Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (3) 417-420 (12012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Pair Programming Evaluation in an Academic Environment

Z. HUSSAIN, A. KEERIO*, J. A. MAHAR**, M. A. SOOMRO

Quaid-e-Awam Engineering University, Nawabshah, Pakistan

Received 13th December 2011 and Revised 27th April 2012

Abstract: Agile methods particularly eXtreme programming (XP) and Scrum play an important role in software development. Software development companies and individual programmers have been increasingly adopting XP and Scrum methodology as a best choice for the development of application software. XP consists of many practices, among them pair programming is a key practice. A research study on pair programming has been conducted in the laboratory of the university. Three simple tasks have been given to 75 students from same age group. These participants were distributed in 30 pairing group (60 students) and remaining 15 students participated as solo programmer. During 6 pairing sessions we have critically analyzed all pairing and solo participants. We have found that a less number of compilation errors were done by pairing groups than solo programmers. The pairing groups have completed their tasks before solo programmers by taking less time. We have also analyzed that driver and navigator frequently switch their role, and switching role is mostly initiated by driver. Overall, pair programming practice was liked by pairing groups

Keywords: Agile methods, eXtreme Programming, Pair programming.

1. INTRODUCTION

In the early days of computing software development was not in organized form. Many large organizations had demand of software application programs due to heavy flow of information, as they were doing thousands of transactions on daily basis, and it was too difficult to manage this huge flow of information manually. To overcome this problem software industry tried to use different software development models. In early 1960's (Jerzy *et al.*, 2002) new tools and techniques were introduced to make software development simpler such as Waterfall model. This model gained popularity in early 1990's, but Waterfall model is heavy weight (Jerzy *et al.*, 2002) and time consuming. This model is not an incremental based, where team of programmers collects the requirement from the customer before starting of the project and afterwards delivers a full and final version of software program to the customer. A customer is not collocated with the team of programmers. Agile software development is lightweight (Michael, *et al.*, 2001) introduced in late 90's, and gained popularity in software community after 2001. Few agile software development methodologies (Mohammadi *et al.*, 2008)such as eXtreme programming (XP), Scrum, Lean, Crystal, Dynamic System Development Method (DSDM), Adaptive Software Development (ASD), and Feature Driven Development (FDD) are the well known choices for software community to select any one of them to improve the standard of organization and manage a successful software project delivery very efficiently. All agile methods particularly XP are iterative (Sean 2008) and incremental based, they are used to design software programs more easily and quickly through customer collaboration and feedback. XP a well known methodology (Mohammadi *et al.*, 2009) has values, principles, and practices, some of the XP practices (Karlheinz 2009) are pair programming, onsite customer, simple design, planning game, small releases, refactoring, 40-hrs/ week, metaphor, acceptance test and continuous integration. By using these practices a team of software developers is able to achieve objectives and deliver high quality software. XP and all other agile methods encourage team work, a team consists of 5 to 10 members with a full time customer, and usually a customer has a non developer role and is very critical in decision making. Onsite customer or customer involvement is the key practice of XP. In many real time software projects, a customer mostly remains collocated with a team and is maintaining regular contact with developers and end users. Customer is mostly appointed by top management of the organization or rarely by project funders, and should be expert in his/her domain and has sufficient knowledge about project. Usually a user story (Angela 2004) is written and prioritized by him and he is fully authorized to change any user story at any time without any intimation. A team of programmers starts to break the stories into tasks and tries to complete all task within a scheduled time frame. An estimated highest prioritized user story should be completed first. Each user story will be tested by customer before integrating program. Those stories that do not pass the test are sent back to the team to work on them again or to fix the bugs.

