On Knowledge Transfer Skill in Pair Programming*

Franz Zieris, Lutz Prechelt

Institut für Informatik
Freie Universität Berlin
Takustraße 9
14195 Berlin, Germany
{zieris,prechelt}@inf.fu-berlin.de

Abstract: *Context:* General knowledge transfer is often considered a valuable effect or side-effect of pair programming, but even more important is its role for the success of the pair programming session itself: The partners often need to explain an idea to carry the process forward. *Goal:* Understand the mechanisms at work when knowledge is transferred during a pair programming session; provide practical advice for constructive behavior. *Method:* Qualitative data analysis of recordings of actual industrial pair programming sessions. *Results:* Some pairs are much more efficient in their knowledge transfer than others. These pairs manage to (1) not attempt to explain multiple things at once, (2) not lose sight of a topic, (3) clarify difficult points in stages. *Conclusions:* Pair programming requires skill beyond software development skill. To be able to identify knowledge needs and then push such knowledge to or pull it from the partner successfully is one aspect of such skill. We characterize a number of its elements.

The agile practice of pair programming has been subject to many small-scale empirical evaluations, often in the form of controlled experiments in comparison with solo developers, which produced remarkably unstable results [HDAS09]. We conjecture that one reason for the high results variance lies in ignoring pair programming skill (as opposed to technical software development skill) as a factor; we consequently pursue a research program for analyzing the actual pair programming process and the skill involved in it. Its long-term goal is to understand what constitutes that skill and to make it teachable and learnable through behavioral patterns (much like design becomes learnable by design patterns).

In the context of pair programming, knowledge transfer has three different roles: It can be the main purpose of a whole pair programming session, for instance to bring a new team member up to speed quickly. It can be a valuable side-effect of a session by spreading relevant knowledge about requirements, technology, existing code, etc. within a team. It is always an unavoidable ingredient of a pair programming session, because the partners constantly explain their thoughts and ideas to each other. We ask the following research question: *What mechanisms underlie knowledge transfer during pair programming and which of these work well or not so well?*

*This paper is a short summary of the full article with the same title [ZP14].

Our analysis is based solely on recordings of pair programming sessions of professional software developers working on real tasks in their original office environment. It uses the Grounded Theory Methodology (GTM, [SC90]) and aims at a Grounded Theory (GT) of knowledge transfer in pair programming such that its main concepts directly inform useful behavioral patterns for practitioners.

So far, we have identified four elements of knowledge transfer skill in pair programming [ZP14]. When putting them together, they allow to formulate a rough sketch of the problem solving process for knowledge transfer challenges.

- Whenever both partners perceive a different (perhaps vague) knowledge need at the same time, they need to determine an order for satisfying those needs and must not both pursue (in our terminology, *propel*) the needs at once.
- The need-pursuing partner, the *Propellor*, must make sure to recognize if the knowledge-to-be-transferred, the *Target Content*, contains multiple elements that together are too complicated to be explained at once and so need to be split up into multiple *Topics* for separate clarification.
- From the viewpoint of the developer-in-need, a *Direct Question* can be a simple way to trigger an explanation. Often though, this is insufficient for successfully communicated the *Topic* at hand to the knowledgeable partner, so the speaker needs to devise a sequence of more sophisticated *Explanation Triggers* to lead first herself and then the partner towards a sufficiently precise understanding. We call this sequence a *Clarification Cascade*.
- The pair must not lose sight of a *Topic* until it is resolved or they find a good reason to give up. Losing sight can happen easily because additional knowledge transfer *Episodes* (aiming at different *Topics*) often intervene.

Although the above process sketch is incomplete, even these few behaviors are sufficiently difficult to make some pairs much more efficient in their knowledge transfer than others. We consider such pairs to possess the superior pair programming skill.

Acknowledgments. This work was supported by a DFG grant.

References

[HDAS09] J.E. Hannay, T. Dybå, E. Arisholm, and D.I.K. Sjøberg. The effectiveness of pair programming: A meta-analysis. *Information and Software Technology*, 51(7):1110–1122, 2009.

[SC90] Anselm L. Strauss and Juliet M. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. SAGE, 1990.

[ZP14] Franz Zieris and Lutz Prechelt. On Knowledge Transfer Skill in Pair Programming. In *ESEM'14 Proc. 8th ACM/IEEE Intl. Symposium on Empirical Software Engineering and Measurement*. ACM, 2014.