug fixing and no code refactoring of other modules, so the pair could work on the task the entire day. Pair programming is a skill just as programming itself, and as any other skill, the more it is practiced, the better it is developed. It took about two weeks before pair developers started to feel an acceptable level of comfort in their ability.

4. RESULTS

Some results in this section were measured objectively (working hours, number of reported bugs). Other results are observations of pair developers and the team leader. In the second case, opinions before and after the experiment were compared.

Other colleagues were informed about the experiment. No one told them they shouldn’t disturb the pair. Still, that is what happened - the amount of disturbance coming from the working environment decreased. This is one of the reasons the pair seemed to be more productive in this working mode.

The other reason is the working behavior of the individuals in the pair. Development velocity is never 100%, but usually somewhere between 70-80% (Pilone & Miles, 2008). However, when developers worked as a pair, they were more focused on the task during the work day. It is exhausting to maintain such a high level of concentration during the day, and this is why the number of hours worked during the experimental period slightly dropped.

Developers have shown greater willingness to try pair programming at the end of the experiment then at the beginning. This can be explained by the feel-good factor when working in pairs (Muller & Padberg 2004). At the beginning, they felt uncomfortable because their partner was observing them all the time. But at the end, they were used to working together and felt comfortable working in pairs. They also described it as a very positive experience.

The quality of code produced in this experiment was greater than the code for other modules produced individually by the same developers forming the pair. In fact, this module was the module with the smallest number of reported bugs in the entire application. Furthermore, both individuals felt comfortable in module code refactoring performed later on, during the project. Both agreed that the code of this module with the completely implemented functionality was smaller and easier to read then in case of some other modules (Bipp et al., 2008).

Pair developers, even those working in the same team from the begging of the project, had different working styles before the experiment. After the experiment, their styles were still different, but much more similar than before, so developers understood code written by each other much better.

5. CONCLUSION

We are aware that this experiment would have been better if more people, ie. pair programmers had participated in it by doing different tasks of various levels of complexity. Further studies should include the development of the same modules by pair programmers and single developers in the same time to obtain more significant results. Even so, this case study was valuable to us. Analysis of the results of this experiment has shown that in our case it would be good to use pair programming for development of modules with complex business logic. In scenarios like this we think that among all project activities, greatest benefits can be achieved in analysis, application design, coding and refactoring.

In developing small, simple modules and activities such as user interface design, testing and bug fixing, single developers should be used.

6. REFERENCES

Bipp, T.; Lepper, A. & Schmedding, D. (2008). Information and Software Technology. Pair programming in software development teams – an empirical study of its benefits, Vol. 50, No. 3, (February 2008) page numbers (231-240), ISSN:0950-5849

McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction, 2nd edition, Microsoft Press, ISBN-10: 0735619670, ISBN-13: 978-0735619678, Redmond, Washington

Muller , M.M. & Padberg , F. (2004). 10th IEEE International Symposium on Software Metrics (METRICS'04), An Empirical Study about the Feelgood Factor in Pair Programming, pp. 151-158, ISBN: 0-7695-2129-0, Chicago, Illinois, USA, September 2004., IEEE Computer Society Washington, DC, USA

Pilone, D. & Miles, R. (2008). Head First Software Development, O'Reilly Media , ISBN-10: 0596527357, ISBN-13: 978-0596527358, United States

Roodyn, N. (2004). eXtreme .NET: Introducing eXtreme Programming Techniques to .NET Developers, Addison-Wesley Professional, ISBN: 0321303636, New Jersey, United States

Williams, L.; Kessler, R.R.; Cunningham, W. & Jeffries, R. (2000). Strengthening the Case for Pair-Programming Available from: http://collaboration.csc.ncsu.edu /laurie/Papers/ieeeSoftware.PDF Accessed: 2010-08-18

Williams, W.M. & Sternberg, R.J. (1988). Intelligence. Group intelligence: Why some groups are better than others, Vol. 12, No. 4, (October –December 1988) page numbers (351-77), ISSN: N/A