++Corresponding author, Email: ayazkeerio@hotmail.com Cell No. +92 3412802830

* Department of IMCS, University of Sindh, Jamshoro, Pakistan

**Department of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan

Pair Programming

Pair programming (Angela2000) is a key practice of XP. This practice has been commonly used by a team of software developers in software development for last couple of years. The successful adaption of pair programming results in increasing software productivity, high code quality, job satisfaction and reducing code compilation time (Irina *et al.,* 2008) Pair programming is a technique in which a pair (two programmers) is working with one mind and responsible for their code/artifact. Pair of programmers mostly perform task on one computer (i.e., one keyboard, one mouse and one monitor) (Diana *et al.,* 2008). The role of one programmer is the driver (Llenia 2011). He is the main programmer who is responsible for typing a code and also runs the same code. The driver has a complete control on keyboard and mouse, he is thinking about the tactics to solve a complex task. The second programmer plays a navigator (Sallyann 2006) role, navigator is bound to continuously review the code and concentrate on to remove the errors from the program/code. In this way all code is constantly reviewed. Both programmers are equally owner of the artifact (code, algorithm and design) and none of them makes responsible to other for defects. They are also equally responsible to review, update and refactor the unnecessary code at any time without any hesitation. The code written by pair is simple, clean, short and having a good standard as compared to code written by programmer alone. A pair usually works together mostly on the same computer up to the completion of user story. A small user story/task is most probably completed in few hours within a day. During the working on the user story pair of developers may switch (Laura *et al.,* 2011) their roles. A switching between pair is more challengeable (Ruud *et al.,* 2007 when two programmer having a different level of experience. Driver becomes the navigator and vice versa. Driver and navigator continuously communicate with each other, they exchange their ideas and improve their knowledge. A pair of young developers may produce more unique results as compared to one experienced programmer. Two skilled programmers having some technical background and good programming skill such as c++, java and c# produce better quality of code that is much simple and faster than programming alone (Keeling, 2011). Pairs usually perform well as compared to solo programmer when task is challengeable and new (Llenia 2011). It is some time felt very difficult to work on the complex user story when one programmer is highly skilled and experienced and on the other hand the second programmer is new in the software development. The skilled programmers are worried about their new untrained partner because the progress may be slow down due to his mis-matching of knowledge and skill (Laura 2006). They prefer to work with the partner having same skill and experienced. Personality traits of pairs play an important role for the success of pair programming (Freudenberg 2007). If the pair of programmers is compatible, and having a good personality then they are well suitable for each other. Compatible partner feels comfortable and shows confidence to work with other partner. Two types to measure the adoption of pair programming is personality traits (Freudenberg 2007). quality of code.

2. MATERIALS AND METHODS

Three simple and easy programs were selected by the authors of this paper for the students from two different departments, computer systems engineering department and department of computer science, Quaid-e-Awam Engineering University, Nawabshah, Pakistan, to conduct the practical experiment in laboratory. Total 75 students were selected. The selection has been done on the basis of the marks obtained by the students in C, C++ and Java programming courses. These 75 students participated in a program to perform all these three tasks/programs. None of the students has any knowledge about any agile methodology particularly about XP. These 75 students were from same age group i.e., about 20 to 22 years old. The participants were distributed into 30 pairing groups (60 students) and remaining 15 students have performed as a solo programmer. We conducted 6 programming sessions in about 1 hour duration. All the available facility in laboratory was equally provided to all the participants as they were performing in a pair or solo. The following tasks were given to the participants during laboratory session.

Task 1: Write a program to calculate the sum of following series
1/1 + 3/4 + 6/8 +9/12 + 12/16 + 15/20
Task 2: Write a program to calculate the factorial of any positive integer
Task 3: Write a program to display a Fibonacci sequence up to 89

3. RESULTS AND DISCUSSION

Fig.1 shows the average number of compilation errors. The data has been collected from six programming sessions, each session was about one hour length. Both boys and girls students participated in pair and solo programming sessions. Based upon the results from the **(Fig. 1)** we found that less number of compilation errors was detected in pair programming sessions as compared to solo programming. In pair programming one programmer was in navigator role who was continuously reviewing the code of the driver. He was also responsible to pointing out spelling mistakes, syntax errors and calling the wrong methods, these results are consistent with (Kelli *et al.,* 2005) A less number of errors increased the high quality of

code (Rafael 2005) Duque and Bravo have suggested in their study that during coding pair of programmers made a lower number of compilation errors. Our results are also consistent with them. A pair programming is a prominent practice of XP and is valuable for better code quality and accuracy.

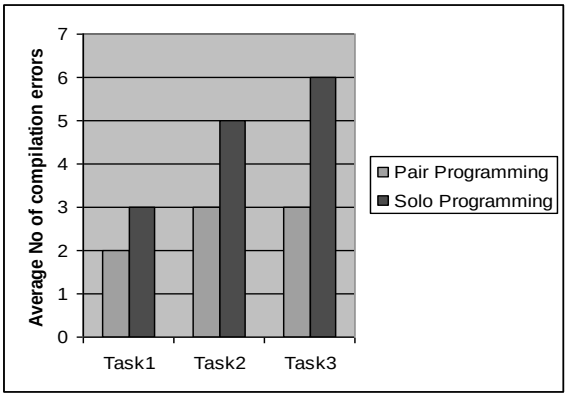

Fig 1: Number of compilation errors per task for pair and solo programming.

