Evaluation of the Effects of Pair Programming on Performance and Social Practices in Distributed Software Development

Muhammad Tauqeer Haider and Imran Ali

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 2 x 20 weeks of full time studies.

Contact Information:
Authors:

Muhammad Tauqeer Haider
mtauqeer.haider@gmail.com

Imran Ali
imran.bth@gmail.com

University advisor:
Dr. Darja Šmite
School of Computing (COM), BTH

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet: www.bth.se/com
Phone: +46 455 38 50 00
Fax: +46 455 38 50 57
ABSTRACT

Context. Agile methods address the challenges of an unpredictable world by relying on “people and their creativity rather than on processes”, accelerate delivery of software and considered as a reaction to plan-based or traditional methods. Distributed software development helps to access a pool of skilled personnel, completion of tasks around the clock and more. Incorporating of agile methods in distributed software development could help to solve some problems of distributed software development such as lack of communication and its dependencies, close collaboration and so on.

Objectives. In this study we investigate the proposed benefits of pair programming, an XP development technique used by agile, and its effects on performance and social practices in distributed software development.

Methods. Systematic literature review and an experiment are utilized to fulfill the objectives of this study. In the systematic review a sub-set of the research articles are selected relevant to the subject of this study from the electronic sources including, ACM Digital Library, IEEE, Xplore, EiVillage (Compendx, Inspec), Science Direct and ISI Web of Science. Experiment is conducted to investigate the pair programming effects on performance and social practices.

Results. Many proposed benefits of pair programming in existing literature are identified and reported in both collocated and distributed settings. Pair programming is reported as an effective software development technique as well as a pedagogical tool. Experimental results showed that pair programming also effects performance in distributed software development, and positively impacts the social practices (human or social factors).

Conclusions. There are many benefits of pair programming reported in collocated settings and less in distributed software development. Pair programming impacts the performance and social practices positively. However, we also conclude that the effective use of pair programming in distributed software development will yield the concrete results as well as the programmers’ pairs should be trained, experienced and well motivated for an effective use of pair programming and to overcome the challenges of distributed software development.

Keywords: pair programming, social practices, systematic review, experiment.
ACKNOWLEDGMENTS

First and foremost, we would like to thank Almighty Allah, on blessing us with the strength and courage to successfully complete this thesis work.

We would like to express our gratitude to our supervisor Dr. Darja Šmite for guidance and support throughout this thesis project.

We are thankful to all the participants in the experiment. We are also thankful to IT Department at BTH for support and providing us the facility of a server to conduct the experiment.

Lastly we are thankful to our families and friends for their support, kindness and motivation throughout our studies.

Tauqeer and Imran
# TABLE OF CONTENTS

EVALUATION OF THE EFFECTS OF PAIR PROGRAMMING ON PERFORMANCE AND SOCIAL PRACTICES IN DISTRIBUTED SOFTWARE DEVELOPMENT---------------------------- 1

ABSTRACT---------------------------------------------------------------------------------------------------- III

ACKNOWLEDGMENTS------------------------------------------------------------------------------------------------ IV

TABLE OF CONTENTS----------------------------------------------------------------------------------------------- V

LIST OF FIGURES------------------------------------------------------------------------------------------------ VI

LIST OF TABLES------------------------------------------------------------------------------------------------ VII

1 INTRODUCTION------------------------------------------------------------------------------------------- 1

1.1 BACKGROUND------------------------------------------------------------------------------------------- 1

1.2 AIMS AND OBJECTIVES----------------------------------------------------------------------------------- 3

1.3 RESEARCH QUESTIONS------------------------------------------------------------------------------------ 3

1.4 THESIS OUTLINE---------------------------------------------------------------------------------------- 3

1.5 TERMINOLOGY------------------------------------------------------------------------------------------- 4

2 RESEARCH DESIGN------------------------------------------------------------------------------------------ 5

2.1 SYSTEMATIC LITERATURE REVIEW---------------------------------------------------------------------- 6

2.1.1 Summary of existing SLRs------------------------------------------------------------------- 6

2.1.2 Review protocol----------------------------------------------------------------------------- 6

2.1.3 Quality assessment criteria--------------------------------------------------------------- 10

2.1.4 Data extraction strategy------------------------------------------------------------------ 10

2.1.5 Review protocol validation------------------------------------------------------------ 11

2.1.6 Synthesis of results----------------------------------------------------------------------- 11

2.2 EXPERIMENT DESIGN----------------------------------------------------------------------------------- 13

2.2.1 Definition-------------------------------------------------------------------------------------- 13

2.2.2 Experiment Planning------------------------------------------------------------------------ 14

2.2.3 Experiment execution----------------------------------------------------------------------- 18

3 RESULTS AND ANALYSIS OF SYSTEMATIC REVIEW------------------------------------------------------------- 20

3.1 CLASSIFICATION OF INCLUDED STUDIES--------------------------------------------------------------- 20

3.2 PROPOSED BENEFITS OF PP------------------------------------------------------------------------- 21

3.2.1 PP benefits in co-located software development----------------------------------------- 21

3.2.2 PP benefits in distributed software development------------------------------------------ 22

3.2.3 Summary of the benefits------------------------------------------------------------------- 24

3.3 PP SUPPORTING TOOLS IN DSD--------------------------------------------------------------------- 26

3.4 ANALYSIS-------------------------------------------------------------------------------------------- 27

3.5 VALIDITY THREATS RELATED TO SLR------------------------------------------------------------------ 29

3.5.1 Conclusion validity------------------------------------------------------------------------ 29

3.5.2 Internal validity--------------------------------------------------------------------------- 29

3.5.3 Construct validity-------------------------------------------------------------------------- 30

3.5.4 External validity--------------------------------------------------------------------------- 30

4 RESULTS AND ANALYSIS OF EXPERIMENT----------------------------------------------------------------- 31

4.1 ANALYSIS OF THE DIRECT MEASUREMENT------------------------------------------------------------- 32

4.1.1 Performance----------------------------------------------------------------------------- 32

4.1.2 Social practices------------------------------------------------------------------------- 36

4.2 ANALYSIS-------------------------------------------------------------------------------------------- 43

4.2.1 Performance----------------------------------------------------------------------------- 43

4.2.2 Social practices------------------------------------------------------------------------- 44

4.3 VALIDITY THREATS RELATED TO EXPERIMENT------------------------------------------------------------ 44

4.3.1 Conclusion validity----------------------------------------------------------------------- 44

4.3.2 Internal validity-------------------------------------------------------------------------- 45

4.3.3 Construct validity------------------------------------------------------------------------- 45
LIST OF FIGURES

Figure 1 Overview of research design ................................................................. 5
Figure 2 Overlap of studies with related SLRs ....................................................... 11
Figure 3 Selected studies by year ........................................................................ 20
Figure 4 Selected studies types ............................................................................ 20
Figure 5 Positive effects of PP in collocated settings ........................................... 28
Figure 6 Negative/No effects of PP in collocated settings ..................................... 28
Figure 7 Effects of PP in distributed settings ....................................................... 29
Figure 8 Box plots for performance of two treatments for task1 ............................ 33
Figure 9 Boxplots for performance of two treatments for task2 ............................ 34
Figure 10 Participants responses percentage for communication ......................... 38
Figure 11 Participants responses percentage for knowledge sharing ..................... 39
Figure 12 Participants responses percentage for satisfaction and confidence .......... 41
Figure 13 Participants responses percentage for enjoyment of work ..................... 42
**LIST OF TABLES**

Table 1 Terminology used in this Thesis ................................................................. 4  
Table 2 Primary studies selection process ............................................................. 9  
Table 3 Kappa statistics and strenght of agreement ......................................... 10  
Table 4 Assignment of subjects to treatments for a randomized design .......... 18  
Table 5 Summary of PP benefits in collocated settings .................................. 24  
Table 6 Summary of PP benefits in distributed settings .................................... 25  
Table 7 Collected data for task1 ........................................................................... 32  
Table 8 Descriptive statistics of task1 ................................................................. 33  
Table 9 Collected data for task2 ........................................................................... 33  
Table 10 Descriptive statistics for task2 ............................................................... 34  
Table 11 The results of t-test on performance of two treatments for task1 ....... 35  
Table 12 The results of t-test on performance of two treatments for task2 ....... 36  
Table 13 Questions for communication ............................................................... 37  
Table 14 Responses for communication .............................................................. 37  
Table 15 Descriptive statistics for communication .............................................. 37  
Table 16 Questions for knowledge sharing/transfer .......................................... 38  
Table 17 Responses for knowledge sharing/transfer ......................................... 38  
Table 18 Descriptive statistics for knowledge sharing/transfer ......................... 39  
Table 19 Questions for satisfaction and confidence about solution ................... 39  
Table 20 Responses for satisfaction and confidence about solution ................. 40  
Table 21 Descriptive statistics for satisfaction and confidence about solution .... 40  
Table 22 Question for enjoyment of work ........................................................... 41  
Table 23 Responses for enjoyment of work ......................................................... 41  
Table 24 Descriptive statistics for enjoyment of work ....................................... 42  
Table 25 Open ended questions .......................................................................... 42
1 INTRODUCTION

1.1 Background

Global software development is a norm in software industry and currently more software projects are running in global distributed environment [1]. There are various reasons that cause companies to become part of global environment such as, a pool of skilled personnel, completion of tasks around the clock and low labor rates in developing countries [2]. Many companies started global environment to earn benefits from cheaper, faster and better development of software systems, products and services but empirical studies reveal that this is not an easy task to achieve these benefits [3]. There are many challenges that global organizations face in global (or distributed) environment like cultural difference, lack of communication and its dependencies, and time zone difference [4]. Agile methods are used in global software projects to overcome these challenges. At first glance, agile methods and global software development might seem incompatible because the main emphasis of the agile development methods is communication and close collaboration among the teams while these are the main challenges for global organizations [5].

Incorporating of agile methods in distributed software development (DSD) has been reported as successful experiences which indicate that agile methods could help to solve some problems of distributed software development [5]. However, there are many challenges associated with this combination such as communication, personnel, culture, different time zones, trust and knowledge management, and to work it effectively more effort is needed [6].

Agile methods can be seen as a reaction to plan-based or traditional methods, which emphasize “a rationalized, engineering-based approach” and address the challenges of an unpredictable world by relying on “people and their creativity rather than on processes” [7]. Agile Software Development fulfills the principles behind agile manifesto\footnote{Visit \url{http://agilemanifesto.org/principles.html} to read about the agile manifesto principles.} and generally characterized by following attributes [8, 9]:

- **Incremental** – Small software releases with rapid cycles. The software project is divided into small pieces and feedback on the small pieces allows to iterate or to refine each release with new functionalities.

- **Cooperative** – Customer and developers working constantly together with close communication.

- **Straightforward** – A well documented method that is easy to learn and to modify.

- **Adaptive** – An agile process allows adding new activities and modifying the existing activities as required (able to make last moment changes).

The foremost agile methods are Extreme Programming (XP) [10-12] and SCRUM [13, 14] [21]. Some other agile methods are Feature Driven Development (FDD) [15], Adaptive Software Development (ASD) [16], Agile Modeling (AM) [17], Crystal Clear [18], Lean software development [19] and Dynamic Systems Development (DSDM) [20].
Currently Extreme Programming (XP) [10-12] is most commonly used agile method [21]. XP is a set of principles, a collection of values and focuses on best practices for software development. XP consists of twelve practices [11, 12]: the planning game, small releases, metaphor, simple design, testing, refactoring, pair programming, collective ownership, continuous integration, 40-hweek, on-site customers, and coding standards. The revised “XP2” consists of the following “primary practices”: sit together, whole team, informative workspace, energized work, pair programming, stories, weekly cycle, quarterly cycle, slack, 10-minute build, continuous integration, test-first programming, and incremental design. The most and commonly used XP practices are: pair programming, planning game and standup meetings [6],[21] retrospectives and continuous integrations [6].

Pair Programming (PP) - all production code is written by two people at one screen/keyboard/mouse [11]. Pair programming is a collaborative approach that makes working in pairs rather than working in individual for code development [22]. One programmer writes a software artifact (e.g. program code or UML diagrams) and other programmer continuously assures quality of the software artifact by watching, asking questions, looking for some alternative approaches, helps to avoid defects etc. The two programmers switch their roles after some time: creator, is also called the Driver [23, 24], becomes quality assurer, is also called the Navigator [23, 24], and vice versa [25].

Despite pair programming growing popularity, we still know very little about pair programming in sense of reliable experimental data and further experiments are needed to investigate effectiveness of pair programming [26]. A systematic review by Hannay et al [27] in terms of Meta analysis on the effectiveness of pair programming is conducted and concluded that there should be great attention given to moderating factors on the effects of pair programming. Pair programming leads to rethinking about the concept of development teams and about how individual programmers can best contribute to the project. Now pair programming has seen increasing interest and adoption, it is useful to consider what has been learned about its more specific effects [22]. Pair programming seems to be dependent upon collocation, but could be extended to support distributed teams [28]. Pair programming in distributed software development gives rise to important question(s) such as “How effective is pair programming?” is related to the effectiveness of pair programming when pairs are not physically next to each other and programmers are geographically distributed [23].

In ideal software development teams, the team members have rich interactions and communication, both formal and informal; share a common culture which facilitates good coordination, rapid access to other team members with similar technical skills and expertise, and familiar with the tools and technologies appropriate for the project [29]. Distributed development adds new demands to the software development process and is a potential threat to each of these ideal properties [29]. In the study by Hinds & Mortensen [30] described the relationships between geographic distribution, shared identity, shared context, spontaneous communication and conflicts among distributed teams.

This study aims to evaluate the pair programming to know its effects on performance – in terms of time to complete the task(s); and social practices\(^2\) in distributed settings with respect to solo programming.

\(^2\) Social practices include Communication, Knowledge sharing/transfer, Satisfaction and Confidence about the solution and enjoyment of work
1.2 **Aims and objectives**

The aim of this study is to explore the benefits and to investigate the effects of pair programming on performance and social practices in distributed development.

