Does Pair Programming Increase Developers Attention?

Ilenia Fronza, Alberto Sillitti, Giancarlo Succi, Jelena Vlasenko
Free University of Bolzano / Bozen
Piazza Domenicani – Domenikanerplatz, 3
I – 39100 Bolzano – Bozen, Italy
+39 0471 016138
{Ilenia.Fronza, Alberto.Sillitti, Giancarlo.Succi}@unibz.it, Jelena.Vlasenko@stud-inf.unibz.it

ABSTRACT
Pair programming is believed to improve quality and productivity in software system development. Yet, it is not clear whether and under what conditions this really occurs. In particular, it is interesting to determine whether pair programming has an impact on attention, which has been proven to have very beneficial effects.

A 10 month empirical study has been conducted for a large Italian manufacturing company, to determine if pair programming increases the attention level among its developers. The results strongly indicate that while working in pairs developers spend more time in directly productive activities also with a higher level of concentration and less switches to private tasks.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *Programming Teams.*

General Terms
Pair programming, Tools usage, Attention level.

1. INTRODUCTION
Pair programming (PP) is a practice in which two programmers work collaboratively at the same computer on the same task [2]. Recently, PP has been largely advocated as a technique that

1. reduces defect rate [8; 9; 16 and many more],
2. improves the design and the structure of the code [5],
3. increases productivity [3; 19],
4. shortens the time-to-market [23],
5. enhances knowledge transfer and team communication [7],
6. promotes job satisfaction [28],
7. facilitates integration of newcomers also reducing training costs [14].

However, other works have contradicted such claims. [22] and [17] have not found any positive effect of PP on development time. A large experiment conducted by [1] evidences that PP neither reduces the time required to correctly perform change tasks nor increases the percentage of correct solutions. [4] reports survey results from Microsoft about high skepticisms over pair efficiency. It is therefore essential to determine what are the roots of the benefits of PP, so it can be applied when it is most appropriate.

This study aims at this goal, and specifically assesses whether PP has a positive effect on raising the level of attention. There are large evidences that keeping a high level of attention while working, results in better work done faster and more effectively [21; 26]. There are already claims that PP reduces the number of interruptions [8]: both intrusions and self-initiated interruptions [20] are reduced for two reasons:

(1) pairing "keeps developers honest", increasing discipline and improving time management: programmers are less likely to skip writing unit tests, spend time web-surfing or on personal email, or other violation of disciplines, when they are working with a pair partner;
(2) other people are more reluctant to interrupt a pair than they are to interrupt someone working alone.

This experimental work has been carried out in the IT department of a large Italian manufacturing company that prefers to remain anonymous. The company practices spontaneous PP – that is, developers pair whenever they feel it is needed. The company wanted to know what were the underlying mechanisms by which PP could improve the overall development process and, specifically, the analysis of the variations of the levels of attention induced by PP. To this end, the work of the company has been analyzed with PROM [10; 25], an AISEMA (Automated In-Process Software Engineering Measurement and Analysis) system, which tracks non-invasively developers' activities, including the tools they use; suitable graphs, the Lean Graphs (L-Graphs) [27], have been used to analyze the collected data.

The level of attention of the developers has been measured using two classes variables:

- the amount of work that is devoted to directly productive activities, along the lines of the key ideas of lean management [24];
- the concentration that developers have on tools, measured by the permanence on development tools and by the frequency of being distracted to other activities.

The results of this investigation are that, while doing PP, developers:

1. spend more time in directly productive activities;
2. have higher level of attention on tools, switching less often between them;
3. move with lower frequencies to private tasks from directly productive activities.

The remainder of the paper is organized as follows. In Section 2, we present some related work; Section 3 describes the structure of the study. Section 4 reports on the results. In Section 5 we discuss limitations, conclusions, and future work.

2. RELATED WORK

2.1 Level of Attention and Quality of Work

Several studies from organizational and work psychologists evidence that keeping a high level of attention while working results in better and faster work [21; 26]. Moreover, there have been evidences that in the modern society people using computers are exposed to the risk of losing such attention [13] and this phenomenon is even more acute for software developers; limited attention can cause both lower quality in specific work tasks and an overall reduction of social ability [12]. This reduction may severely impact the interactions with customers and colleagues, with negative consequences on the overall development process.

Two major groups of means to increase the level of attention have been identified in [18] :

- external stimulus that keeps the attention level high;
- an internal cognitive control mechanism to keep the attention aligned with the priorities of the tasks to complete.

Research has primarily considered externally-driven intrusions on individual workers, but the nature of work, the work environment and the team configuration may all influence how workers handle both externally and self-initiated interruptions. Moreover, ethnographic observations indicate that interruption length, content, type, occurrence time, and interrupter/interruptee strategies differ for pair programmers versus solo programmers [6].

