Preliminary Analysis of the Effects of Pair Programming on Job Satisfaction

Giancarlo Succi
Center for Applied Software Engineering
Free University of Bolzano - Bozen
Piazza Domenicani 3
I-39100 Bolzano-Bozen, Italy
+39(0471)315-640
Giancarlo.Succi@unibz.it

Witold Pedrycz
Department of Electrical and Computer Engineering
University of Alberta
ECERF
Edmonton, Alberta, Canada T6G 2G7
+1(780)492-4661
.Witold.Pedrycz@ee.ualberta.ca

Michele Marchesi
Dipartimento di Ingegneria Elettrica ed Elettronica
Università di Cagliari
p.zza D'Armi
I-09123 Cagliari, Italy
+39(070)675-5757
michele@diee.unica.it

Laurie Williams
Department of Computer Science
North Carolina State University
1010 Main Campus Road, 407 EGRC
Raleigh, NC 27695, USA
+1(919)513-4151
williams@csc.ncsu.edu

ABSTRACT
Pair programming is one of the most controversial parts of XP. Claims are mostly based on anecdotal evidence and limited experimentation performed in classroom settings.

This paper reports the preliminary results of an analysis of the effects of pair programming on job satisfaction. A questionnaire on pair programming techniques has been compiled and posted on the web.

108 responses have been collected from around the world.

The preliminary results evidence a very positive effect of pair programming on job satisfaction.

Keywords
Extreme programming, pair programming, job satisfaction, quasi-experimentation

1 INTRODUCTION

Is XP yet another fad? This is the Hamletic question for software developers of the new millennium, especially those working in very dynamic markets, such as web-based systems, office automation tools, etc.

Sometimes the question is explicitly posed. Sometimes it is politely masked under other forms. Still, apart from a handful of XP and Agile Methodologies evangelists and an equally small troop of anti-XPers, most people live in the dilemma.

It is not the task of this preliminary paper to address this dilemma. Also, because we think that it is not correctly posed.

XP is a set of practices often requiring customization (Beck, 2000). It is quite difficult to identify homogeneous groups of developers practicing XP in the same way. Therefore, we decided to concentrate our effort on those practices that appear most controversial.

In the case of this work, we have started with pair programming, that is, having two developers working together on the same code, in front of the same monitor, one typing and the other telling what to type.

Pair programming itself may have a variety of impacts on the overall production system. There are claims of improved product quality, better reliability, shorter learning curve for new developers, lower sensitivity to turnover, shorter time to market and higher job satisfaction of developers.

This paper analyses the latter, higher job satisfaction of developers. In an environment where, despite the crisis occurring after the 11th of September, 2001, still experiences a critical lack of developers, attracting a retaining a fully satisfied workforce is of extreme importance.

We think that satisfied workers are also more productive and build better systems. However, proving a relationship between pair programming and job satisfaction does not imply at all any relationship between pair programming and any of the other expected effects of it, especially quality and productivity. We have to keep this well in mind, both to avoid conclusions, which would be scientifically wrong, may induce companies in wasting resources, and may stop the other, required research to conduct on the "other" effects of pair programming.

We follow the guidelines of GQM, the groundbreaking Goal-Question-Metrics approach by Vic Basili (1995). We first set our general goal using comprehensive template, then we define a series of questions to determine whether we are achieving the goal, and lastly we collect several metrics to answer the questions.

Section 2 presents the design of the experiment. Section 3 outlines the results of the preliminary analysis. Section 4 draws some conclusions, outlining the lines for future research.

2 DESIGN OF THE EXPERIMENT

As mentioned, we use the GQM templates. They support the appropriate definition of the overall goal, avoiding inconsistencies and ambiguities. Here below there is our

goal.

- *Analyze* **pair programming**
- *For the purpose of* **evaluating it**
- *With respect to* **job satisfaction**
- *From the view point of* **software developers**
- *In the context of* **development of software systems**

To produce a valuable, original contribution to the understanding of pair programming, we need to analyse how "real" developers in general apply such practice.