This aim was realized by achieving following objectives:

- Exploring the benefits of pair programming in both collocated and distributed development as well as to present the supporting tools for distributed pair programming reported in the related published literature.

- Evaluation of the possible impacts of pair programming on social practices of each individual in distributed pairs.

- Investigating the pair programming effects on performance in distributed pairs.

1.3 **Research questions**

As previously stated the first objective of this thesis is to explore the benefits of PP and PP supporting tools in distributed settings. The second and third objectives relate our empirical findings of PP effects on performance and social practices in distributed software development with the reported findings in the relevant literature.

Following research questions (RQs) were derived based on the objectives:

- **RQ1**: What are the benefits of pair programming claimed in the related published literature in both collocated and distributed development?

  - **RQ1.1**: How can distributed pair programming be enabled through computer mediation?

- **RQ2**: What are the possible impacts of pair programming practice on social practices of each individual in distributed development?

- **RQ3**: What are the effects of pair programming on performance in distributed development?

1.4 **Thesis outline**

The rest of the thesis report is organized as follows:

*Chapter 1, Introduction*, gives an introduction to the subject of this study.

*Chapter 2, Research Design*, describes the research design of this study describing the systematic literature review and experiment in detail.

*Chapter 3, Results and Analysis of Systematic Review*, and *Chapter 4, Results and Analysis of Experiment*, present the results and analysis of systematic review and experiment respectively.
Chapter 5, Discussion, presents a discussion on the results of systematic review and experiment.

Chapter 6, Conclusions, contains a brief summary of the thesis.

Chapter 7, Reference, provides a list of the references used in this study.

Appendix in Chapter 8 provides a list of the studies selected in systematic review as well as the guidelines for pair programming, participant’s data form, data collection forms and survey questionnaire that we used in the experiment.

1.5 Terminology

Table 1 Terminology used in this Thesis

<table>
<thead>
<tr>
<th>Terms/Abbreviations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSD</td>
<td>Distributed Software Development</td>
</tr>
<tr>
<td>PP</td>
<td>Pair programming</td>
</tr>
<tr>
<td>DPP</td>
<td>Distributed pair programming</td>
</tr>
<tr>
<td>VPP</td>
<td>Virtual pair programming</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme programming / eXtreme programming</td>
</tr>
<tr>
<td>SP</td>
<td>Solo programming</td>
</tr>
<tr>
<td>Authors</td>
<td>Students responsible to write this report (Tauqeer and Imran)</td>
</tr>
</tbody>
</table>
2 RESEARCH DESIGN

A mixed approach, qualitative and quantitative, is used in this study. The systematic literature review and experiment were two main activities to answer the research questions. This chapter illustrates the research design for this study.

Figure 1 Overview of research design


2.1 Systematic Literature Review

A Systematic Literature Review (SLR), is also called Systematic Review (SR), is defined as a process of identifying, assessing and interpreting all available research evidence with the purpose to provide the answers for specific research questions [31]. The guidelines provided in [31] were followed to perform the SLR.

The purpose of the SLR for this study was to identify and analyze all the published research evidence to fulfill the aim and objectives of the study by answering the RQs as described in Section 1.3 above.

Initially, we found the following existing SLRs [6], and [21, 22] that were partly relevant to this study as summarized in Section 2.1.1.

2.1.1 Summary of existing SLRs

In 2010, Jalali & Wohlin [6] conducted a systematic map on agile practices in Global Software Engineering (GSE) from 1999 to 2009. The systematic map presented the current research literature on the use of agile practices in GSE. The systematic review highlighted under which circumstances the agile methods had been applied successfully i.e. agile practices and distribution types, for example, pair programming and distributed teams. The systematic review provided the guidelines for literature review in GSE context besides it covered more than one agile practices and distribution types.

In 2008, Dybå & Dingsøyr [21] carried out a SLR of Agile Software Development empirical studies to find the empirical evidence for benefits, limitations, and strengths of agile methods. They found low strength of evidence supporting PP in agile methods. However, in 2007, Dyba et al. [22] conducted a SLR focusing on effectiveness of PP. The study [22] investigated the empirical evidence and supported the claims that PP is more advantageous than solo programming. The PP’s aspects investigated were related to effectiveness focusing “duration” (time spent to produce the system), “effort” (person-hours spent), and “quality of the final product”. The SLR [22] as an intermediate analysis was extended by Hannay et al [27] as a full-scale analysis in 2009, which summarized pair programming experiments published up to and until 2006. In addition, the studies published up to August 2007 were also taken into account [27].

2.1.2 Review protocol

The purpose of this review method was to specify the plan which the systematic review followed to identify, appraise and collect the evidence on the benefits and empirical evaluation of PP practice in both collocated and distributed software development.

2.1.2.1 Objectives

The objectives of the review were to:
• Find and summarize all existing evidence on the benefits and evaluation of PP practice in both collocated and distributed software projects.

• Select a sub-set of studies for review to answer the RQs described in Section 1.3 above.

2.1.2.2 Inclusion and exclusion criteria

To select the primary studies for the review we considered the following inclusion criteria:

Studies of pair programming as following:

• It was an empirical study of PP (e.g. PP is evaluated, investigated or studied for software quality, programmers’ productivity, development cost, development time etc.).

• PP benefits, advantages or impacts were presented or described quantitatively, qualitatively or both in the study, for example, PP impacts on social interaction, social practices or behavioral factors were described.

• PP was investigated using its variations such as side-by-side programming, collaboration programming, virtual pair programming or distributed pair programming (DPP) in which PP benefits, evaluation, investigation or challenges were presented.

• PP benefits, advantages or challenges were described with other agile/XP development practices, or as a combined study with other agile practices.

• Studies that only described tools that could support PP in distributed settings.

• PP was studied in industry, academia or both using students, professionals or both in which PP benefits, challenges or validation were presented.

• Study of pair programming in which a comparison was made between pairs and individuals, possibly in a team context.

Exclusion criteria for studies:

• Studies were excluded if the focus, or main focus, was not PP or variations of PP.

• Studies did not present qualitative or quantitative data.

• Studies presented only the opinion(s) of the author(s), “lessons learned” studies (paper without a research question and research design) were not included.

• Studies presenting claims by the author(s) with no supporting evidence were not included.

2.1.2.3 Search strategy for identification of studies
Following electronic sources of relevance for software engineering subjects were searched:

**Data sources:**
- ACM Digital Library
- IEEE Xplore
- EiVillage2 (Compendex, Inspec)
- Science Direct (Elsevier)
- ISI Web of Science

Above listed electronic databases were also searched in related SLRs [6], [21, 22] and [27].

In [27] authors refer to [32] and state that:

"We did not perform separate searches in the Software Engineering (SE) specific databases Kluwer Online, Science Direct, Springer Link, and Wiley Inter Science Journal Finder, because previous experience with systematic search strategies has shown that articles retrieved from these databases are also returned by either ISI Web of Science or Compendex."

The existing SLRs spanned over the years from 1998 [21], on the bases of first study found, to mid of 2009 [6] including [22] and [27] (See section 2.1.1 above). We searched the electronic databases from 1998 till 2010 including.

**Search strategy:**

Following are the keywords extracted from the RQs:
- Pair programming
- Experiments
- Benefits, social practices

Adding possible synonyms or relevant terms to above keywords:
- Evaluation, measurement, assessment, investigation, validation
- Evolution, efficiency, impacts, performance, productivity, social, factors, behavioral

**Search string:**

("pair programming") AND (experiment* OR evaluation OR measurement OR assessment OR investigation OR validation) AND (benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR "social practice*")

The electronic databases were searched for published literature using the above search string. Results are shown in Table 2 that also describes the steps for primary studies selection process. Appendix 8.11 shows the search string searched in each electronic database.
2.1.2.4 Method of the review

The research strategy identified all relevant articles and these were reviewed by two reviewers (authors of this study). Only studies written in English language were included. If it was unclear from the title, abstract, and introduction whether a study conforms to the screening criteria or not, it was included for a detailed study to make further decision.

Table 2 Primary studies selection process

<table>
<thead>
<tr>
<th>Electronic Databases</th>
<th>ACM</th>
<th>IEEE</th>
<th>EiVillage (Compendex, Inspec)</th>
<th>Science Direct (Elsvier)</th>
<th>ISI Web of Science</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>522</td>
<td>621</td>
<td>202</td>
<td>110</td>
<td>49</td>
<td>1504</td>
</tr>
<tr>
<td>1. Applying possible filters such as remove duplicates at database level:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>522</td>
<td>617</td>
<td>171</td>
<td>110</td>
<td>49</td>
<td>1469</td>
</tr>
<tr>
<td>2. Selected studies based on Title (includes repetitions and duplicates):</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>98</td>
<td>123</td>
<td>23</td>
<td>26</td>
<td>371</td>
</tr>
<tr>
<td>3. Repetition of studies removed based on Title among the studies at step 2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>98</td>
<td>92</td>
<td>33</td>
<td>18</td>
<td>14</td>
<td>255</td>
</tr>
<tr>
<td>4. Relevant studies selected based on Titles, Abstract and Key words from step 3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>45</td>
<td>20</td>
<td>7</td>
<td>11</td>
<td>125</td>
</tr>
<tr>
<td>5. Final selection (based on detailed review/screening):</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>22</td>
<td>15</td>
<td>2</td>
<td>6</td>
<td>68</td>
</tr>
</tbody>
</table>

Repetition of studies removed based on Title among the studies at step 2 as follows:

If we found, for example, [S2], [S3] – 2 studies retrieved from ACM as well as from EiVillage(Compendex,Inspec) then we removed the duplicated studies from EiVillage(Compendex,Inspec) because we searched manually among ‘Titles’ and removed the repetitions initially based on ACM. One possible effect is that number of the studies retrieved from each electronic database will change, for example, by removing duplicated studies [S2],[S3] from EiVillage(Compendex,Inspec) will increase the number of studies in ACM and vice versa.

2.1.2.5 Kappa

The statistics kappa is a measure used to examine the agreement level between two people (researchers/raters/observers) [33]. Researchers rate each of a sample of subjects on a nominal scale [33]. Kappa is useful when all disagreements may be considered equally serious, and weighted kappa is useful when the relative seriousness of the different kinds of disagreement can be specified [33]. Weighted kappa also helps to measure the inter observer bias that is found more in categorical data [33] [34].

Kappa values ranges from 0.0 to 1.0, where large number means better reliability. Lower values or values near to zero suggest that agreement is attributable to chance alone [34]. In order to maintain consistency following labels are assigned to the corresponding ranges of kappa [34] as shown in Table 3.
In this study, the Kappa statistics was performed to determine the reliability of selection between authors (raters) on the subjects (empirical studies of PP) where disagreement was found. The kappa, reliability, for the raters was found 0.677 i.e. strength of agreement is Substantial (See in Appendix: Kappa Analysis³).

Table 3 Kappa statistics and strength of agreement

<table>
<thead>
<tr>
<th>Kappa Statistic</th>
<th>Strength of Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0.00</td>
<td>Poor</td>
</tr>
<tr>
<td>0.00-0.20</td>
<td>Slight</td>
</tr>
<tr>
<td>0.21-0.40</td>
<td>Fair</td>
</tr>
<tr>
<td>0.41-0.60</td>
<td>Moderate</td>
</tr>
<tr>
<td>0.61-0.80</td>
<td>Substantial</td>
</tr>
<tr>
<td>0.81-1.00</td>
<td>Almost Perfect</td>
</tr>
</tbody>
</table>

2.1.3 Quality assessment criteria

Mainly the inclusion and exclusion criteria (Section 2.1.2.2 above) worked as the quality assessment criteria. In addition, the primary study that had empirical evidence i.e. empirical data was available either in descriptive, qualitative or quantitative form.

However, we did not assess the quality of the included studies in terms of, for example, research methodology, subjects and tasks selection, validity threats or study was successful or not.

2.1.4 Data extraction strategy

We used the Microsoft Excel (MS) spread sheet to extract the data from the primary studies. We considered following information of a study:

2.1.4.1 General information

- Title of the study
- Name of the Author(s)
- Electronic search database used to retrieve the study
- Publication year of the study

2.1.4.2 Specific information from the study relevant to RQs

Study environment:

- Industrial
- Academia
- Mix (Industrial and Academia)

Study type:

- Experiment

³ Kappa is performed using SPSS
• Case study
• Survey
• Interviews

Study participants:
• Students
• Professionals
• Number of participants

The results of the research article to answer the RQs:
• Proposed benefits in collocated and distributed settings
• Results relevant to social practices

2.1.5 Review protocol validation

The review protocol’s internal validation was performed by the thesis project supervisor and for the external validation; we discussed our SLR process with a library staff at BTH Library. In addition, the search process is also evaluated partly against the results from previous systematic reviews [6], [22] and [27] as described in Section 2.1.1 above. The overlap of the studies among own SLR and the related SLRs is presented in Figure 2 and Appendix 8.10 shows corresponding studies of our SLR and related SLRs. The final set of selected studies is listed in Appendix 8.1 below.

Figure 2 Overlap of studies with related SLRs

It should be noted that overlapped studies were also retrieved by our search string and assessed based on inclusion/exclusion criteria as described in Section 2.1.2.2.

2.1.6 Synthesis of results

Collecting and summarizing the results of the primary studies is called data synthesis [31][35]. Synthesis can be descriptive (non-quantitative or narrative) and it is sometimes possible to complement with a quantitative summary [31]. In descriptive synthesis extracted information about the studies (i.e. intervention, population, context, sample sizes, etc) should be tabulated in a manner to answer the review [31]. Homogeneous or heterogeneous types of results should be identified and extracted into structured tables [31]. Line of argument synthesis, an approach to qualitative synthesis, is used when researchers are concerned about what they can infer about a topic as a whole from a set of selective studies that look at a part of the issue [31].
The first stage of the synthesis was to identify the proposed or studied benefits of PP in both collocated and distributed settings respectively. The original terms (proposed benefits of PP) were extracted into a data extraction form and then placed into structured tables. Finally, primary studies were grouped with similar findings or results. In Chapter 3 “Results and Analysis of Systematic Review” results of the systematic literature review are presented and described in detail.


