Contents lists available at ScienceDirect

Int. J. Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Knowledge transfer in pair programming: An in-depth analysis☆

Laura Plonka[a], Helen Sharp[a,*], Janet van der Linden[a], Yvonne Dittrich[b]

[a] *Centre for Research in Computing, The Open University Milton Keynes, UK*
[b] *Software Development Group IT University of Copenhagen, Denmark*

ARTICLE INFO

Article history:
Received 24 April 2014
Received in revised form
25 July 2014
Accepted 1 September 2014
Communicated by Francoise Detienne
Available online 16 September 2014

Keywords:
Pair programming
Knowledge transfer
Interaction analysis
Cognitive apprenticeship
Qualitative analysis

ABSTRACT

Whilst knowledge transfer is one of the most widely-claimed benefits of pair programming, little is known about how knowledge transfer is achieved in this setting. This is particularly pertinent for novice – expert constellations, but knowledge transfer takes place to some degree in all constellations. We ask "what does it take to be a good "expert" and how can a "novice" best learn from a more experienced developer?". An in-depth investigation of video and audio excerpts of professional pair programming sessions using Interaction Analysis reveals: six teaching strategies, ranging from "giving direct instructions" to "subtle hints"; and challenges and benefits for both partners. These strategies are instantiations of some but not all teaching methods promoted in cognitive apprenticeship; novice articulation, reflection and exploration are not seen in the data. The context of pair programming influences the strategies, challenges and benefits, in particular the roles of driver and navigator and agile prioritisation which considers business value rather than educational progression. Utilising these strategies more widely and recognizing the challenges and benefits for both partners will help developers to maximise the benefits from pairing sessions.

1. Introduction

"Two heads are better than one" is a common idiom referring to the advantages of collaborative work. The value of collaboration is explicitly encouraged in software development through a practice known as pair programming. Pair programming (PP) is a software development technique where two developers work closely together to solve a development problem (Williams and Kessler, 2002; Beck, 1999).

Several benefits of PP have been claimed including improved understandability and maintainability of code and design (Vanhanen and Korpi, 2007; Vanhanen and Lassenius, 2007), decreased defect rates (Phaphoom et al., 2011; Jensen, 2003; Pandey et al., 2003; Cockburn and Williams, 2001; Phongpaibul and Boehm, 2006) and knowledge transfer (Luck, 2004; Katriou and Tolias, 2009; Pandey et al., 2003; Sanders, 2002; VanDeGrift, 2004; Vanhanen and Korpi, 2007; Vanhanen and Lassenius, 2005; Vanhanen et al., 2007; Williams, 1999). This paper focuses on knowledge transfer in PP. Indeed, Salinger et al. (2013) have identified PP roles other than driver and navigator, including one called task expert which brings in task expertise relevant to the session. This acknowledges the fact that all sessions include some level of knowledge transfer.

Most software development teams are composed of developers with different knowledge levels of some kind, including different programming experiences, different domain expertise and knowledge about different technologies. PP is one way to share their knowledge with other team members while also achieving meaningful work. In some cases, knowledge transfer is the explicit goal of a PP session (Plonka et al., 2012). This is common when a more experienced developer teaches a less experienced developer, for example, to bring new staff up to speed (Belshee, 2005; Williams et al., 2004). However, given that developers never have identical knowledge, a certain degree of knowledge transfer would be expected within every PP constellation.

Pairing with someone who has a different knowledge level can be problematic (Begel and Nagappan, 2008; Cao and Xu, 2005) and developers tend to interact differently in this situation in comparison to pairing with other developers with similar knowledge levels (Chong and Hurlbutt, 2007; Cao and Xu, 2005). For example, Plonka et al. (2012) showed that less knowledgeable developers (novices) can disengage in PP sessions and can sometimes not follow their more knowledgeable partner (expert).

Although knowledge transfer is widely reported as a benefit of PP, there is currently not much insight into how developers approach this in practice nor how knowledge transfer can be improved. What does it take to be a good "expert" and how to learn best as a "novice"? What are the challenges? Here, we

☆This paper has been recommended for acceptance by Francoise Detienne.
* Corresponding author.
E-mail addresses: Laura.Plonka@open.ac.uk (L. Plonka), Helen.Sharp@open.ac.uk (H. Sharp), Janet.VanderLinden@open.ac.uk (J. van der Linden), ydi@itu.dk (Y. Dittrich).

http://dx.doi.org/10.1016/j.ijhcs.2014.09.001
1071-5819/© 2014 Elsevier Ltd. All rights reserved.

present an in-depth investigation of knowledge transfer in professional PP sessions to address the following research questions:

- RQ1: What teaching strategies do developers use in pair programming?
- RQ2: In which ways do the roles of driver and navigator influence knowledge transfer in pair programming?
- RQ3: What challenges do developers with different knowledge levels face when pairing together?

