

Pair Programming: When and Why it Works

Jan Chong¹, Robert Plummer², Larry Leifer³, Scott R. Klemmer², Ozgur Eris³, and George Toye³

1 Stanford University, Department of Management Science and Engineering, Terman Engineering Center,
3rd Floor, Stanford CA 94305
jchong@cs.stanford.edu

2 Stanford University, Department of Computer Science, 353 Serra Mall, Stanford CA 94305
{plummer, srk}@cs.stanford.edu

3 Stanford University, Department of Mechanical Engineering, Building 530, 440 Escondido Mall,
Stanford CA 94305
ozgur@stanford.edu; {toye, leifer}@cdr.stanford.edu

Pair programming is a software development technique where two programmers work together at a single PC. Over the past few years, pair programming has emerged as a promising method for creating higher-quality software in a time-efficient manner. It is a central aspect of many agile software development methods. While prior research has demonstrated the effectiveness of pair programming, there is still limited understanding as to *when* and *why* it is effective. Our research into the underlying reasons for success – and limitations of – pair programming employs a two-phase method. In the first phase, we are conducting ethnographic studies of software development teams in industry that currently employ pair programming. We will use the results of this phase of the research to drive the second phase of the research: a laboratory study of pair programming with professional developers as participants.

1 Introduction

This study is an investigation into the socio-cognitive factors of pair programming, a method of where two people work together shoulder-to-shoulder at a single computer. Studies of pair programming in university programming classes have shown that pair programming yields better design, more compact code, and fewer defects for roughly equivalent person-hours [1-5]. Studies have also noted that pair programmers exhibit greater confidence in their code and more enjoyment of the programming process [5-8]. Positive results with pair programming have led to speculation that a collateral benefits of the practice may include improved morale and project knowledge shared efficiently across the development team in a manner that can be expected to improve productivity in subsequent development cycles [9].

While these results are compelling, the adoption of pair programming has faced resistance and skepticism from both managers and programmers. While this may simply be a result of either the novelty of the practice or skepticism of the larger methodological context (Extreme Programming/Agile methods) in which pair programming is often introduced, there is some evidence that pair programming may not necessarily be appropriate for everyone [10]. While prior research has demonstrated the effectiveness of pair programming, there is still limited understanding as to *when* and *why* it is effective. Additionally, the bulk of prior work has studied university students, and it is not clear to what extent these results transfer to professional programmers. Our study focuses on pair programming practices among software development professionals, both in a natural work setting and in the laboratory. Such an understanding would greatly aid managers and programmers in determining when and how to use pair programming to improve the software development process. Our research on pair programming fits into the broader picture of studying collaboration in software development.