2.2 Experiment Design

The main purpose of an experiment is to draw meaningful conclusions regarding the problem at hand [36]. Statistical analysis methods are applied to collect and manipulate the data and to interpret the results [36]. To achieve real benefits and advantages of an experiment it is necessary that the experiment is carefully planned and designed [36]. The application of a particular statistical technique depends on experiment design chosen and the measurements scales used [36].

2.2.1 Definition

The object of study is performance. The subjects are the students participated in the experiment. The purpose of the experiment is to evaluate the effects of pair programming on performance and on social practices in distributed environment with respect to the solo programming. The goal definition framework [36] is described as:

Object of study. The objects studied are performance and social practices.

Purpose. The purpose is to evaluate the impacts of PP practice on performance and social practices with respect to SP.

Quality focus. The quality focus is performance in PP.

Perspective. The perspective is from the researcher’s point of view.

Context. The experiment is run with the help of CS/SE students as subjects involved in the programming tasks in Java.

Social practices are briefly described below:

Communication – plays vital role for success in distributed software projects when there is an absence of face-to-face meetings that are recommended for agile methods, for example, in pair programming [37]. As an alternative of face-to-face meetings information and communications technology (ICT) are used, for example, instant messaging, emails or video conferences to discuss the issues, problems or solutions in distributed software development [38].

Knowledge sharing/transfer – Conventional software development projects (non-agile projects) have limited interaction among programmers and do not promote knowledge sharing on a day-to-day basis [39]. On the other hand agile practices force more interaction and discussion among the team members to enhance the knowledge sharing [39].

A claimed benefit of pair programming is that it fosters knowledge leveraging between the two programmers, particularly tacit knowledge [40]. The building of tacit knowledge through the practice is affected by a huge amount of factors difficult to capture and measure, such as personal attitude, capability, experience. Pair programming, therefore, enables knowledge sharing and cross training between programmers and is supposed to be a practice suitable for this purpose [40].

---

4 Performance - time to complete a task. It was measured in a unit of time. In this study, less time to complete a task represents that performance was high.
Enjoyment of work – Group work, in general, is more enjoyable compared to working alone. Group membership fulfills social and emotional human needs and, therefore, pair programming promotes the formation of good team spirit [41]. This is due to the sheer fun of social interaction and to positive feelings that rise as a result of helping and being helped [42].

Satisfaction and Confidence about the solution – Compared to individuals working alone, group members tend to have higher goal commitment, more positive attitude toward goal attainment, and report higher satisfaction with their performance [43]. Confidence (feeling of trust or sure or certain of a fact or issue or something⁵) about the solution presents the strength of the participant’s belief about the quality of his/her programming solution [43].

2.2.2 Experiment Planning

2.2.2.1 Experiment context

The context is the environment in which experiment is run [36]. The context describes which personnel is involved in the experiment (subjects) and which software artifacts (objects) are used in the experiment [36]. The experiment context is characterized in terms of the number of subjects and objects involved in the study [36].

We conducted experiment in a university environment in distributed settings at Blekinge Institute of Technology (BTH). The experiment was conducted with CS/SE students from Department of Computing (COM) at BTH as subjects, which had experience of software development in Java programming language. The subjects were the students and object of the study was performance in PP and SP respectively. The experiment context was characterized as multi test within object study [36].

2.2.2.1.1 Distributed environment

The distributed environment for the experiment was achieved by considering the geographically distributed spaces and distributed software development tools were used.

The distributed environment was achieved as:

- Subjects were dispersed in geographically distributed spaces such as separate silence rooms within the university space.

- The use of a tool that supports PP in distributed settings. PP supporting tools in distributed settings are briefly described in Section 3.3 below. We used Xpairtise⁶ – a distributed pair programming plug-in for Eclipse. The plug-in supports instant pair programming and offers shared editing, project synchronization, shared program and test execution, users’ management, built-in chat communication and a shared whiteboard.

Following files are required to work XPairtise⁷ in Eclipse⁸:

⁵ http://www.oed.com/view/Entry/38806?rskey=qYAEfw&result=1#
⁶ See details at http://xpairtise.sourceforge.net/
⁷ These can be download at http://en.sourceforge.jp/projects/sfnet_xpairtise/
⁸ http://www.eclipse.org/
• de.fuh.xpairtise_1.1.0.ZIP – the Xpairtise plug-in to be placed in Eclipse plug-in directory.

• de.fuh.xpairtise.subclipse_1.1.0.ZIP – optional xpairtise plug-in that is used for version-controlled projects and to be placed also in Eclipse plug-in directory.

• de.fuh.xpairtise_server_1.1.0.ZIP – is extracted into a new and empty directory. The server uses this directory to store its working data (user-, and session data base, project data) during operation.

It is recommended that JDK 1.5.0 and JRE or higher should be installed and configured with Eclipse.

2.2.2.1.2 Experimental Tasks

Following tasks were selected for development:

• Program 1. Write a program to estimate the mean and standard deviation of a sample of n real numbers [26] [44].

• Program 2. Write a program to count the logical lines in a program, omitting comments and blank lines [26] [44].

• Program 3. Write a program to count the total program LOC, the total LOC in each object the program contains, and the number of methods in each object [26] [44].

• Program 4. Write a program to calculate linear regression estimating parameters for a set of n programs where the historical LOC and new and changed LOC are available [26] [44].

The tasks selected in [26] and [44] were taken from Humphrey [45], programming exercises for Personal Software Process (PSP) of the level 0 and 0.1. Both experiments [26] and [44] were conducted in colocated environment with CS students as participants in a university setting. Size of each task was 150 to 400 lines of code (LOC) and all tasks were handled in pair and solo programming.

In [26], the aim of the experiment was to compare the pair programming along with other extreme programming practices with the PSP approach with respect to the development time and the number of defects. In [44], the purpose of the experiment was to compare pair programming with solo programming with respect to defect density and productivity.

We did not validate or replicate the studies [26] and [44] because we conducted the experiment in distributed environment.

2.2.2.2 Hypothesis Formulation
**Definition:** We assume from the definition presented earlier that pair programming increases performance with respect to solo programming. In addition, pair programming has positive effects on social practices.

In our particular case, SP was compared with distributed PP while subjects in PP were dispersed in geographically distributed spaces and programmers in SP were positioned in their assigned locations.

This informal statement of hypothesis can be stated formally as following:

*Null hypothesis, $H_0$: Performance remains the same using distributed PP or SP.*

*Alternative hypothesis, $H_1$: Performance increases in distributed PP versus SP.*

*Alternative hypothesis, $H_2$: Performance decreases in distributed PP versus SP.*

$H_0$: Performance (PP) = Performance (SP)

$H_1$: Performance (PP) > Performance (SP)

$H_2$: Performance (PP) < Performance (SP)

The hypotheses for the social practices are as follows:

**Hypothesis Communication:** programmers do effective communication while performing pair programming.

**Hypothesis Knowledge:** pair programming enhances knowledge sharing/transfer among programmers.

**Hypothesis Enjoyment:** programmers enjoy work more while performing pair programming.

**Hypothesis Satisfaction:** pair programming increases satisfaction and confidence about the solution among programmers.

**Measure(s) needed:** Performance – time to complete the tasks in minutes. Likert scale was used for social practices, for example, 1 for ‘strongly agree’ and 5 for ‘strongly disagree’.

### 2.2.2.3 Dependent and independent variables selection

Those variables that are studied to see the effect of the changes in the independent variables are called dependent variables [36]. All variables in a process that are manipulated and controlled are called independent variables [36]. The dependent variable of this experiment was performance and the independent variables were the participants (students) and the tasks.

The independent variables those are controlled and manipulated to see the effect on dependent variable are called factors [36]. All other independent variables are controlled at a fixed level during the experiment. The factor(s) for this experiment were individuals and tasks. A treatment is a one particular value of a factor [36]. There were two treatments of the factor(s): pair programming and solo programming. An experiment consists of a set of tests (trials) where each test is a combination of treatment, subject and object.
2.2.2.4 Selection of subjects

The selection of subjects is closely connected to the generalization of the results from the experiment [36]. As the subjects were the students from CS/SE programs of COM at BTH who had experience of software development using Java programming language. The selected subjects were sample from population for the study. The selections of subjects were organized through convenience sampling [36] – the nearest and most convenient persons were selected as subjects.

2.2.2.5 Design of the experiment

The problem definition, independent and dependent variables has been stated. Furthermore, the measurement scale for the dependent variable has selected that is time - in minutes for performance and Likert scale for social practices. Thus, we are now able to design the experiment.

Randomization, blocking and balancing are the general design principles [36]. We did not use randomization for selection of the subjects (students). However, we used randomization to form teams for both PP and SP. In addition, subjects were allocated randomly to PP and SP teams after completing a task or one module. The purpose was to get the concrete results about social practices.

Balancing is considered to have a balanced data set because it simplifies and strengthens the statistical analysis data [36]. For this experiment balancing was achieved by assigning the equal number of subjects to each treatment.

Blocking is used to systematically eliminate the undesired effect in the comparison among the treatments when we have a factor that probably has an effect on the response, but we are not interested in the effect [36]. It was possible that some subjects (students) were familiar with the agile development methods or had industrial experience so they might affect the results in both PP and SP. To minimize this problem we provided the guidelines, instructions and training to the participants’ specific to the experiment.

2.2.2.6 Standard design type

There are four standard design types which are frequently utilized [36]:

- One factor with two treatments.
- One factor with more than two treatments.
- Two factors with two treatments.
- More than two factors each with two treatments.

The design type suitable for this experiment is ‘One factor with two treatments’. The factor is performance while the treatments are PP and SP. Following trials (tests) are executed:

Trial 1 (Task (1), treatment (PP, SP))
.
.
.
Trial n (Task (n), treatment (PP, SP))
Where ‘n’ represents the maximum number of tasks and trials.

We measured dependent variable (performance) in unit of time - minutes. We used a parametric test [36] particularly a t-test – assesses whether the means of two groups are statistically different from each other, for hypothesis testing.

The sample size of participants was not high (N=9). The completely randomized design [36] was used to allocate the participants to each treatment as one is presented in Table 4. Subjects participating in treatment 1, i.e. PP, worked in distributed pairs, for example, subjects 1 and 3 formed a distributed pair.

**Table 4** Assignment of subjects to treatments for a randomized design

<table>
<thead>
<tr>
<th>Subjects</th>
<th>Treatment 1 (PP)</th>
<th>Treatment 2 (SP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**2.2.2.7 Instrumentation**

There are three types of instruments for an experiment namely objects, guidelines and measurement [36]. All the required and necessary instruments were used by making sure that they were aligned with experiment design and data collection method [36]. Experiment instrumentations provided to participants for this experiment were: specifications for programming tasks, guidelines for PP, instructions about the use of the tools, data collection forms, and survey questionnaire (see Appendix 8).

**2.2.3 Experiment execution**

**2.2.3.1 Preparation**

First of all, the remote server was configured and required software installed then it was used in the training and in the experiment. The participants’ username and password were created on the server.

Participants were well prepared, brief and trained before the actual experiment took place. They were trained individually. The purpose was not to expose the participants to each other before the actual experiment, to minimize the influence on the results and to avoid the competition environment. The training was related to the data entry, installation and configuration of required software on client machines in the experiment. In the training, a small program, simple calculator, was developed to make sure that participant has learned the tool, specifically, XPairtise plug-in for Eclipse.
The guidelines related to the successful execution of the experiment were provided to the participants in order to perform the distributed PP and SP effectively while participants were dispersed in geographically dispersed spaces. Data collection forms were also provided to the participants to collect the experiment data for hypothesis testing, i.e. time to complete the tasks and filled survey questionnaire (see Appendix 8). The experimental data was collected manually.

2.2.3.2 Execution

The experiment was spanned over six hours, started at 10:00 and ended on 16:00 on Saturday, March 26, 2011 at BTH Campus Grasvik, Karlskrona. All the instruments required for the execution of the experiment were provided to the participants who were located on reserved locations within the university premises. The clients’ machines were configured with the remote server to make sure that all the participants have connected with the server and their status online or offline is visible to all other participants. Actual experiment was started after 2 hours when all the participants were ready to start the programming and they were assigned to one of the treatment distributed PP or SP. All the required sessions were created and participants joined the sessions as pairs. Among 4 programs participants completed only 2 programs (program 1 and program 2) in the specified time. Participants were supposed to switch the roles frequently. This was observed and controlled by joining a session as a ‘Spectator’ – a predefined and specified role in XPairtise other than ‘Driver’ and ‘Navigator’. One experimenter visited the participants to resolve any issue or problem. The major issue resolved was related to the synchronization of code. A short break was also given to the participants after completing the first task.

At the end of the experiment participants answered the survey questionnaire specific to the social practices.

2.2.3.3 Data validation

After collecting the data, we checked that it was reasonable and we had collected it correctly, and the experiment was conducted in the same way as we wanted. To collect the correct data we used the data collection forms that were easy to understand and fill. Despite, we provided guidelines and trained each participant how to fill the forms and answer the survey questionnaire. In addition, the survey questionnaire was easy to understand because we avoided the ambiguous questions and tried to minimize misunderstandings.

The results of the experiment and the analysis are presented in Chapter 4 below.
3 RESULTS AND ANALYSIS OF SYSTEMATIC REVIEW

This chapter presents the results of the systematic review. We used narrative, tabulation or graphical form of analysis to present the results of the systematic review.

3.1 Classification of included studies

A total 68 studies were selected for the review. 21 (approx. 31%) were from distributed and 47 (approx. 69%) from co-located settings.