2.2 Analysis of Data from Tool Usage

As discussed in [27], information on usage of tools is very important to understand the software development process in a company. To visualize this information we use L-Graphs [27]. An L-Graph is composed by three kinds of elements:

- Node: a node represents a tool and the size of a node is proportional to the percentage of time spent in the corresponding tool.
- Arc: an arc represents the transitions between two tools. The arc can be mono-directional, if there is a one-way switching from one tool to another, and bi-directional, if there is a two-way switching between tools.
- Label: nodes and arcs are annotated. The label near each node contains the percentage of time spent in the tool and average time of permanence in it before switching to another one. The label near the arc provides information about the frequency to switch from another.

In this study we focus on how developers switch from one tool to another knowing also the average time they spend in a tool before switching and the frequencies of switching between the tools, all information reported in L-Graphs.

3. DATA COLLECTION

As mentioned, this industrial work has been carried out in the IT department of a large Italian manufacturing company. The work spanned a period of approximately 10 months. The team was composed of 15 Italian developers having from 10 to 15 years of programming experience. They all hold university degrees in computer-related areas.

The programming language used is mainly C# under Windows XP using Visual Studio as development environment. The team uses a customized version of Extreme Programming that was adopted about two years before the start of the study. In particular, they use weekly iterations, PP, user stories, planning game, collective code ownership, coding standards, and test-first. The company practices spontaneous PP: this does not mean that the management does not support PP, but that it leaves developers free to implement it. The company wanted to know what were the underlying mechanisms by which PP could improve the overall development process and, specifically, the analysis of the variations of the levels of attention induced by PP.

The company has a policy to record all the story points to measure the associated velocity. Before the start of each story point, developers specify the user story they work on. If they do PP, they also specify with whom they pair. The team works in an open space, where each member has his/her own personal workstation. Therefore, there is a significant amount of informal communication between developers.

We have collected the data non-invasively with PROM [10; 25], an AISEMA system. Developers had access to the collected data and were asked to constantly check its correctness. They never reported any inconsistency about the collected data, Therefore, the data set can be considered very reliable.

Furthermore, PROM allows to store headers of the visited web pages. We analyzed keywords presented in the accessed web pages and divided Browsing into two categories: Private Browsing and Business Browsing. Business Browsing represents all developers' activities when they use Internet for business purposes like searching for information they need for their work. Private Browsing represents all their private activities.

In this study we consider three variables:

- the relative amount of effort spent in directly productive activities, measured by the percentage of effort spent in Visual Studio;
- the average permanence in Visual Studio before switching to another tool, measured in seconds;
- the amount of switches from directly productive activities to to personal tasks, measured by the percentage of switches from Visual Studio to Private Browsing.

4. RESULTS

We found that the developers used 26 different tools, but only 9 of them were used regularly and by all the developers. These 9 tools absorbed more than 80% of the total effort both for Solo Programming and PP: Visual Studio, Browser, Outlook, Office, Excel, Management Console, Messenger, Remote Desktop, and Windows Explorer. Therefore, we focus only on them. Figure 1 contains the L-Graphs for Solo Programming (i.e., Solos) and PP.

In terms of the percentage of effort spent in Visual Studio, we notice that the developers working alone spend 34% of their time in this tool and when working in pairs 64% what is almost twice more. It is also interesting to note that the time spent in Private Browsing is halved.

For the average permanence in Visual Studio before switching to another tool, we found that the developers when they work alone stay in Visual Studio 28 seconds on the average and when they do PP 128 seconds. These results indicate that developers are definitely more focused on what they do when pairing. Actually, this is a result that can be extended for almost any tool – when working in pair people are more concentrated than when they