These three questions are addressed through a qualitative analysis (using Interaction Analysis (Jordan and Henderson, 1995)) of video recordings of professional developers working together on their day to day tasks. As a result, we identified a set of teaching strategies and behaviours that are related to the roles of driver and navigator and influence teaching and learning, together with associated challenges and benefits for both pairing partners. An increased awareness of working practices for knowledge transfer in PP will help developers to maximise the benefits from such sessions.

The remainder of the paper is organized as follows. Section 2 overviews existing research on knowledge transfer in PP. In Section 3, we present the research methodology including data collection and analysis approach, followed by the findings of this study (Section 4). In Section 5, the findings are discussed with respect to existing literature and Section 6 discusses the limitations of the study. The last Section 7 presents conclusions and implications for developers.

2. Knowledge transfer in pair programming

The positive effect of PP on knowledge transfer, no matter what may be the knowledge levels of the developers, is widely acknowledged across a range of studies in industry (Luck, 2004; Katriou and Tolias, 2009), (Vanhanen et al., 2007; Vanhanen and Korpi, 2007; Pandey et al., 2003) and academia (Sanders, 2002; Vanhanen and Lassenius, 2005; VanDeGrift, 2004; Williams, 1999). Knowledge transfer is also one of the main perceived benefits according to two surveys: Schindler (2008) surveyed developers and managers in 42 Austrian companies; and Begel and Nagappan (2008) conducted a web-based survey of 487 Microsoft developers. Three industrial case studies (Luck, 2004; Vanhanen and Korpi, 2007; Vanhanen et al., 2007) report more detail on developers' perceptions. In (Luck, 2004), developers report that PP increased their knowledge of the code and in (Vanhanen and Korpi, 2007), developers report increased knowledge of the software system. Gaining knowledge about development tools, work practices, refactoring old code, new technologies and programming languages are all perceived benefits reported in (Vanhanen et al., 2007).

Belshee (2005) suggested very frequent changes of the pair constellation to promote fast knowledge transfer and to spread knowledge among different team members. Pandey et al. (2003), suggests that this can reduce project risk because multiple developers are familiar with the code and there is less reliance on one individual. Increased flexibility also means that developers can pick up a variety of different tasks. For example, Hodgetts (2004) reports on one team that had only one database expert, but too much work for one expert. When this caused a bottleneck, the team decided to use PP to spread the database knowledge among developers. They learned quickly through pairing with the database expert and were then able to do database tasks by themselves.

PP has also been studied in the context of training and mentoring, but not always with a positive effect. For example, in the context of developing firmware for processors, Greene (2004) found that the training effect of PP was not as high as expected, which may be due to the very specialized and complex domain knowledge needed in that context. On the other hand, Williams et al. (2004) investigated PP for mentoring and hence focused on pair constellations with different levels of expertise. They examined the relationship between PP and Brooks' Law[1] based on a survey and a case study. They found that PP reduced the mentoring needed per day from 37% of a developer's time to 26% of their time, and that PP reduced the time for a developer to be independently productive from 27 to 12 days.

The developers' view of combining different knowledge levels when pairing was investigated by Jensen (2003) and Vanhanen et al. (2007), Vanhanen and Lassenius (2007). Jensen, (2003) found that pairing developers with similar expertise was counter-productive, while Vanhanen and Lassenius (2007) found two good partner combinations: when the pair consists of a senior and a junior developer; or partners have complementary knowledge.

When asked about the challenges of PP, developers surveyed by Begel and Nagappan (2008) perceived working with someone with different skills as one of the main challenges. Williams and Kessler (2002) also point out that pairing experts and novices can be problematic. Novices can slow down experts and some experts might not have a mentoring attitude.

One study by Cao and Xu (2005) examined the interactions of pairs in more detail according to their expertise. They assigned students according to expertise and found that the expert asked for the novices' opinions frequently at the beginning of the session but stopped asking after realising that they did not get valuable information.

Although there is some evidence that pairing developers with different knowledge is useful but challenging, there is currently a lack of understanding about what interactions take place to achieve knowledge transfer and what challenges developers face.

3. Research design

It is known that people working jointly on a computer use a combination of gesture, language and screen object manipulation to construct an understanding of the problem (see (Roschelle and Clancey, 1992) for example). In PP developers work closely together on one computer and all these aspects needs to be considered when analysing knowledge transfer between the developers. For this study, we chose a data gathering approach that captures rich data about the PP sessions and an analysis approach that allows for a detailed investigation of how human beings interact with each other, and with objects in their environment (both verbally and non-verbally) (Jordan and Henderson, 1995).

3.1. Data gathering

Different aspects of the PP session were captured by using a combination of data gathering methods

- Audio and video recordings were used to record the *developers' interactions* during their PP sessions: audio recordings of all verbal communication; a video of the programmers; and a full-resolution recording of the screen, showing the code and capturing the developers' computer activities. These were fully synchronized into a single video file (see Fig. 1).

[1] Brooks Law says that “adding manpower to a late software project makes it later” (Brooks, 1975).