Fig. 2 shows the average number of conversation or intra communication made by two programmers (pair) during writing of code. This verbal communication was not more than three minutes at one time. Driver and navigator frequently communicated with each other. A driver was contributing more than the navigator, these results are consistent with **(Fig. 2)** shows that 80% of intra communication is made to

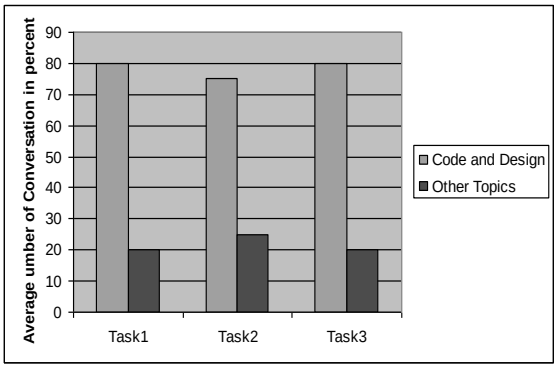

Fig. 2: Average number of conversations per task for code and other topics

discuss on code, design and algorithm of task and remaining 20 % conversation is made to discuss some other academic issues such as home work, assignment and examination issues. Few benefits such as high quality code and team building are achieved through communication when working in pair.

Fig.3 shows the switching role between driver and navigator. Driver and navigator both are equally owner of the code. Driver has full control on keyboard and mouse (s) he may use both input devices (keyboard and mouse) for writing and executing the code. (Llenia 2011) suggest in their study that pair programming is highly collaborative practice and many tasks can be performed by pair programmers. During the pair programming session driver and navigator may exchange their role at any time. This switching role usually happens after lunch break or work (Laura 2006). **(Fig. 3)** shows that both programmers frequently exchanged their role. On the basis of result we found that 80% of switching role is initiated by a driver and only 20 % is initiated by navigator. A switch is mostly initiated by driver, these results are consistent with (Laura 2006). The driver offered both input devices (keyboard and mouse) physically or verbally to the navigator. In the same way switch was also initiated by navigator when (s) he asked for input device (keyboard and mouse) physically or verbally from driver.

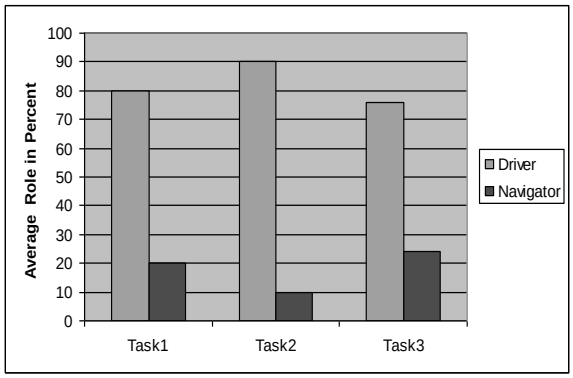

Fig. 3: Switching role between driver and navigator

Fig. 4 shows average time to perform each task for Pair and Solo programming. Forever and Ingalls have reported in their study that the estimation of user story is one pairing session, and Solo programmers spent two programming sessions on it. (Rafael 2005) have reported in their study that pair of programmer takes lower time than Solo programmers. The data has been collected from six pairing sessions. **(Fig. 4)** shows that all these three tasks are performed by pair of programmers in less amount of time as compared to solo programming. These results are consistent with (Rafael 2005). We also asked opinions from the pairs about their pair programming experience. Almost all liked this practice of XP.