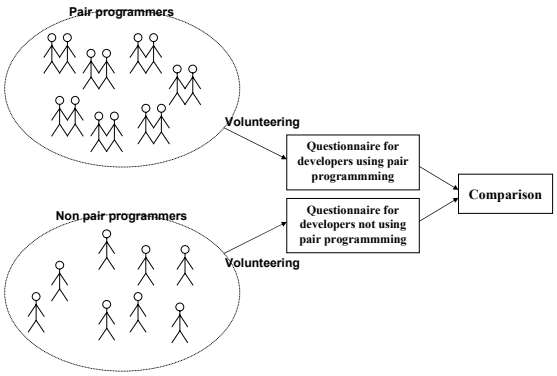

Figure 1: Design of the experiment

As it would be unfeasible to run a formal experiment on a wide number of developers, we have decided to use a "quasi-experimental" approach (Figure 1) (Campbell and Stanley, 1966).

Based on the GQM goal, we have developed questions with answers on nominal or ordinal scales. These questions have been used to design two questionnaires, one for developers using pair programming and one for developers not using pair programming.

Two PHP pages have been developed, containing each one questionnaire, and then they have been posted on the web site of the Software Engineering Group of the University of Alberta.

Volunteers have been recruited via conference announcements (ICSE 2001, XP2001, and XP Universe), mailing lists (SEWORLD and SEA), newsgroups (comp.software-eng), and private networks of researchers and colleagues.

The data have been collected on the period June 2001-December 2001. Then the data have been analysed.

Altogether, 21 questions and 27 metrics have been developed. The questions are listed here below. The questions are omitted for space reasons.

Q1. What are the phases of the PP process?

Q2. What are the phases of the non-PP process?

Q3. Which phases are actually executed during the PP process?

Q4. Which phases are actually executed during the non-PP process?

Q5. What is the experience of the software engineer?

Q6. What is the software engineer's experience with the non-PP process?

Q7. What is the software engineer's experience with the PP process?

Q8. How well do the software engineers know what the PP process is?

Q9. What is the software engineer's experience/approach with design review?

Q10. What is the software engineer's experience/approach with code review?

Q11. What is the software engineer's experience/approach with unit testing?

Q12. What is the developer's opinion towards PP?

Q13. What is the developer's opinion towards Collective Code Ownership?

Q14. What is the developer's opinion towards using Coding Standards?

Q15. What is the developer's opinion towards adopting the PP process?

Q16. What factors are critical to success in PP?

Q17. What importance is attributed to individual work?

Q18. What is the developer using the PP approach's opinion towards his/her job?

Q19. What is the developer using the PP approach's opinion towards her/his working environment?

Q20. What is the developer using the non-PP approach's opinion towards his/her job?

Q21. What is the developer using the non-PP approach's opinion towards her/his working environment

Clearly, this kind of "quasi-experimentation" suffers from several drawbacks, including the following.

- There is no a priori insurance of an even distribution of the respondents, representative of the wider population of developers; on the contrary…
- … The respondents may be only those people strongly biased in favour or against pair programming;
- There is no check that the respondents fully understand the questions being posed;
- The statistical techniques employed are not powerful, presenting the risk of not being able to conclude anything significant.

3 ANALYSIS OF THE RESULTS

We had a total of 108 responses, evenly divided among the pair programming and the non pair programming groups -54 and 54.

While it is not possible to ensure the complete gener-

alizability of the results, we have assessed whether the two samples of developers using PP and developers not using PP were significantly different.

We have used non parametric analysis techniques taking into account the nominal and ordinal scale of the date. In particular, we have employed the Mann-Whitney U test, the Wilcoxon W test, and the Chi-square test.

As usual in Software Engineering, we have used α=0.05 as the threshold for the significance test.

Whenever needed, we have employed the technique of contrasts to determine the preferences between two partitions of a multi-valued set (Stevens, 1996).