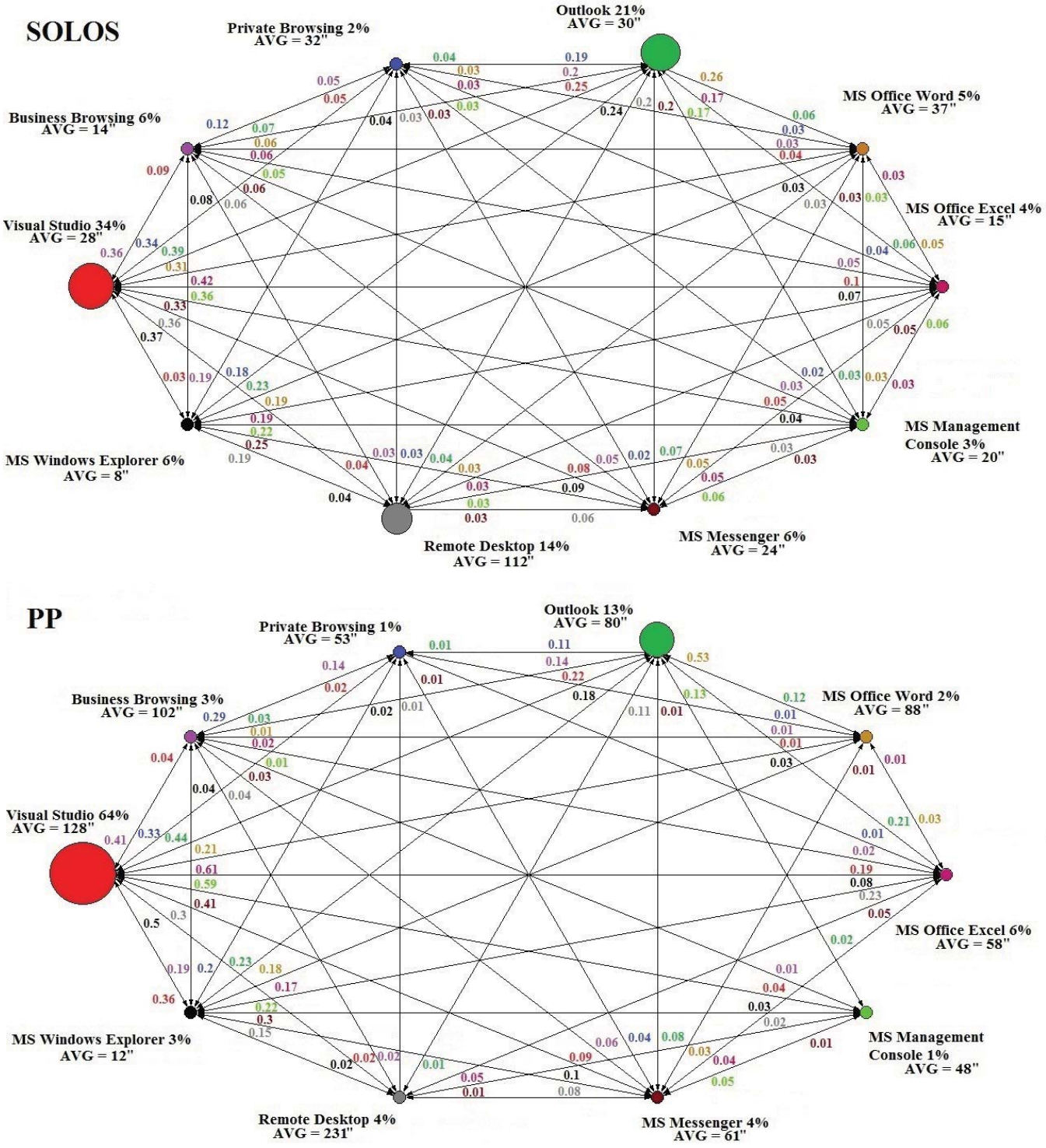

Figure 1: L-Graphs for Solo programing and PP

work alone.

The frequency of switching from Visual Studio to Private Browsing is 0.05 during Solo programming and 0.02 - less than half, during PP. These results indicate that PP decreases the amount of time spent for non-productive activities. This is also a further confirmation of the claims of [6] that interruptions are different in Solo programming and in PP.

5. CONCLUSIONS, LIMITATIONS AND FUTURE WORK

In this work we studied how an application of PP affects tools usage and level of attention. We found that the developers working in pairs exhibit higher levels of attention, that is, they are

more focused on their work, they switch less frequently between tools, and devote more time to software development that to other distracting activities.

This is clearly only one study based on one dataset, but the industrial source of the data, the length of the observation (10 month) and the soundness of the collected data (an AISEMA tool was used and the data was verified by the developers) make the results relevant. A further indication of the quality of the results is that these results were fed back to the company and were used by the company to take strategic decisions on the structure and the process of the development team – however, for confidentiality we cannot detail such decision here.

Our future work consists of collecting further data to determine the extensibility of our results.

6. REFERENCES

[1] Arisholm, E., Gallis, H., Dyba, T., and Sjoberg D.I.K. Evaluating pair programming with respect to system complexity and programmer expertise. *IEEE Trans. Softw. Eng.*, 33(2): pp. 65–86, 2007.

[2] Beck, K. Embrace Change with Extreme Programming. *Computer* 32, 10, pp. 70 – 77, 1999.

[3] Beck, K. and Andres, C. 2000. *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

[4] Begel, A. and Nagappan, N. 2000. Pair Programming: what's in it for me?. In *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement* (Kaiserslautern, Germany, October 09 - 10, 2008).

[5] Canfora, G., Cimitile, A., Garcia, F., Piattini, M., and Visaggio, C.A. Evaluating performances of pair designing in industry. *Journal of Systems and Software* 80, 8, pp. 1317-1327, 2006.