Among 47 studies in collocated settings, PP was empirically evaluated in 30 studies and in remaining 17 studies PP was empirically investigated as a pedagogical tool. Among 21 studies in DSD the evaluation of PP were focused in 9, PP supporting tools in DSD were discussed in 9 and 3 studies were mixed (where PP tools evaluation were performed by conducting the experiment using distributed PP). The classifications of the selected studies retrieved from 1998 – 2010 including are shown in figures below:

![Selected studies by year](image)

**Figure 3** Selected studies by year

![Selected Studies type](image)

**Figure 4** Selected studies types
3.2 Proposed benefits of PP

Many benefits of PP have been proposed and reported in literature such as improved code quality or better quality software, increased productivity or faster code production, knowledge sharing, increased programmer satisfaction and confidence, fewer defects etc. These benefits are studied (investigated, evaluated, etc.) in experiments, case studies, surveys and interviews among the selected studies for the review as Figure 4 shows ‘Selected study types’. Section 3.2.1 and 3.2.2 present the previous studies on PP benefits in both co-located and distributed settings separately.

3.2.1 PP benefits in co-located software development

This section presents the benefits of PP that are previously studied and presented in the collocated settings.

3.2.1.1 Productivity

Research on the impacts of pair programming on productivity has found that PP improves productivity [S27], [S35], [S36], [S41], [S59], and [S62]. Productivity was studied in different perspectives, for example, in terms of lines of code per hour [S35], [S36], work load per unit time [S27], [S36], successful completion of tasks [S41], [S59] and [S62]. Whereas, in [S43] it is also concluded that PP did not support the productivity - successful completion of tasks.

3.2.1.2 Quality

Many studies showed that PP increased the quality [S9], [S11], [S15], [S19], [S24], [S25], [S26], [S28], [S30], [S31], [S33], [S34], [S35], [S36], [S37], [S40], [S41], [S45] [S43], [S50], [S52], [S57], [S59], [S61], [S62], [S65], [S64], [S66]. Quality was studied in different perspectives using PP, for example, in terms of fewer defect, better design and code [S9], [S15], [S25], [S26], [S31], [S33], [S34], [S35], [S36], [S37], [S40], [S43], [S57], [S61], [S62], [S65], [S64], and [S66]. Quality in terms of (test) grades obtained [S11], [S19], [S24], [S41], and [S45]. Quality in terms of test cases passed [S28], [S30], and [S50]. Quality in terms of fulfills the requirements [S52] and [S59]. In studies [S53], [S55], and [S60] quality was studied in terms of effectiveness and concluded that PP did not impact the quality.

3.2.1.3 Performance

PP showed positive effects on performance [S10], [S18], [S20], [S21], [S23], [S29], [S30], [S46], [S52], and [S57]. Performance was studied in terms of grades obtained/ higher grades [S18], [S20], [S21], [S23], [S29], [S57] and in terms of accuracy of code produced [S10], [S30], [S46], and [S52]. Correctness in terms of passed test was also reported high using PP [S27], [S28].

3.2.1.4 Effort
Results of PP evaluation on effort showed both positive and negative impacts. Less effort is required using PP to accomplish the tasks [S9], [S24], [S26], [S27], [S31]. However results from other studies [S33], [S34], [S66] lead to opposite conclusion.

3.2.1.5 Confidence, satisfaction and enjoyment of work

It has been reported that the use of PP increase the confidence of the programmers/students [S1], [S3], [S12], [S13], [S14], [S18], [S23], [S45], [S57]. High level of satisfaction is also found [S1], [S3], [S11], [S13], [S14], [S28], [S37], [S52]. Enjoyment of work, positive attitude in programmers and in students towards working is also found that PP impacts positively [S1], [S3], [S14], [S19], [S20], [S21], [S23], [S28], [S29], [S31], [S34], [S35], [S37], [S41], [S45]. However in [S33] weak evidence is found regarding enjoyment of work i.e. participants reported that they did not like the collaborative work or the use of PP to complete the tasks.

3.2.1.6 Knowledge sharing/transfer and communication

In the literature the evidence is provided that the use of PP cause knowledge sharing/transfer and enhance learning. The evidence regarding the knowledge sharing/transfer is found in [S31], [S33], [S44], [S58], [S61]. However, no impact of PP on knowledge sharing is reported in [S41]. The effects of PP on enhanced learning are reported in [S19], [S20], [S21], [S31], [S35]. PP also resulted as a better communication practice among the students [S19].

3.2.1.7 Cost and scheduling

PP impacts cost in different phases, i.e. increases or decreases both which solely depends upon the tasks and requirements [S65]. It is also found that the use of PP in design and implementation phase cost more [S67]. Less scheduling conflicts arise among the programmer pairs [S34].

3.2.2 PP benefits in distributed software development

This section presents the overview of the previous studies as well as the related work and the benefits of PP studied in distributed environment.

B. Hanks, in [S2], [S22], [S49] and [S63] empirically evaluated the pair programming as a pedagogical tool for students who are learning to program. In study [S49] students participated as subjects in the experiments and they completed their assignments. The students were grouped into two groups, one group was allowed to pair programming using the VNC tool, while the second group paired while collocated. In [S2], the author studied the effects of pair programming on academic performance and on student confidence in both collocated and distributed settings. In [S22] conducted a survey to know the students (participated in the experiments) attitude towards pair programming. In [S63], the impact of pair programming on academic performance, time to complete programming assignments and scheduling conflicts were studied. Students were grouped into ‘tool group’ for distributed programming and ‘collocated group’ for collocated PP.

Edwards et al [S5] surveyed the pair programming as a pedagogical tool in online environment. First they conducted a pilot study using the Adobe’s Connect Now
software to support distributed pair programming. The survey evaluated three assessment areas: motivation and engagement with pair programming, self evaluation of learning outcome, and satisfaction with distance learning experience.

Baheti et al [S56] conducted an experiment to compare different working arrangements of student teams in both collocated and distributed environments; teams worked in collocated programmers, co-located pairs, distributed programmers and distributed pairs. Zacharis [S48] conducted an experiment to investigate effects of VPP on students’ performance (grades obtained), code productivity (lines of code/time for assignment), software quality (in terms of defects) and satisfaction in an introductory Java course. All students worked on homework programming assignments in distributed pairs and as solos.

Domino et al [S54] investigated the impacts of programmers’ experience on performance (accuracy of code produced) and on the programmers’ satisfaction that performed collaboration work using test-driven approach in both face to face and virtual settings. G. Canfora et al [S68] conducted an experiment and a replica to evaluate the impact of distribution on pair programming. They evaluated the effects of pair programming on effort (time to complete task) and quality in maintenance tasks in both collocated and distributed environment with students as participants.

The benefits of PP in distributed settings are described below.

### 3.2.2.1 Productivity

Research on the impacts of pair programming on productivity has found that PP improves productivity [S32], [S48], [S56]. Productivity was studied in different perspectives, for example, in terms of lines of code per hour [S48], [S56] and work load per unit time [S32].

### 3.2.2.2 Quality

PP increased the quality [S32], [S47], [S48], [S56], [S68]. Quality was studied in different perspectives using PP, for example, in terms of fewer defects [S32], [S47], [S48], [S68]. Quality in terms of (test) grades obtained [S56].

### 3.2.2.3 Performance

PP showed positive effects on performance [S2], [S48], [S49], [S54]. Performance was studied in terms of grades obtained/ higher grades [S2], [S48], [S49] and in terms of accuracy of code produced [S54].

### 3.2.2.4 Effort

Less effort is required using PP to accomplish the tasks [S63] and [S68] while performing PP in distributed settings.

### 3.2.2.5 Confidence, satisfaction, communication and attitude
It has been reported that the use of PP increases the confidence of the programmers/students [S49], [S2], [S54]. High level of satisfaction is also found [S5] and [S48]. Better communication in distributed pairs [S56]. Positive attitude towards PP is also found among the students [S22].

### 3.2.2.6 Scheduling

Less scheduling conflicts arise among the programmer pairs performing PP in distributed settings [S63].

### 3.2.3 Summary of the benefits

PP benefits in collocated settings are classified as productivity, quality, performance, effort, correctness, confidence, satisfaction, enjoyment of work and attitude, enhanced learning, knowledge sharing/transfer, cost and scheduling conflicts. Table 5 summarizes the benefits presented in collocated settings.

**Table 5 Summary of PP benefits in collocated settings**

<table>
<thead>
<tr>
<th>Category</th>
<th>Effects</th>
<th>Positive Effects</th>
<th>Negative/ No Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>• Lines of code per hours</td>
<td>[S27],[S35],[36],[S41],[S59],[S62]</td>
<td>[S43]</td>
</tr>
<tr>
<td></td>
<td>• Work load per unit time</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Successful completion of tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td>• Fewer or decreased errors/defects</td>
<td>[S9],[S11],[S15],[S19],[S24],[S25],[S26],[S28],[S30],[S31],[S33],[S34],[S35],[S36],[S37],[S40],[S41],[S45],[S43],[S50],[S52],[S57],[S59],[S61],[S62],[S65],[S64],[S66]</td>
<td>[S53],[S55],[S60]</td>
</tr>
<tr>
<td></td>
<td>• Better test grades obtained</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test cases passed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fulfills the requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Better design</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Better code (e.g. variable names, readability etc.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Effectiveness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>• Higher grades obtained</td>
<td>[S10],[S18],[S20],[S21],[S23],[S29],[S30],[S46],[S52],[S57]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accuracy of code produced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effort</td>
<td>• Time to complete the tasks</td>
<td>[S9],[S24],[S26],[S27],[S31]</td>
<td>[S33],[S34],[S66]</td>
</tr>
<tr>
<td></td>
<td>• More effort required to accomplish the tasks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correctness</td>
<td>• Passed test</td>
<td>[S27],[S28]</td>
<td></td>
</tr>
<tr>
<td>Confidence</td>
<td>• More or increased confidence in student and programmers</td>
<td>[S1],[S3],[S12],[S13],[S14],[S18],[S23],[S45],[S57]</td>
<td></td>
</tr>
</tbody>
</table>
PP benefits in distributed settings are classified as productivity, quality, performance, effort, correctness, confidence, satisfaction, enjoyment of work and attitude, enhanced learning, knowledge sharing/transfer, cost and scheduling conflicts. Table 6 summarizes the benefits presented in distributed settings.

**Table 6 Summary of PP benefits in distributed settings**

<table>
<thead>
<tr>
<th>Category</th>
<th>Effects</th>
<th>Positive Effects</th>
<th>Negative/ No Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>• Lines of code per hours</td>
<td>[S32],[S48],[S56]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Work load per unit time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td>• Fewer defects</td>
<td>[S32],[S47],[S48]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Better test grades obtained</td>
<td>[S56],[S68]</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>• Higher grades obtained</td>
<td>[S2],[S48],[S49],[S54]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Accuracy of code produced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effort</td>
<td>• Less or decreased time to complete the tasks</td>
<td>[S63],[S68]</td>
<td></td>
</tr>
<tr>
<td>Confidence</td>
<td>• More or increased confidence in students</td>
<td>[S2],[S49],[S54]</td>
<td></td>
</tr>
</tbody>
</table>
3.3 PP supporting tools in DSD

This section answers to the question “How can distributed PP be enabled through computer mediation?”

Several tools supporting DPP has been developed with different features. Mainly two kinds of collaboration tools exist; screen sharing and remote desktop applications (e.g. VNC or Microsoft Net-meeting), and awareness tools, further sub-classified into collaborative editors (e.g. SubEthaEdit) and specialized editors (e.g. Xpairtise), which extend an application in specific areas only to create a shared collaboration space [46]. This section presents an overview of the PP supporting tools in DSD.

Natsu et al [S39] presented the design of the COPPER system, a synchronous source code editor that allows two distributed software engineers to write a program using pair programming. They evaluated the COPPER system by conducting an experiment using the students as participants. The results of the experiment provided evidence that COPPER system successfully supported DPP. Favela et al [S47] also conducted an experiment to evaluate the COPPER system. They investigated the effects of pair programming on code quality in terms of errors found with two distributed teams. One distributed team used NetMeeting to support application sharing mode while other used COPPER editor in collaborative mode. The experiment results provided the evidence that tasks completed with COPPER in distributed mode resulted in a slightly lower number of errors found.

R. Duque [S32] presented the COLLECE (COLLoborative, Edition, Compilation and Execution) system, a synchronous distributed groupware system with specific support for the creation of programs through DPP practices. He conducted a series of experiments to study the influence of DPP on work process (productivity - in terms of workload per unit of time) and the quality of software products built using COLLECE as technological platform with students as participant in a university setting. Pair programming was performed among groups of distributed pairs and solos. It was concluded from the experiment results that distributed pair programmers spend more time to complete task than solos because of time spend on communication and coordination interactions. It was also reported that programs quality (fewer errors) developed by distributed programmers was higher than that of solo programmers.

Ho et al [S4] designed and developed Sangam (Hindi word which means “confluence” in English), a tool for distributed pair programming as a plug-in for Eclipse. Sangam provides: Editor, launch, resource and refactoring synchronization. Navoraphan et al [S6] discussed Sangam and Facetop to support DPP. Sangam facilitates DPP by sending events back and forth between a driver and a navigator.

<table>
<thead>
<tr>
<th>Satisfaction</th>
<th>Students/programmers satisfied more with the solution</th>
<th>[S5],[S48]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>Better communication in distributed teams</td>
<td>[S56]</td>
</tr>
<tr>
<td>Attitude</td>
<td>Positive attitude</td>
<td>[S22]</td>
</tr>
<tr>
<td>Scheduling</td>
<td>Low level of scheduling conflicts arise</td>
<td>[S63]</td>
</tr>
</tbody>
</table>
working under the Eclipse development environment. Facetop developed for the purposes of facilitating DPP by providing a sense of physical presence for the users. The Sangam and Facetop combination promises to be the effective collaborative software development tool for distributed partners as it provides same integrated development (Eclipse) and allow them to communicate visually while they work.

Salinger et al [S7] provided an overview of the Eclipse plug-in Saros, a software implementation supporting distributed PP. Saros focuses on the awareness - *a person’s understanding about the shared environment and activities of others*, issues in collaboration work when programmers not co-located and offers some solutions to the awareness issues; six features are introduced in Saros for this purpose.

