Observations on Knowledge Transfer of Professional Software Developers during Pair Programming

Franz Zieris[1], Lutz Prechelt[2]

This is an extended abstract of the paper with the same title [ZP16] which was presented at the 38th International Conference on Software Engineering (2016).

Keywords: pair programming; knowledge transfer; grounded theory

1 Background, Context, and Research Method

Pair programming (PP) in industrial settings is usually understood as either a productive practice in the XP-sense [Be99] or as a mentoring technique to bring new team members up to speed. It appears to be common sense that pair programmers are either good enough to perform a productive session (without any relevant knowledge transfer) or their skill levels are too far apart to be productive, so they retreat to a knowledge transfer session. But such a strict dichotomy does not capture the full reality of industrial software development: Knowledge transfer is a key ingredient of any pair programming session (*especially* between two experts), and even a knowledge-wise inferior developer can have big positive influence on the PP session's progression.

In contrast to just looking for an effect of pair programming on developers' knowledge levels, we want to understand the actual process of how knowledge transfer happens during pair programming. For this, we record in-vivo PP sessions of industrial software developers and analyze this material (consisting of screen capture, webcam, and audio) on an utterance-level granularity employing Grounded Theory Methodology [SC90].

2 Results

In a previous article [ZP14], we reported our initial findings on what we called "Knowledge Transfer Skill" in pair programming: Good pairs manage to (1) not pursue multiple knowledge needs at once, (2) recognize complex Topics and split them into separate Topics,

[1] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland zieris@inf.fu-berlin.de
[2] Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin, Deutschland prechelt@inf.fu-berlin.de

(3) recognize complicated Topics and deal with them in stages, and (4) not lose sight of Topics. In our recent study [ZP16], we analyzed well over 400 knowledge transfer episodes from 13 PP sessions and were able to conceptualize more observations that are relevant for understanding knowledge transfer in pair programming:

- There is usually no pair member who is more knowledgeable in *all* relevant areas. Even in sessions dedicated to introducing a new team member, the senior developer learned something along the way.
- Occasionally, it is the knowledge-wise inferior developer who has the bigger positive impact on the session's progress, either through solving a problem (insight) or by avoiding a mistake (diligence).
- Existing knowledge is not just *"pulled"* from the more knowledgeable developer through asking questions; developers will also start explanations (*"push"*), even if her partner did not specifically ask for it. In PP, knowledge gaps can be dealt with even if the developer in need is not aware of them.
- When new understanding is being *"produced"* by one pair member alone (e.g., by reading source code), sinking into silence is a bad choice compared to uttering intermediate insights as they would allow the partner to both help and learn.
- When both developers cooperatively work on new understanding, one developer occasionally pulls ahead. The pair then needs to get close together again to fully use their complementary knowledge. This *resynchronization* requires some additional effort, which will pay off later.
- But even equally quick grasp of both developers does not guarantee common understanding: Producing new understanding as a pair always requires some additional synchronization. Otherwise the pair risks ending up on *parallel tracks* and taking avoidable detours in their process.

References

[Be99] Beck, Kent: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 1999.

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

[ZP14] Zieris, Franz; Prechelt, Lutz: On Knowledge Transfer Skill in Pair Programming. In: Proc. 8th ACM/IEEE Int'l. Symposium on Empirical Software Engineering and Measurement. ESEM '14, ACM, New York, NY, USA, pp. 11:1–11:10, 2014.

[ZP16] Zieris, Franz; Prechelt, Lutz: Observations on Knowledge Transfer of Professional Software Developers During Pair Programming. In: Proc. 38th Int'l. Conf. on Software Engineering Companion. ICSE '16, ACM, New York, NY, USA, pp. 242–250, 2016.