[6] Chong, J. and Siino, R. Interruptions on software teams: a comparison of paired and solo programmers. In *Proceedings of the Conference on Computer Supported Cooperative Wo*rk (Banff, Alberta, Canada, November 04 – 08, 2006).

[7] Chong, J. and Hurlbutt, T. The social dynamics of pair programming. In *Proceedings of the 29th international conference on Software Engineering*, pp. 354–363, Washington, DC, USA, 2007.

[8] Cockburn, A. and Williams, L. 2001. The costs and benefits of pair programming. In G. Succi and M. Marchesi, editors, The XP Series. *Extreme Programming Examined*, pp. 223-243. Addison-Wesley.

[9] Coman, I.D., Sillitti, A. and Succi, G. 2008. Investigating the Usefulness of Pair-Programming in a Mature Agile Team. In *Proceedings of the International Conference on Agile Processes and eXtreme Programming in Software Engineering.* Limerick, Ireland, June, 2008.

[10] Coman, I.D., Sillitti, A., and Succi, G. A case-study on using an Automated In-process Software Engineering Measurement and Analysis system in an industrial environment. In *Proceedings of the International Conference on Software Engineering* (Vancouver, Canada, May, 2009).

[11] Czerwinski, M., Horvitz, E., and Wilhite, S. 2004. A diary study of task switching and interruptions. In *Proceedings of Human factors in computing systems* (Vienna, Austria, April 24, 2004).

[12] Dukas, R. Behavioural and ecological consequences of limited attention, *Philosophical Transactions B of the Royal Society* 357 (2002), pp. 1539–1547.

[13] Falkinger, J. 2008. Limited Attention as the Scarce Resource in an Information-Rich Economy. *The Economic Journal* 118, 532 (2008), pp. 1596–1620.

[14] Fronza, I., Sillitti, A., and Succi, G. 2009. An Interpretation of the Results of the Analysis of Pair Programming During Novices Integration in a Team. In *Proceedings of the International Symposium on Empirical Software Engineering* (Lake Buena Vista, Florida, 15-16 October, 2009).

[15] Fronza, I., Sillitti, A., Succi, G., and Vlasenko, J. Understanding how novices are integrated in a team analysing their tool usage. In *Proceedings of the International Conference on Software and System Process* (Honolulu, Hawaii, 21-22 May, 2011).

[16] Heiberg, S., Puus, U., Salumaa, P., and Seeba, A. Pair-Programming Effect on Developers Productivity. In *Proceedings of the International Conference on Agile Processes and eXtreme Programming in Software Engineering* (Genova, Italy, May 26 - 29, 2003).

[17] Hulkko, H. and Abrahamsson, P. A multiple case study on the impact of pair programming on product quality. In *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*, pp. 495–504, New York, NY, USA, 2005.

[18] Lavie, N., Hirst, A., de Fockert, J.W., and Viding, E. 2004. Load theory of selective attention and cognitive control. *Journal of Experimental Psychology* 133, 3 (2004), 339–354.

[19] Lui, K.M. and Chan, K.C.C. When does a pair outperform two individuals? In *Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering*, pp. 225–233, Berlin, Heidelberg, 2003.

[20] McFarlane, D. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. Human-Computer Interaction 17, 1, pp. 63 – 139, 2002.

[21] Marchington, M. and Wilkinson, A. 2005. *Human Resource Management at Work*, Chartered Institute of Personnel & Development.

[22] Nawrocki, J. and Wojciechowski, A. Experimental evaluation of pair programming. In *Proceedings of the European Software Control and Metrics Conference (ESCOM)*, 2001.

[23] Phongpaibul, M. and Boehm, B. A replicate empirical comparison between pair development and software development with inspection. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, pp. 265–274, Washington, DC, USA, 2007.

[24] Poppendieck, M. and Poppendieck, T. 2003. *Lean software development: an Agile toolkit*. Addison Wesley.

[25] Sillitti, A., Janes, A., Succi, G., and Vernazza, T. 2003. Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data. In *Proceedings of the Conference on EUROMICRO,* Antalya, Turkey, September 2003.

[26] Stellman J.M., editor. 1998. *International Labour Office. Encyclopedia of Occupational Health and Safety*. Geneva: International Labour Office.

[27] Sillitti, A., Succi, G, and Vlasenko, J. Toward a better understanding of tool usage. In *Proceedings of the International Conference on Software Engineering,* Honolulu, Hawaii, May, 2011.

[28] G. Succi, W. Pedrycz, M. Marchesi, and L. Williams. Preliminary analysis of the effects of pair programming on job satisfaction. In *In Proceedings of the 3rd International Conference on Extreme Programming*, pp. 212–215, 2002.