Moreno et al [S38] presented JeCo (Jeliot Collaborative), an integration of *Jeliout 3* – a program visualization tool and *Woven Stories* – a tool supports collaborative writing, to support the pair programming. JeCo is a simple tool and intended to support novices in learning process.

Bowen & Maurer [S42] described the MILOS that supports project coordination, information routing, team communication, pair programming and experience management. MILOS also supports the coordination and initiation of pair programming sessions with Microsoft NetMeeting.

Boyer et al [S8] presented the RIPPLE (Remote Interactive Pair Programming and Learning Environment), a development environment for distributed synchronous collaborative programming as a plug-in of Eclipse.

Flieger & J. D. Palmer [S16] describes the JavaGrinder, a web-based platform designed to support students using pair programming. JavaGrinder aims to increase the retention of students for programming courses.

Meneely & Williams [S17] discussed and evaluated the Jazz, a synchronous and collaborative development platform developed by IBM and available as Eclipse plug-in. Jazz integrates support for team development in one platform, supports version control, track defects, and synchronizes teamwork.

Schummer & Lukosch [S51] presented XPairtise, a plug-in for Eclipse that allows instant pair programming in distributed development teams, to support the social practices of pair programming. They evaluated XPairtise support for social practices (*Coordination, Coding, Communication, Teaching and Testing*) in two long term projects and resulted that XPairtise supports the social practices for pair programming in distributed projects.

### 3.4 Analysis

This systematic literature review introduces the pair programming benefits that have been studied in the literature. Pair programming mainly studied in two main areas in the selected studies, one for the purpose of software development and second as an effective pedagogical tool.

The proposed benefits include productivity, quality, performance, cost, effort and correctness specific to the programming activities, etc. Other proposed benefits are related to social practices (also called human or social factors) are satisfaction,
enjoyment of work, attitude, enhanced learning, knowledge sharing/transfer and scheduling conflicts.

One main finding is that PP proposed benefits are studied in different perspectives, for example, productivity is studied in terms of lines of code per hours, work load per unit time, and successful completion of tasks. In few studies some of the proposed benefits of PP such as productivity, quality, effort, enjoyment of work, knowledge sharing, and cost are also negatively reported. Therefore, it can be said that there are mixed findings identified and reported in the literature.

There is less evidence found regarding the proposed benefits in distributed settings (approx. 18%) than co-located settings (approx. 69%), whereas remaining 13% studies reported PP tools in DSD. Figures below show the ‘positive’ and ‘negative/no effects’ of PP in collocated and distributed settings where number below each benefit, for example, ‘productivity 6’ means that productivity is studied in 6 selected research articles.

![Figure 5 Positive effects of PP in collocated settings](image)

![Figure 6 Negative/No effects of PP in collocated settings](image)
3.5 Validity threats related to SLR

To study the validity threats is the last phase of a systematic review as suggested by [31] and [47]. Several validity threats were identified from various threats and sources to the conducted systematic review for this study as discussed below:

3.5.1 Conclusion validity

Conclusion validity is related to the reliability of the research study results [36]. It is also concerned with issues that affect/limit the ability to draw the correct conclusions from the research [36].

For a systematic review the main risk concerns the ability of the authors to select accurate research publications for the review and data extraction. The reliability, to select accurate papers for the review, of the systematic literature review is increased by following the review protocol (See section 2.1.2). Moreover, one of the main aims of defining a review protocol is to reduce researcher bias [31] by defining the inclusion/exclusion criteria explicitly. Inconsistency in the extracted data or incorrect terms leads to incorrect classification of the research results [48]. To minimize the data extraction threat and to increase the consistency of the extracted data, authors used a designed data extraction form.

3.5.2 Internal validity

Internal validity is concerned with the deductions, whether accurate or not, from the gathered data [49].

A threat with the systematic literature review is associated with publication biasness. To overcome this threat the defined review protocol is strictly followed where inclusion/exclusion criteria, search string, selected electronic databases etc are described explicitly. Internal validity is also minimized by extracting both positive and negative findings of the primary studies into the data extraction form, so that they can
be reported in the report. In addition, we did not try to interpret the primary study’s findings. As a result, the potential threat to reliability of data is minimized. One more potential threat is associated to grouping the benefits reported in primary studies and therefore it causes the inaccurate grouping of benefits. To overcome this threat, we extracted the data with explicit explanations of proposed benefits.

3.5.3 Construct validity

Construct validity concerns the ability to generalize the results [36]. It is possible that some useful aspects of research articles are not understood by the authors and missed. This risk is reduced by presenting the aims of the review are to explore the benefits of PP. Further, the limited scope of the review also reduces the possibility of misinterpretations.

3.5.4 External validity

A threat to the external validity might be the missing of primary studies. As, the systematic literature review was conducted between the year 1998 and 2010 including and first primary study found in 1998. This was a low potential threat to the external validity for this systematic review.
4 RESULTS AND ANALYSIS OF EXPERIMENT

This chapter presents results of the experiment and analysis of collected data by using statistical techniques. The collected data is analyzed according to the design type of experiment in three steps i.e. descriptive statistics, data reduction and hypothesis testing [36].

First, descriptive statistics is utilized to evaluate collected data from the experiment. Descriptive statistics summarizes the collected data in a clear and understandable way [36]. It is also used for graphical presentation of collected data, for example, box plot to analyze how collected data is grouped [36]. The common measures of descriptive statistics are: mean value, standard deviation, kurtosis and skewness [36].

Second, data reduction is performed. Data reduction is the process of finding and reducing outliers from data set [36]. It is related with data validation that deals with identifying false data points based on the execution of experiment [36].

Third, hypothesis testing is performed on the collected data to see whether the null hypothesis $H_0$ can be rejected or not with significance level and to draw some sort of conclusion about the experiment results [36]. Significance level or p-value is used to reject null hypothesis against the alternative hypothesis [36]. Commonly used p-value is 0.05. Null hypothesis is rejected in case if p-value from statistical test is less than value of p=0.05.

There are many statistical tests available and appropriate test should be selected based on assumptions and design type of the experiment [36]. As design of the experiment is ‘One factor with two treatments’, therefore, according to [36] four different tests can be applied for hypothesis testing including two parametric test (t-test and F-test) and two non-parametric tests (Mann-Whitney and Chi-2) [36].

Parametric tests are assumed to utilize if sampling data is normally distributed, and parameters are measured at least on an interval scale [36]. On the basis of the collected data from two treatments, trials, and number of participants (N=9) for this experiment, t-test is applied. In addition, there is no dependency exists among two treatments. Independent sample t-test with equal sample variances is utilized for comparison of two sample means and this test is also used to confirm hypothesis separately for each task performed in experiment. SPSS\(^9\) software is utilized for applying independent samples t-test on collected data; F-test is run as part of t-test and it is used to check the equality of variances of two data sets. Z-test can be used instead of t-test but Z-test is preferable when sample size is more than 30 and t-test is used for sample size less than 30\(^{10}\).

Non-parametric tests are assumed that sampling distribution is not normal. Chi-2 is not appropriate for this experiment results because such tests require data in form of frequencies [36]. Mann-Whitney test is an alternative for t-test, based on ranks of sample data and can be used to compare unpaired groups [36]. This test is not suitable for small samples sizes and has a very little power if collective size of two samples is


\(^{10}\) Please see: [http://www.differencebetween.net/miscellaneous/difference-between-z-test-and-t-test/](http://www.differencebetween.net/miscellaneous/difference-between-z-test-and-t-test/)
less than seven\textsuperscript{11}. Collective size of two samples for each task is less than seven in this experiment.

4.1 Analysis of the direct measurement

In the following sections, collected data from the experiment is analyzed using descriptive statistics and hypothesis testing is performed to evaluate the performance. Performance of each experimental task is evaluated separately.

4.1.1 Performance

Performance was measured in \textit{minutes} - time spent to complete the task in both PP and SP. In the context of this study, less time utilized to complete the task means performance was high.

4.1.1.1 Descriptive statistics

Mean value or average value is the measure of central tendency that indicates the middle of data set \textsuperscript{36}. Standard deviation of data set is the measure of dispersion that indicates the level of variation from central tendency \textsuperscript{36}. Normal distribution of data is presented in form of symmetric curve. There are different measures used to check the sample data for its normal distribution such as skewness and kurtosis. Data curve is skewed towards right or left if value calculated for skewness is measured as positive or negative respectively and curve is exactly symmetric if value for skewness is zero. It can be concluded that skewness value near to zero is the indication that data is normally distributed. Kurtosis is another measure for normal distribution. It is used to measure whether data set is flat or peaked relative to normal distribution\textsuperscript{12}. Kurtosis could not found for this study due to small sample size used for each treatment in the experiment.

Box plot is good for visualize distribution of data around median. Middle bar in box plot is the median, values outside the lower and upper tails are called outliers that are meaningless values and can be truncated, while values around the lower tail will indicate the high performance for this experiment \textsuperscript{36}. Box plots of collected data for Task1 and Task2 are presented below.

4.1.1.1.1 Descriptive statistics for Task1

Table 7 below shows the collected data for Task1 of both treatments, PP and SP.

\begin{table}[h]
\centering
\begin{tabular}{|l|c|l|c|}
\hline
\textbf{Treatment 1 (PP)} & \textbf{Time (minutes)} & \textbf{Treatment 2 (SP)} & \textbf{Time (minutes)} \\
\hline
Pair 1 & 60 & Solo 1 & 47 \\
Pair 2 & 43 & Solo 2 & 65 \\
Pair 3 & 36 & Solo 3 & 45 \\
\hline
\textbf{Total} & 139 & \textbf{Total} & 157 \\
\hline
\end{tabular}
\caption{Collected data for task1}
\end{table}

\textsuperscript{11} \url{http://helpdesk.graphpad.com/help/Prism5/prism5help.html?how_the_mann_whitney_test_works.htm}
\textsuperscript{12} \url{http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm}
Descriptive statistics: mean value, standard deviation, skewness and kurtosis of collected data for Task1 are presented in Table 8 below.

**Table 8** Descriptive statistics of task1

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Mean (x)</th>
<th>Standard Deviation (s)</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>46.3333</td>
<td>12.34</td>
<td>0.46</td>
<td>-</td>
</tr>
<tr>
<td>SP</td>
<td>52.3333</td>
<td>11.01</td>
<td>0.68</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 8 descriptive statistics for task1 shows that time spent, $\bar{x}_{PP}$ is less than $\bar{x}_{SP}$, to complete the task1 in DPP is slightly less than that of in SP. This is also confirmed by the box plots in Figure 8 below:

![Box plots for performance of two treatments for task1](image)

**Figure 8** Box plots for performance of two treatments for task1

4.1.1.1.2 Descriptive statistics for Task2

Table 9 shows the collected data for Task2 of both treatments, PP and SP.

**Table 9** Collected data for task2

<table>
<thead>
<tr>
<th>Treatment 1 (PP)</th>
<th>Time (minutes)</th>
<th>Treatment 2 (SP)</th>
<th>Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1</td>
<td>65</td>
<td>Solo 1</td>
<td>58</td>
</tr>
<tr>
<td>Pair 2</td>
<td>40</td>
<td>Solo 2</td>
<td>70</td>
</tr>
<tr>
<td>Pair 3</td>
<td>60</td>
<td>Solo 3</td>
<td>60</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>165</strong></td>
<td><strong>Total</strong></td>
<td><strong>188</strong></td>
</tr>
</tbody>
</table>

Descriptive statistics: mean value, standard deviation, skewness and kurtosis of collected data for Task2 are presented in Table 10 below.
Table 10 Descriptive statistics for task2

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Mean ((\bar{x}))</th>
<th>Standard. Deviation((s))</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>55</td>
<td>13.23</td>
<td>-0.60</td>
<td>-</td>
</tr>
<tr>
<td>SP</td>
<td>62.67</td>
<td>6.43</td>
<td>0.63</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 10 shows that time spent, \(\bar{x}_{PP}\) is less than \(\bar{x}_{SP}\), to complete task2 in pair programming is slightly less than that of solo programming. This is also confirmed by the box plots in Figure 9 below:

![Boxplots for performance of two treatments for task2](image)

**Figure 9** Boxplots for performance of two treatments for task2

4.1.1.2 Data reduction

Data set is not reduced as there was no outlier or false data points found in collected data from the experiment because collected data for two treatments was small.

4.1.1.3 Hypothesis testing

Based on the collected data, specifically time spent to complete the tasks, the performance in two treatments is analyzed. A two tail independent t-test with equal sample variances was chosen to compare the means of two data sets. Levene’s test for equality of variances is also performed to decide whether two sample variances are equal or not, this test is run as part of SPSS independent t-test procedure.

4.1.1.3.1 Performance in Task1
In task 1, collected data is considered to be normally distributed as skewness value is nearly zero as shown in Table 8. Based on above data (see Table 7), two independent samples t-test with equality of variances is performed with significant level ($\alpha = 0.05$). Table 11 shows the calculated values of Independent sample t-test.

**Table 11** The results of t-test on performance of two treatments for task1

<table>
<thead>
<tr>
<th>Performance</th>
<th>Levene's Test for Equality of Variances</th>
<th>t-test for Equality of Means</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>Sig.</td>
</tr>
<tr>
<td>Equal variances assumed</td>
<td>.032</td>
<td>.868</td>
</tr>
<tr>
<td>Equal variances not assumed</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

From Table 11 we can see that the significance value of Levene’s test for equality of variances (0.868) is greater than 0.05. According to SPSS software, it means that it assumes the variances of two samples are equal. However, the significance value (0.564) for the test is higher than 0.05, which indicates that there is no significant difference between the means of the two treatments.

Based on the analysis of the collected data of performance in DPP and SP for task1, the null hypothesis $H_0$ is not rejected because there is no significant difference between the two means. However, on the bases of the results, particularly descriptive statistics (Section 4.1.1.1.1 above), performance in DPP is slightly higher than the performance in SP for task 1.