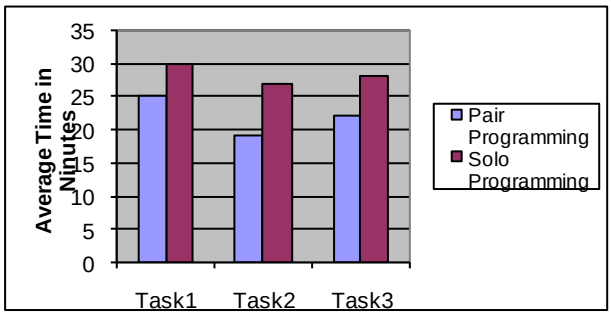

Fig. 4: Average time to perform each task for pair and solo programming

4. CONCLUSION

The agile software development particularly XP is famous in software developers community. Many developers throughout the world are adopting XP. Developers use many XP practices. In this paper we conducted research study on the pair programming practice. The results show that 40 % less number of compilation errors are done by pair programmer while performing the tasks as compared to solo programmer. High quality code and team building are achieved through intra communication when working in pair. During the pair programming session 80% to 85% pair programmers discussed about code and design, remaining 15% to 20% have discussed on academic issues. We have also analyzed the switching role where average 83% switching role is initiated by driver and 17% switching role is initiated by navigator. Pairs of programmers take lower amount of time to perform a task than solo programmer. Pair programming practice resulted in better code quality and accuracy and pairs exchanged their role frequently for sharing the knowledge, ideas and tactics. In future more research studies will be conducted for gaining the insight of the other XP practices

REFERENCES:

Brayant S., P. Romero, and B. du Boulay, 2006) "The Collaborative Nature of Pair Programming", in lecture notes on computer science (LNCS), vol. **(4044):** 53-64.

Chao, and G.. Atli, (2006) "Critical Personality Traits in Successful Pair Programming", in proceeding of agile development conference, 89-93.

Cohanv S, (2008) "Successful Customer Collaboration Resulting in the Right Product for the End User", in proceeding of agile development conference, 284-288.

Diana I. C., A. Sillitti, and G. Succi, (2008) "Investigating the Usefulness of Pair– Programming in a Mature Agile Team", in lecture notes of business information processing (LNBIP), vol. **(9):** 127-136.

Freudenberg S. (2007) (nee Bryant), P. Romero, B. du Boulay, "Talking the talk': Is intermediate – level conversation the key to the pair programming success story?" in proceeding of agile deve. conference, 84-91.

Jerzy R. N., B. Walter, and A. Wojciechowski, (2002) "Comparison of CMM Level 2 and eXtreme programming", in lecture notes on computer science (LNCS), vol. **(2349):** 288-297.

Kircher, M. P. Jain, A. Corsaro, and D. Levine, (2001) "Distributed eXtreme Programming", second international conference on extreme programming and agile processes in software engineering, 66-71.

Karlheinz K. (2009) "Customer and User Involvement in Agile Software Development", in lecture notes of business information processing (LNBIP), vol. **(31):** 168-173.

Keeling, M. (2010) "Put It to the Test: Using Light weight Experiments to Improve Tea m Processes", in lecture notes of business information processing (LNBIP), vol. **(48):** 287-296.

Kelli M. S., M. Droujkova, S.B . Berenson, L. Williams, L. Layman, (2005) "Undergraduate Students Perceptions of Pair Programming and Agile Software

Kai S., E. Knauss, K. Schneider, and M. Becker, (2010) "Towards Understanding Communication Structure in Pair Programming", in lecture notes of business information processing (LNBIP), vol. **(48):** 117-131.

Methodologies: Verifying a Model of Social Interaction", in proceeding of agile development conference, 323-330.

Laura P., J. Segal, H. Sharp, and J. van der Linden, (2011) "Collaboration in Pair Programming Driving and Switching", in lecture notes of business information processing (LNBIP), vol. **(77):** 43-59.

Llenia F., A. Sillitti, G.. Succi, and J. Vlasenko, (2011) "Analysing the Usage of Tools in Pair Programming", in lecture notes of business information processing (LNBIP), vol. **(77):** 1-11.

Laura P., (2010) "pair Programming: The Choice of a Partner", in lecture notes of business information processing (LNBIP), vol. **(48):** 395-396.

Mohammadi, S., B. Nikkhahan, and S. Sohrabi, (2009) "Challenges of User Involvement in Extreme Programming projects", in proceeding of international journal of software engineering and its application, vol. **(3):** 19-32.

Martin, A. R. Biddle, J. Noble, (2004) "The XP Customer Role in Practice: Three Studies", in proceeding of agile development conference, 42-54.

Ruud W. and I.V. Dijk, (2007) "Mutli- Tasking Agile Projects: The Pressure tank", in lecture notes on computer science (LNCS), vol. **(4536):** 231-234.

Troy F., P. Ingalls, (2006) "The Pairing Session as the Atomic Unit of Work", in proceeding of agile development conference, 65-169.