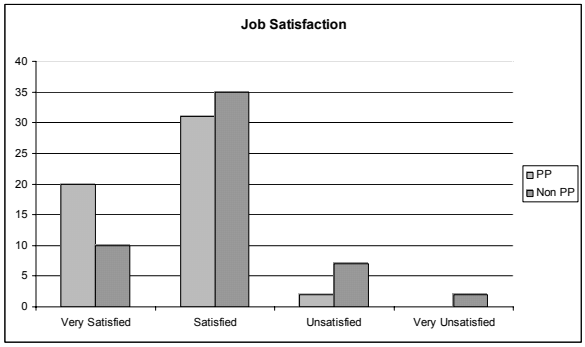

Figure 2: Distribution of job satisfaction

Altogether, we have found that no significant statistical demographic difference existed between the two groups in terms of:

- Gender
- Age
- Programming experience

We have then tested whether the developers using pair programming experienced higher job satisfaction than those not using pair programming. The null hypothesis – no difference between the groups, has been rejected by the chi-square test (α<0.5) (Figure 2).

Other tests have been performed on the groups to determine whether pair programming was the cause or it was the effect of other practices and environmental variables.

The variable considered included:

- Communications between departments
- Communications between developers
- Speed of communication of design changes
- Organization of meetings
- Workspace and office layout
- Lighting
- Noise
- Heating

There were no significant differences between communications between departments, workspace and office layout, lighting, noise, and heating.

Significant better results were found in the pair programming group for communications between developers, speed of communication of design changes, and organization of meetings.

Pair programming appear to influence significantly how development teams communicate internally and organize themselves. Pair programming has insignificant influences in the communications with other groups and in the working environment –there is no positive or negative "bias" of the management team toward pair programming groups as far as the working environment goes.

The regression between job satisfaction and communications between developers, speed of communication of design changes, and organization of meetings does not evidence any significant difference, reinforcing the conclusion that pair programming positively affect job satisfaction, in a significant way.

4 CONCLUSIONS

In this paper, we have presented the first results of a "quasi-experimental" study on the effects of pair programming on job satisfaction.

The work is based on a questionnaire administered via the Internet in the period June-December 2001.

108 answers have been analysed, 54 of developers using pair programming and 54 of developers not using pair programming. Given the nominal and ordinal nature of the data, we have used non-parametric tests.

It appears that pair programming has a significant, positive influence on the satisfaction of developers. This comes with increased communications between developers, speed of communication of design changes, and organization of meetings.

These findings do identify a positive aspect of pair programming. However, they do not support or defy the other claims on pair programming, such as the increased quality and reliability, the higher productivity etc.

Clearly, more research is required, at least at four levels:

- Extension of the results of the present work to more studies, if possible using formal experiments and larger datasets;
- Explorations of the "other" expected effects of pair programming
- Analysis of the effects of the other individual practices of XP
- Determination of the cross effects of multiple practices of XP together

The latter aspects is especially important, as there are claims that (some of) the practices are strongly linked one another. Only an empirical study would identify such cross effects.

A clear determination of the cross effects would then enable more suitable customizations and more informed

introductions of XP.

5 ACKNOWLEDGEMENTS

We thank Teresa Baldassarre for compiling the questionnaire, Jacob Bresciani for writing the PHP scripts to administer the questionnaires and to collect the results, Dagmar Morandell for the statistical analysis of the data, Wolfgang Polasek for the very valuable support in the statistical analysis, and all the respondents who made possible this work.

This work has been supported by the Alberta Software Engineering Research Consortium, the Canadian Natural Science and Engineering Research Council, and Nortel Networks.

REFERENCES

Basili, V. (1995) "Applying the Goal/Question/Metric Paradigm in the Experience factory" in *Software Quality Assurance and Measurement: A Worldwide perspective*, Chapter 2, pp 21- 44, International Thomson Computer Press

Beck, K. (2000) *Extreme Programming Explained – Embracing the Change*, Addison Wesley

Campbell, D.T., and J.C. Stanley (1966) *Experimental and Quasi-Experimental Designs for Research*, Rand McNally

Stevens, J. (1996) *Applied multivariate statistics for the social sciences*, Hillsdale.