4.1.1.3.2 Performance in Task2

In task 2, collected data is considered to be normally distributed as skewness value is nearly zero as shown in Table 10. Based on above data (see Table 9), two independent samples t-test with equality of variances is performed with significant level ($\alpha = 0.05$). Table 12 shows the calculated values of Independent sample t-test.
From Table 12 we can see that the significance value of Levene’s test for equality of variances (0.184) is greater than 0.05. According to SPSS software, it means that it assumes the variances of two samples are equal. However, the significance value (0.418) for the test is higher than 0.05, which indicates that there is no significant difference between the means of the two treatments.

Based on the analysis of the collected data of performance in DPP and SP for task2, the null hypothesis $H_0$ is not rejected because there is no significant difference between the two means. However, on the bases of the results, particularly descriptive statistics (Section 4.1.1.1.2), performance in DPP is slightly higher than the performance in SP for task 2.

### 4.1.2 Social practices

The social practices (communication, knowledge sharing/transfer, confidence about solutions, and satisfaction and enjoyment) are evaluated based on descriptive statistics. The participants answered the survey questionnaire based on their experience and judgments at the end of the experiment. Likert scale is used to collect the survey data. The points given to each category in a five-point Likert scale as follows: strongly agree= 1, agree= 2, neither= 3, disagree= 4, and strongly disagree= 5.

Descriptive statistics for Likert scale data includes mean value, mode, positive percentage, negative percentage and neutral percentage of participants’ responses for the survey questionnaire. Mode is the most frequent value of data set [36]. Positive percentage is calculated for the points corresponds to ‘strongly agree’ and ‘agree’ options, neutral percentage is calculated for neutral responses or points for ‘neither’ option and negative percentage is calculated for the points corresponds to ‘disagree’ and ‘strongly disagree’ options. Mean value and positive percentage of collected data
is used to make decision about the hypothesis for each social practice (see Section 2.2.2.2 above).

Each social practice is described separately below with the questions, participants’ responses to each question, descriptive statistics for each practice based on the responses and the percentage of the responses to the questions.

4.1.2.1 Communication

Table 13 below shows the questions related to communication in distributed settings that were part of the survey questionnaire.

**Table 13 Questions for communication**

<table>
<thead>
<tr>
<th>Questions for communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
</tbody>
</table>

Table 14 shows responses of the participants about communication in distributed settings for the questions shown in Table 13.

**Table 14 Responses for communication**

<table>
<thead>
<tr>
<th>Responses for Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question #</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
</tbody>
</table>

Table 15 shows the descriptive statistics for communication of the collected data as shown in Table 14.

**Table 15 Descriptive statistics for communication**

<table>
<thead>
<tr>
<th>Question #</th>
<th>Mean value</th>
<th>Mode</th>
<th>Positive Response (%)</th>
<th>Neutral Response (%)</th>
<th>Negative response (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>2</td>
<td>2</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>2.11</td>
<td>2</td>
<td>66.67</td>
<td>33.33</td>
<td>0</td>
</tr>
</tbody>
</table>

The Figure 10 shows positive, neutral and negative percentage of the participant’s responses for communication using PP in distributed settings.
The descriptive statistics of the responses for communication indicates that pair programming enhances the communication among the programmer pairs in distributed settings. All participants were ‘agree’ on the statement that “partner active communication allows me to more active in expressing my views”. However, 67% were respond positively (agree/strongly agree) and 33% were neutral (‘neither’) on the statement “My partner described his/her views clearly and I was able to fully understand him/her”.

### 4.1.2.2 Knowledge sharing /transfer

Table 16 below shows the questions for knowledge sharing/transfer in distributed settings that were the part of survey questionnaire.

<table>
<thead>
<tr>
<th>Questions for Knowledge sharing/transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
</tbody>
</table>

Table 17 shows responses of the participants related to knowledge sharing/transfer in distributed settings for the questions as shown in Table 16.

<table>
<thead>
<tr>
<th>Responses for knowledge sharing/transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question #</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
</tbody>
</table>

Table 18 shows the descriptive statistics for knowledge sharing/transfer of the collected data as shown in Table 17.
Table 18 Descriptive statistics for knowledge sharing/transfer

<table>
<thead>
<tr>
<th>Question #</th>
<th>Mean value</th>
<th>Mode</th>
<th>Positive Response (%)</th>
<th>Neutral Response (%)</th>
<th>Negative response (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1.67</td>
<td>2</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>1.11</td>
<td>1</td>
<td>88.89</td>
<td>0</td>
<td>11.11</td>
</tr>
<tr>
<td>03</td>
<td>1.11</td>
<td>1</td>
<td>88.89</td>
<td>0</td>
<td>11.11</td>
</tr>
</tbody>
</table>

The Figure 11 shows positive, neutral and negative percentage of the participant’s responses for knowledge sharing/ transfer using PP in distributed settings.

Figure 11 Participants responses percentage for knowledge sharing

The descriptive statistics of collected data about knowledge sharing/transfer indicates that PP is an effective technique for knowledge sharing/transfer in distributed teams. All the participants respond positively (agree/strongly agree) on statement “Knowledge sharing is higher using pair programming in distributed teams.” while 89% agree on statement that they can complete task individually after completing it in pair programming. The results indicate that knowledge was transferred among individuals during pair programming worked in pairs. In addition, 89% participants respond positively (agree/strongly agree) that pair programming is a good learning practice from peers.

4.1.2.3 Satisfaction and confidence about solution:

Table 19 below shows the questions related to satisfaction and confidence about solution using PP in distributed settings that were the part of survey questionnaire.

Table 19 Questions for satisfaction and confidence about solution

<table>
<thead>
<tr>
<th>Questions for Satisfaction and Confidence about solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
</tbody>
</table>
Q3. I can produce a better solution if I work in pair programming than in solo programming.

Q4. In pair programming, I believe that I had an easier time reaching the solutions.

Q5. I am satisfied to work with partner to handle tasks using pair programming.

Q6. I feel that my overall experience to work on programming tasks in pair programming is better than solo programming.

Table 20 shows responses of the participants related to satisfaction and confidence about solution in distributed settings for the questions as shown in Table 19.

Table 20 Responses for satisfaction and confidence about solution

<table>
<thead>
<tr>
<th>Question #</th>
<th>Strongly Agree</th>
<th>Agree</th>
<th>Neither</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>4</td>
<td>4</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>1</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 21 shows the descriptive statistics for satisfaction and confidence about solution of the collected data as shown in Table 20.

Table 21 Descriptive statistics for satisfaction and confidence about solution

<table>
<thead>
<tr>
<th>Question #</th>
<th>Mean value</th>
<th>Mode</th>
<th>Positive Response (%)</th>
<th>Neutral Response (%)</th>
<th>Negative response (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1.78</td>
<td>1,2</td>
<td>88.89</td>
<td>0</td>
<td>11.11</td>
</tr>
<tr>
<td>02</td>
<td>2.11</td>
<td>3</td>
<td>55.56</td>
<td>44.44</td>
<td>0</td>
</tr>
<tr>
<td>03</td>
<td>1.78</td>
<td>2</td>
<td>88.89</td>
<td>11.11</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>1.89</td>
<td>2</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>1.89</td>
<td>2</td>
<td>88.89</td>
<td>11.11</td>
<td>0</td>
</tr>
<tr>
<td>06</td>
<td>1.67</td>
<td>1,2</td>
<td>66.67</td>
<td>22.22</td>
<td>11.11</td>
</tr>
</tbody>
</table>

The Figure 12 shows positive, neutral and negative percentage of the participant’s responses for satisfaction and confidence about solution using PP in distributed settings.
The descriptive statistics of collected data indicates that satisfaction and confidence about solution was higher in PP. 89% responses showed that participants were satisfied and confident about the correctness and completeness of the solutions, however one participant was disagree. A large difference found for the quality of programs developed in pair programming, 56% were ‘agree’ while 44% responses were ‘neutral’. Participants’ positive responses (89%) showed that a better solution can produce in pair programming. All the participants were agree/strongly agreed to an easy time reaching solution in pair programming. Most of the participants were satisfied to working with a partner. 67% participants respond positively with their overall experience, while 22% were neutral and one participant was ‘disagree’.

4.1.2.4 Enjoyment of work

Table 22 below shows the questions for enjoyment of work in distributed settings that were the part of survey questionnaire.

Table 22 Question for enjoyment of work

<table>
<thead>
<tr>
<th>Questions for Enjoyment of Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 To work in pair programming is more enjoyable compared to solo programming.</td>
</tr>
<tr>
<td>Q2 I would like to work again using pair programming in my future projects.</td>
</tr>
</tbody>
</table>

Table 23 shows responses of the participants related to enjoyment of work in distributed settings for the questions in Table 22.

Table 23 Responses for enjoyment of work

<table>
<thead>
<tr>
<th>Responses for Enjoyment of work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question #</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
</tbody>
</table>
Table 24 shows the descriptive statistics for enjoyment of work of the collected data as shown in Table 23.

Table 24 Descriptive statistics for enjoyment of work

<table>
<thead>
<tr>
<th>Question #</th>
<th>Mean value</th>
<th>Mode</th>
<th>Positive Response (%)</th>
<th>Neutral Response (%)</th>
<th>Negative response (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1.67</td>
<td>1</td>
<td>77.78</td>
<td>11.11</td>
<td>11.11</td>
</tr>
<tr>
<td>02</td>
<td>1.78</td>
<td>1</td>
<td>77.78</td>
<td>11.11</td>
<td>11.11</td>
</tr>
</tbody>
</table>

The Figure 13 shows positive, neutral and negative percentage of the participant’s responses for enjoyment of work using PP in distributed settings.

Figure 13 Participants responses percentage for enjoyment of work

The descriptive statistics of collected data for enjoyment of work indicates that enjoyment of work is higher in pair programming. 78% of participants were enjoying their work more in pair programming than solo programming while one student was neutral and one give negative response. In addition, 78% wants to work again in pair programming in their future projects that indicates positive attitude towards pair programming.

4.1.2.5 Participants answers to the open ended questions

There were three open ended questions part of the survey questionnaire to know the participants’ opinion about the weaknesses and strengths of PP (see Table 25). Participants were supposed to answer the questions based on their previous experience in industry and experience gained in the experiment.

Table 25 Open ended questions

<table>
<thead>
<tr>
<th>Open ended questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
</tr>
<tr>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
</tr>
</tbody>
</table>
The summary of the answers provided by the participants is as follows:

Most of the participants showed positive attitude towards PP in distributed settings. PP is considered as a good tool for learning from peers. PP is productive, more fun while working and lot of work can be done. It is helpful to enhance the design and solution quality of the work. PP is more supportive than SP. More discussion while working enhances the speed of work. The sharing of ideas and partners’ continuous review suggests different solutions. However, one of the participants suggests that there should be a parameter like level of skills to be considered while making pairs. The parameters to be considered like skills or expertise levels are studied in some previous PP studies, for example, expertise levels is studied in [50] and [51].

The participants experiences about the problems or weaknesses of PP reported that lack of coordination due to expertise and skill level utilize more time to absorb partners logic and therefore it’s difficult for critical timeline of performing task. Difference of opinions and solutions arise conflicts in teams cause to stop and effect working and overall development process. Sometime more dominant partner tries to get his/her solution to be accepted. Impacts of conflicts on results of the collaborative software development are studied in [52].

In PP if time is utilized in proper way, under managed and controlled settings then PP is quite useful to meet the deadlines. More productivity in less time in PP than SP when skill levels are same.

4.2 Analysis

4.2.1 Performance

The results of our experiment indicate the average completion time in PP was less than in SP for the tasks developed in the experiment. Task1 was completed in 8% less time than in SP. Task 2 was completed in 10% less time than in SP. Therefore, on the bases of descriptive statistics as discussed in Section 4.1.1.1 it can be said that performance in PP was higher than in SP in both tasks.

Distributed programmers did not achieve statistically significance better results than the solo programmers as presented in Section 4.1.1.3. However, it should be noted that statistical significance is difficult to obtain in pair programming experiments because there was 1) limited number of trials and 2) limited number of observations i.e. small sample size in our experiment, similar result is also discussed in [S11].

There are some others factors that should be considered for PP in distributed settings. Communication is a big challenge in distributed settings among the programmers. It might possible that programmers spent more time in communication while they are geographically dispersed than in co-located settings. Coordination may take some time too. In addition, lack of participants’ previous experiences in distributed software development and of pair programming for our experiment may affect the results.
However, we checked the quality (coding standard and correctness) of the developed tasks in both PP and SP on the basis of following quality criteria: I) Coding standard and II) Test cases passed (see Appendix 8.9).

Each item in coding standard and for each test case a value of 1 or 0 was given. Value was 1 if test case or coding standard is ‘Yes’ otherwise 0.

The quality of task1 was at same level i.e. all test cases were resulted ‘Yes’ for task1 developed in both PP and SP. However, quality on the bases of coding standard was lower in SP than in PP i.e. the programmers’ pairs followed the coding standards. The possible reason for following a coding standard in PP might be to facilitate the reviewer/navigator or partners requested each other to follow the coding standards.

The quality of task2 was at different level in PP and SP as well as within the pairs and solos i.e. all test cases were not resulted ‘Yes’ for task2 either developed in SP or in PP. However, passed test cases were slightly high in task2 developed in pairs. The possible reasons might be: I) individuals were assigned to new pairs for task2 (more conflicts due to skills and expertise level as also reported by the participants), II) programmers lost interest as time passed and then programmers did not develop the task seriously. However, quality on the bases of coding standard was again higher in PP than in SP.

On the bases of quality assessment criteria, we can summarize that programmers in PP wrote correct and more comprehensive code, readability was higher and code was well structured, than the programmers in SP.

4.2.2 Social practices

The average positive response shows that PP impacts social practices positively among distributed programmers. Average positive response is the average of the responses for ‘Strongly Agree’ and ‘Agree’ for each question in each corresponding social practice. Communication as one of the core function of negotiation and cooperation that allows the information and ideas to be exchanged is shown 83.33% by participants. The average positive response for knowledge sharing/transfer is shown 92.60% by participants, i.e. PP is a good practice for knowledge sharing/transfer. Satisfaction and confidence about solution is found higher among participants (average positive response is 81.48%). Participants enjoyed the work more in PP than SP; the average positive response found is 77.78% for enjoyment of work.

4.3 Validity threats related to experiment

There can be number of threats to the validity of the results from the experiment. Following level of validity such as internal validity, external validity, conclusion validity, and construct validity are considered [36]:

4.3.1 Conclusion validity

Conclusion validity is concerned with the relationships between the treatment and the outcome. It concerns with the issues that affect the correct conclusion and helps to determine the significance of statistical relationships [36].
The main threat to the conclusion validity is the quality and correctness of the collected data, time in minutes, from the experiment. The participants were expected to report the actual time spent to develop programs and not the time they might utilize to know about each other and skill level. It was ruled out by providing a designed data form to enter the data. All the participants were bound to use the designed data forms in order to provide the data during the experiment.

The low sample size and few trials resulted small experimental data, therefore, it has quite low statistical power. This is a threat that affects the ability to draw the correct conclusion about the relations between the treatment and the outcome [36]. The statistical power can be increased in future by increasing the sample size and trials.

The wrong selection of statistical test for hypothesis testing can also affect the conclusion because the statistical significance of the experimental data can be affected by the use of statistical tests. To overcome this threat the appropriate test, t-test, is applied for small sample sizes.

Heterogeneity of subjects in a group increases probability of variation due to individual differences is larger than due to the treatment. This effect of heterogeneity was minimized by providing the training to the participants.

4.3.2 Internal validity

Internal validity deals with the question whether the outcome (the effect) is caused by independent variables or by other factors, without the researcher’s knowledge [36].

The behavior of the subjects (participants) can be changed based on the fact that they are part of the experiment and they can act differently in the experiment. Therefore, they can affect the experiment results. This is, therefore, bias arises as they believe that they are being evaluated during the experiment and inaccurate data can be an outcome. To overcome, we realized the participants that the aim of the experiment is to evaluate the pair programming practice, not their skills during the selection and training.

Maturation is the effect of that the subjects react differently as time passes [36]. This was a threat in this study, since participants spent time to write programs. This threat was minimized by providing a small break after the completion of first task.

Poorly designed data collection forms and other instruments may have huge impact on the experimental results; known as instrumentation [36]. Instrumentation effect was minimized by providing all the instruments and data collection forms to the participants.

Selection is the effect of natural variation in human performance that can vary depending on how the subjects are selected from a large group [36]. To minimize this effect in this study, all participants were organized through convenience sampling – the nearest and most convenient persons were selected as subjects [36]. In same way, the threat to mortality is minimized by choosing the subjects on the basis of relevant background and experience in Java.

4.3.3 Construct validity
Construct validity concerns generalizing the result of the experiment to the concept or theory behind the experiment [36]. Construct validity threats for this study can be:

The lack of understanding of the distributed development tools may affect the results of the experiment. In order to reduce this threat, the participants were trained well to use the selected tools in the experiment, preferably in distributed environment, to get reliable and accurate data.

The experiment was conducted in a university environment in which students participated having different background knowledge, experience and skill level. Although students were trained to minimize this threat but it might be difficult to generalize the findings from the experiment for industry or organizations.

Experimental tasks complexity may affect the experimental results. Tasks with low or high complexity can produce different results of the experiment. To minimize this threat, the tasks chosen have not higher level of complexity and therefore minimize the variations in experimental results.

4.3.4 External validity

These threats are related with the conditions that limit the ability to generalize the experimental results to industrial practice [36].

The sample size (N=9) for this experiment is small and there is a risk to generalize the results for industry. This is a threat in this study because large sample can provide better results. To minimize this risk, the selected participants (students) had professional development experience and therefore results can be generalized to some extent to industry. In addition, using students in a controlled experiment is not always a threat for external validity [53].

The small duration and low complexity tasks are also a risk to generalize the results of our experiment.
5 *DISCUSSION*

This chapter gives a discussion on major findings of this thesis.

The SLR was aimed to answer RQ1 – to explore the benefits of PP in collocated and distributed settings as well as PP supporting tools in DSD. Experiment was aimed to answer RQ2 – to study the possible impacts of PP on social practices, and RQ3 – to investigate the effects of PP on performance in DSD.

Several benefits of pair programming over individual programming with empirical evidence are reported in the literature. Pair programming is an effective tool for quality software as well as a pedagogical tool. PP as pedagogical tool is considered as an effective practice for the students who learn to program. The employment of PP enhances students learning and results in completion of programming assignments on time with fewer defects in the code. Furthermore, students obtain higher grades in the exams, as findings of SLR.

The results of this study suggest that individuals perceive higher level of social practices - *communication, knowledge sharing/transfer, satisfaction and confidence about the solution, and enjoyment of work* while working in pairs. These findings are consistent with the findings of SLR.

Communication is an important issue among the members of distributed teams. For successful software projects communication plays a vital role. Distributed PP is an effective practice for communication among distributed team members as reported ‘very good’ in SLR findings [S56] as well as in our study it is reported high (67%). Continuous communication among programmers in distributed teams facilitates programmers to produce quality work – defect/error free code, more test cases passed. Our experimental results also show that high quality code produced in distributed pairs. The communication increases overall information flow among distributed team members and, therefore, more knowledge sharing/transfer and learning in distributed team members, in line with the results [S31].

Knowledge sharing/transfer is the main advantage of pair programming as reported in literature [S20, S21, S35, S61] and our study results also confirm SLR finings. Knowledge transfer/sharing is improved in pair programming as compared to solo programming, specifically in distributed settings, as it was not studied in distributed settings before according to our SLR results. Our study indicates that programmers can complete tasks individually after completing it in pairs that confirms the result of [S33] that measures the knowledge transfer by understanding of each module in a task.

Results of our study indicate that distributed pair programming is a good learning practice from peers which confirms study results of [S20, S21, S61, S35]. In addition, PP also enhances learning from peers in distributed teams and growth of knowledge allows completing the tasks in short time confirming the results in [S30]. In DPP, one member’s active communication in distributed pairs makes other to be more active to express their views during the development process that verifies the claim in related work [S19]. Consequently, distributed programmers show higher level of satisfaction and confidence about the solution as well as they enjoy the work more, work while talking.
However, communication and coordination in some decision making aspects for programming assignments need consensus on a preferred approach. This adds an overhead, specifically in distributed team members, and therefore affects performance in pair programming, in line with the findings in [S52]. In addition, in distributed pairs if one member does not respond or one could not understand partners’ views clearly. It could be resulted delayed in project development because in shared environment, it takes more time to absorb the partner logic, then sometimes it is difficult to complete the programming tasks in time. As a result, it affects other social practices in a bad manner such as lack of coordination, confidence, and so on.

Many studies investigated the benefits of pair programming in terms of performance, quality, productivity, effort, correctness, etc. and results report significant findings for the proposed benefits.

Quality is most commonly studied in several experiments and results show that software developed in pair programming is of high quality. Moreover, quality of software is investigated in various perspectives using pair programming practice, for example, better design and code, fewer or decreased errors/defects, more test cases passed, and more. (see Section 3.2).

In our experiment context, we investigated the effects of DPP on performance in terms of time to complete the task(s). Descriptive statistics show that performance of programmers in distributed pairs is slightly higher than individual programmers. In distributed settings, the quality of code produced by distributed pairs is higher than solos on the basis of defined quality criteria, this confirms the results of [S54], [S68]. Therefore, experimental results reveal that in distributed PP, programmers produce high quality code.

In few studies contradictory effects of pair programming are reported, specifically with respect to quality [S53], [S55], [S60] and effort [S33], [S34], [S66]. The contradictory differences are reported due to particular study types, such as, study arrangements, participants of the study, complexity of developed system or tasks. It is also explained that when a software project is not big enough or there are tasks with varied complexity, then the impact of PP practice may be insignificant. In addition, the lack of development skills affect the results significantly in controlled experiments. Moderating factors, such as expertise, task complexity, training in pair programming, motivation, etc. play an important role for contradictory differences and, therefore, needed to be controlled efficiently as suggested in related work too [27].

In ideal software development teams, team members have formal and informal communication, coordination, social interaction. Distribution alters these properties of an ideal software development team. Therefore, several tools has been developed that enable communication, coordination, social interaction, etc. to support DPP with different features. Mainly two types of tools are reported in the literature: collaboration tools and awareness tools that support ideal software development team properties. Both types of tools are used based on the nature of the programming assignments or software projects. Collaboration tools such as VNC or Microsoft Net meeting, etc. are commonly used for the purpose of educational learning. Most commonly awareness tools, for example Xpairtise plug-in for Eclipse, are used in large software development projects that support social interaction.
6 CONCLUSIONS

In this thesis, pair programming—a key practice of extreme programming—was evaluated in distributed settings. In the first step, the proposed benefits of PP were investigated in both co-located and distributed settings. This was done by conducting a systematic literature review. Second, the PP impacts on performance and on social practices were investigated in distributed settings. This was done by performing a short duration experiment with the help of students.

6.1 Answer to research questions

In this section, results are mapped to the relevant research questions. Each research question along with its brief answer provided by this thesis work is described below.

6.1.1 Research question - 1

What are the benefits of pair programming claimed in the related published literature in both collocated and distributed development?

Sections 3.2.1 and 3.2.2 present the proposed benefits of PP in both collocated and distributed settings respectively, which were obtained through systematic literature review. In these sections, many proposed benefits of PP have been identified and presented. Mostly proposed benefit is quality of the work (design, code, etc.) using PP. Other proposed benefits include effort, productivity, performance, enjoyment of work, confidence and satisfaction etc.

6.1.1.1 Research question – 1.1

How can distributed PP be enabled through computer mediation?

Various tools are designed and evaluated to enable distributed PP through computer mediation. Section 3.3 presents the tools supporting distributed PP in detail.

6.1.2 Research question - 2

What are the possible impacts of pair programming practice on social practices of each individual in distributed development?

Section 4.1.2 presents the results relevant to the social practices (communication, knowledge sharing/transfer, satisfaction and confidence about solutions, and enjoyment of work). In this section, results to each selected social practice have been presented. The results showed that more effective and positive communication was found among the distributed programmers. PP was also found as an effective technique for knowledge sharing/transfer in distributed programmers. Programmers’ satisfaction and confidence about the solution were higher using PP. PP increased the enjoyment of work among distributed programmers.
6.1.3 Research question - 3

What are the effects of pair programming on performance in distributed development?

Average performance of the programmers in PP was slightly higher than SP. 8% in task1 and 10% in task2 i.e. the pair programmers completed the programming tasks in less time than the programmers in SP. However, there was no statistical significance to reject the null hypothesis. The answer is presented in detail in section 4.1.

6.2 Limitations

The main limitations to this study are as follows:

- The systematic literature review focused only on the primary studies of PP in which its benefits are studied. If some other factors such as personality or pair compatibility issues are under investigation then the results of SLR will be different than the results of this review due to the different search strings.
- A short duration experiment in a university setting.
- Participants of the experiment are students.
- Experimental tasks are of small duration and of low level of complexity.
- Sample size for the experiment is small.

It was difficult to conduct experiment for more complex tasks because students had short time to participate in experiment including separate time for training session.

6.3 Future work

In the current scenario a short duration experiment was conducted with the help of 9 students who programmed low level complexity tasks. As a result, small data was collected. To be able to draw statistically significant conclusions experiment has to be repeated on a large scale if possible, for example, for a large project in a software company part of distributed software development operations. This will help to draw conclusions about PP in DSD, and whether distributed programmers should be a standard practice in industry. However, more concentration should be given to “How agile practices, particularly PP, can be implemented successfully in DSD?”
7 REFERENCE


8  APPENDIX

8.1  List of Selected Studies


[S40] S. Xu and V. Rajlich, “Pair programming in graduate software engineering course projects,” 2006, p. FIG.


56


8.2 Pair Programming Guidelines

Please follow the guidelines below:

- Engage your partner in a dialogue
- Listen each other
- Take frequent breaks
- Do not be a sleeping partner
- Make programming practical
- Trust each other
- Avoid confusions, use simple words during the discussions e.g. shared language and vocabulary.
8.3 Participants Data Form

1. Last name: ________________________________
2. First name: ________________________________
3. Student ID: ________________________________
4. Study Program: ________________________________
5. Email: ________________________________
6. Skype ID: ________________________________
7. Contact Number: ________________________________
8. How much industrial experience do you have? _____ Year(s) _____ Months
9. How much java programming experience do you have? ___ Year(s) ____ Months
10. How much pair programming experience do you have? ___ Year(s) ____ Months
11. How much distributed software development experience do you have? ___ Year(s) ____ Months
12. Please list the tools you have used for programming in general, specific to java programming and for distributed software development.

____________________________________________________________________________________________________
____________________________________________________________________________________________________
____________________________________________________________________________________________________
____________________________________________________________________________________________________
____________________________________________________________________________________________________
____________________________________________________________________________________________________
8.4 Experiment Data Collection Form (Pair Programming)

<table>
<thead>
<tr>
<th>Task No:</th>
<th>Partner ID:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your ID:</td>
<td>Start Time:</td>
</tr>
<tr>
<td>Your email address:</td>
<td>End Time:</td>
</tr>
</tbody>
</table>

Treatment: **Pair Programming**

Further Information:

- Time to understand the problem: _____ (minutes)
- Time to find the solution: _____ (minutes)
- Time for solution design: _____ (minutes)
- Time for implementation/coding: _____ (minutes)
- Time for testing: _____ (minutes)

Additional Information (e.g. unnecessary pause, break or disturbance):

Your name: _____________________________
Your student ID: _____________________________
Your email address: _____________________________
8.5 Experiment Data Collection Form (Solo Programming)

| Your name: | ____________________________ |
| Your student ID: | ____________________________ |
| Your email address: | ____________________________ |

Task No: __________ Treatment: Solo Programming

Your ID: __________

Start Time: __________ End Time: __________

Further Information:

Time to understand the problem: ______ (minutes)
Time to find the solution: ______ (minutes)
Time for solution design: ______ (minutes)
Time for implementation/coding: ______ (minutes)
Time for testing: ______ (minutes)

Additional Information (e.g. unnecessary pause, break or disturbance):
8.6 Survey Questionnaire

**Please fill in the questionnaire.**
**Please respond below with 1, 2, 3, 4 or 5; where choices are explained.**

**Communication:**

1. My partner’s active communication allowed me to be more active in expressing my views.

   ![Rating Options]

   Your choice: ______

2. My partner described his/her views clearly and I was able to fully understand him/her.

   ![Rating Options]

   Your choice: ______

**Knowledge sharing/transfer:**

1. Knowledge sharing is higher using pair programming in distributed teams.

   ![Rating Options]

   Your choice: ______
2. I can complete the task alone after developing it in pair programming.

   1. Strongly Agree
   2. Agree
   3. Neither
   4. Disagree
   5. Strongly Disagree

   Your choice: _______

3. Pair programming is a good practice for learning from peers.

   1. Strongly Agree
   2. Agree
   3. Neither
   4. Disagree
   5. Strongly Disagree

   Your choice: _______

Satisfaction and Confidence about solution:

1. I am more confident in completeness and correctness of results of pair programming than solo programming.

   1. Strongly Agree
   2. Agree
   3. Neither
   4. Disagree
   5. Strongly Disagree

   Your choice: _______

2. Quality of the solution is higher in pair programming than in solo programming.

   1. Strongly Agree
   2. Agree
   3. Neither
   4. Disagree
   5. Strongly Disagree

   Your choice: _______

3. I can produce a better solution if I work in pair programming than in solo programming.

   1. Very Confident
   2. Confident
   3. Undecided
   4. Not very Confident
   5. Not at All Confident

   Your choice: _______
4. In pair programming, I believe that I had an easier time reaching the solutions.

1. Strongly Agree
2. Agree
3. Neither
4. Disagree
5. Strongly Disagree

Your choice: _______

5. I am satisfied to work with partner to handle tasks using pair programming.

1. Strongly Agree
2. Agree
3. Neither
4. Disagree
5. Strongly Disagree

Your choice: _______

6. I feel that my overall experience to work on programming tasks in pair programming is better than solo programming.

1. Strongly Agree
2. Agree
3. Neither
4. Disagree
5. Strongly Disagree

Your choice: _______

Enjoyment of work:

1. To work in pair programming is more enjoyable compared to solo programming.

1. Strongly Agree
2. Agree
3. Neither
4. Disagree
5. Strongly Disagree

Your choice: _______

2. I would like to work again using pair programming in my future projects.

1. Very Interested
2. Somewhat Interested
3. Neutral
4. Not Very Interested
5. Not at All Interested

Your choice: _______
Please briefly describe your experiences from the experiment by answering the following statements:

<table>
<thead>
<tr>
<th>Please explain the feelings or experiences about the strengths of pair programming practice during the experiment.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Please explain your experiences about the problems or weaknesses of pair programming practice during the experiment.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Time utilization in pair programming is more productive and yields performance higher than the time in solo programming.</th>
</tr>
</thead>
</table>
8.7 Kappa Analysis

RaterA * RaterB Crosstabulation

<table>
<thead>
<tr>
<th>Count</th>
<th>RaterB</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Experiments</td>
<td>Case study</td>
</tr>
<tr>
<td>RaterA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experiments</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Case study</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Tools</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Survey</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Interviews</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Symmetric Measures

<table>
<thead>
<tr>
<th>Measure of Agreement</th>
<th>Value</th>
<th>Asymp. Std. Error</th>
<th>Approx. (t)</th>
<th>Approx. . Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kappa</td>
<td>0.677</td>
<td>0.120</td>
<td>6.020</td>
<td>0.000</td>
</tr>
<tr>
<td>N of Valid Cases</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. Not assuming the null hypothesis.
b. Using the asymptotic standard error assuming the null hypothesis.

8.8 Answers to the open ended questions

Please explain the feelings or experiences about the strengths of pair programming practice during the experiment.

- Pair programming is good practice and should be adopted to perform modular development in more flexible way. It is good experience to work in interesting setting of experiment with random pairs.
- It was good; it really helps to understand different problems. But sometimes it feels boring as one partner becomes more active and one lenient.
- Pair programming is really good if you do not know some concepts or techniques, but I think there should be a parameter like level of skills, that should be considered when we do pp.
- (1) - Lot of work can be done with lot of fun, (2) - more productive than usual (3) - always good to have a second pair eye on the proposed solution.
- I can share my ideas with my peer; we can reach on solution with speed.
- (1) Working while talking. (2) More ideas regarding design.
- Good way to spend 30 odd minutes on work.
- It helps to enhance solution quality as pairs can review each other’s work and suggest different solution.
- Pair programming is more supportive than solo; a supportive environment provided by pair programming is always good for development.

**Please explain your experiences about the problems or weaknesses of pair programming practice during the experiment.**

- One of main problem I observed is lack of coordination due to expertise level. It happens when we working in shared environment, we take time to absorb the partner logic and sometimes it’s difficult for critical timeline of performing task.
- Good in resolving problems, gain experience, one take control and other waits.
- A weakness of pp when skill level varies substantially is, hindering skilled programmer from solving the problem especially in distributed pp it is more tangible.
- No weakness at all
- If one member is more dominant, he will try to get his solution accepted therefore leading to problems in team.
- Sometimes difference of opinion on solution
- Sometimes conflict arises in different solutions proposed by pair programmers.
- Sometimes it stops the working, the output of partner effect the overall working. Problem with partner can cause the overall development

**Time utilization in pair programming is more productive and yields performance higher than the time in solo programming.**

- Agreed, if programming is done under managed and controlled settings. But again solo programming is quite useful when we have to meet the deadlines if we cannot afford delaying. But if partner is of same level in coding then pair programming is blessing.
- Again it depends on skill level of both partners, if skill level are close to each other, it can help to more productive pairs, but in other situation it will decrease performance
- Time is utilized in proper way and task will be completed
- Yes it is more productive.
I agree. As you do not need to stop work to talk to peers, I can do pair programming its simultaneous.

Yes, absolutely.

Yes. But not always as in case of conflicts, there are time delays.

In solo programming, sometimes things make confusion that affect the working and take more time.

**8.9 Quality assessment criteria for developed tasks**

**8.9.1 Check list for coding standard**

<table>
<thead>
<tr>
<th>Item</th>
<th>Result (Yes/No)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding standard</td>
<td></td>
</tr>
<tr>
<td>Are there variables or attributes with meaningful names?</td>
<td></td>
</tr>
<tr>
<td>Is every variable and attribute properly initialized?</td>
<td></td>
</tr>
<tr>
<td>Are there enough comments in the code?</td>
<td></td>
</tr>
<tr>
<td>Is written code well structured and more readable?</td>
<td></td>
</tr>
</tbody>
</table>

**8.9.2 Test cases for developed tasks**

<table>
<thead>
<tr>
<th>Task1 TC-01</th>
<th>Purpose: To verify that input fields for Mean and Standard Deviation does not accept alphabets as input</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#.</td>
<td>Steps</td>
<td>Expected Result</td>
</tr>
<tr>
<td>1</td>
<td>Get Input of value A in alphabets</td>
<td>Value of A should be entered</td>
</tr>
<tr>
<td>2</td>
<td>Get input value of B in alphabets</td>
<td>Value of B should be entered</td>
</tr>
<tr>
<td>3</td>
<td>Apply Mean formula/Standard Deviation formula</td>
<td>An information message should appear that alphabets are not allowed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task1 TC-02</th>
<th>Purpose: To verify that formula for Mean work correctly when input is provided</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#.</td>
<td>Steps</td>
<td>Expected Result</td>
</tr>
<tr>
<td>1</td>
<td>Get Input of value A</td>
<td>Value of A should be entered</td>
</tr>
<tr>
<td>2</td>
<td>Get input value of B</td>
<td>Value of B should be entered</td>
</tr>
<tr>
<td>3</td>
<td>Apply Mean formula</td>
<td>Result should be generated according to the Mean formula</td>
</tr>
</tbody>
</table>

---

13 [This is a general checklist developed specific to our experiment. There can be a more detailed checklist for manual code analysis.](#)
### Task1 TC-03
**Purpose:** To verify that formula for standard deviation work correctly when all required real numbers are provided

<table>
<thead>
<tr>
<th>#.</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Results / Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enter input values</td>
<td>Input should be entered in the fields</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Apply Standard Deviation formula</td>
<td>Result should be calculated correctly</td>
<td></td>
</tr>
</tbody>
</table>

### Task2 TC-01
**Purpose:** To verify that program return zero when there is no line

<table>
<thead>
<tr>
<th>#.</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Results / Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>Execute program which don’t have any line of code but has comments lines</td>
<td>Program should return zero</td>
<td></td>
</tr>
</tbody>
</table>

### Task2 TC-02
**Purpose:** To verify that program return 50 when user execute a program which has 50 lines of code except comments

<table>
<thead>
<tr>
<th>#.</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Results / Observations</th>
</tr>
</thead>
</table>
| 1)  | Execute program with a source code file which has 50 lines | ➢ Program run successfully  
➢ Program return 50 |                               |

### Task2 TC-03
**Purpose:** To verify that program return correct number of logical lines when every second line has comments

<table>
<thead>
<tr>
<th>#.</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Results / Observations</th>
</tr>
</thead>
</table>
| 1)  | Execute program where every second line has 50 lines | ➢ Program run successfully  
➢ Program return 50 |                               |

### Task2 TC-04
**Purpose:** To verify that program does not count lines which has textual statements but those are not comments

<table>
<thead>
<tr>
<th>#.</th>
<th>Steps</th>
<th>Expected Result</th>
<th>Actual Results / Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>Add textual lines in the program</td>
<td>Lines should be added</td>
<td></td>
</tr>
</tbody>
</table>
| 2)  | Execute program              | ➢ Error message should appear about textual lines  
➢ Textual lines should be in count |                               |
### 8.10 Our SLR studies overlap with related SLRs

<table>
<thead>
<tr>
<th>Sr.No.</th>
<th>Our SLR</th>
<th>Jalali &amp; Wohlin [6]</th>
<th>Dyba, Arisholm et al. [22] and Hannay et al. [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[S42]</td>
<td>[S7]</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[S68]</td>
<td>[S9]</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[S51]</td>
<td>[S55]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[S27]</td>
<td></td>
<td>[1]</td>
</tr>
<tr>
<td>5</td>
<td>[S56]</td>
<td></td>
<td>[2]</td>
</tr>
<tr>
<td>6</td>
<td>[S66]</td>
<td></td>
<td>[3]</td>
</tr>
<tr>
<td>7</td>
<td>[S46]</td>
<td></td>
<td>[4]</td>
</tr>
<tr>
<td>8</td>
<td>[S54]</td>
<td></td>
<td>[5]</td>
</tr>
<tr>
<td>9</td>
<td>[S50]</td>
<td></td>
<td>[6]</td>
</tr>
<tr>
<td>10</td>
<td>[S60]</td>
<td></td>
<td>[7]</td>
</tr>
<tr>
<td>11</td>
<td>[S55]</td>
<td></td>
<td>[8]</td>
</tr>
<tr>
<td>12</td>
<td>[S65]</td>
<td></td>
<td>[9]</td>
</tr>
<tr>
<td>13</td>
<td>[S67]</td>
<td></td>
<td>[10]</td>
</tr>
<tr>
<td>14</td>
<td>[S1]</td>
<td></td>
<td>[12]</td>
</tr>
<tr>
<td>15</td>
<td>[S9]</td>
<td></td>
<td>[13]</td>
</tr>
<tr>
<td>16</td>
<td>[S26]</td>
<td></td>
<td>[14]</td>
</tr>
<tr>
<td>17</td>
<td>[S33]</td>
<td></td>
<td>[16]</td>
</tr>
<tr>
<td>18</td>
<td>[S37]</td>
<td></td>
<td>[17]</td>
</tr>
<tr>
<td>19</td>
<td>[S28]</td>
<td></td>
<td>[18]</td>
</tr>
</tbody>
</table>

### 8.11 Electronic data sources with search string

<table>
<thead>
<tr>
<th>Data-Sources</th>
<th>Search String</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM</td>
<td>(&quot;pair programming&quot;) AND (experiment* OR evaluation OR measurement OR assessment OR investigation OR validation) AND (benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR &quot;social practice*&quot;)</td>
</tr>
<tr>
<td>IEEE</td>
<td>(&quot;pair programming&quot;) AND (experiment* OR evaluation OR measurement OR assessment OR investigation OR validation) AND (benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR &quot;social practice&quot;)</td>
</tr>
<tr>
<td>EiVillage (Inspec and Compendex)</td>
<td>((((&quot;pair programming&quot;) ) WN All fields) AND (((experiment* OR evaluation OR measurement OR assessment OR investigation OR validation)) WN All fields)) AND (((benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR &quot;social practice&quot;)) WN All fields)), English only, 1998-2010</td>
</tr>
<tr>
<td>Database</td>
<td>Query</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Science Direct</td>
<td>pub-date &gt; 1997 and pub-date &lt; 2011 and (&quot;pair programming&quot;) AND (experiment* OR evaluation OR measurement OR assessment OR investigation OR validation) AND (benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR &quot;social practice&quot;*)</td>
</tr>
<tr>
<td>ISI Web of Science</td>
<td>Topic=(&quot;pair programming&quot;) AND (experiment* OR evaluation OR measurement OR assessment OR investigation OR validation) AND (benefit* OR evolution OR efficiency OR impact* OR performance OR productivity OR social OR factor* OR behavioral OR &quot;social practice&quot;*) Timespan=1998-2010. Databases=SCI-EXPANDED, SSCI, A&amp;HCI, CPCI-S, CPCI-SSH.</td>
</tr>
</tbody>
</table>