Salleh, N., Mendes, E., & Grundy, J. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: a systematic literature review.

Originally published in *IEEE Transactions on Software Engineering*, *37*(4), 509–525.
Available from: http://dx.doi.org/10.1109/TSE.2010.59

Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education: A Systematic Literature Review

Norsaremah SALLEH, Emilia MENDES, and John GRUNDY, Member, IEEE

Abstract— OBJECTIVE – The objective of this paper is to present the current evidence relative to the effectiveness of pair programming (PP) as a pedagogical tool in higher education CS/SE courses. METHOD – We performed a systematic literature review (SLR) of empirical studies that investigated factors affecting the effectiveness of PP for CS/SE students and studies that measured the effectiveness of PP for CS/SE students. RESULTS – Seventy four (74) papers were used in our synthesis of evidence, and 14 compatibility factors that can potentially affect PP's effectiveness as a pedagogical tool were identified. Results showed that students' skill level was the factor that affected PP's effectiveness the most. The most common measure used to gauge PP's effectiveness was time spent on programming. In addition, students' satisfaction when using PP was overall higher than when working solo. Our meta-analyses showed that PP was effective in improving students' grades on assignments. Finally, in the studies that used quality as a measure of effectiveness, the number of test cases succeeded, academic performance, and expert opinion were the quality measures mostly applied. CONCLUSIONS – The results of this SLR show two clear gaps in this research field: i) lack of studies focusing on pair compatibility factors aimed at making PP an effective pedagogical tool; ii) a lack of studies investigating PP for software design/modeling tasks in conjunction with programming tasks.

Index Terms— empirical studies, pair programming, systematic review.

1 Introduction

PAIR programming (PP) involves two people sitting side by side, using only one computer and working collaboratively on the same design, algorithm, code or test [4]. One is the "driver", who is responsible for designing, typing the code, and having control over the shared resource (e.g. computer, mouse, keyboard). The second is the "navigator" or "observer", who has responsibility for observing how the driver works in order to detect errors and offer ideas in solving a problem. Throughout their work, pairs typically alternate their roles after a certain duration [70].

PP's popularity has drawn the attention of many researchers, thus causing an increase in the number of studies conducted in both industrial as well as in educational contexts [1]. A survey of organizations from a software process improvement user group showed that 72% of the organizations, from a variety of industries, have implemented the PP practice [41]. Some studies have investigated PP's usefulness and effectiveness as a Computer Science/Software Engineering (CS/SE) pedagogical tool, e.g. [46],[47],[53],[18],[59], some with promising results.

Early research on the use of PP as a pedagogical tool focused mainly on its ability to benefit students in terms of productivity and quality of work produced [69]. For example, evidence suggests that PP could enhance enjoyment [46],[47],[69],[73]; increase students' confidence level [5],[46],[28]; reduce workload [9]; improve course completion rate [46],[53]; increase homework submission rate [28]; improve exam's performance [47],[49],[53]; and facilitate working more efficiently on programming tasks [11],[67].

In 2000, Cockburn and Williams [12] investigated the cost and benefit of PP based on empirical evidence [71],[68],[55]. They conclude that, with an increase of only 15% in the cost of development time, PP offers significant benefits such as improving design quality (fewer defects), team communication and rapid solutions to problems, enhancing the learning process, and increasing the enjoyment to learn. They suggest that PP is a promising approach to use as a pedagogical tool due to its capability to increase learning capacity [12].

Dybå et al. [19] conducted a systematic literature review (SLR) investigating whether existing empirical evidence supports the claims that PP is more advantageous than solo programming. They reviewed 15 studies comparing solo and pair programming, and involving both students and software practitioners as subjects. The general aspects investigated were related to PP's effectiveness, including "duration" (time spent to produce the system), "effort" (person-hours spent), and "quality of the final product". Their meta-analysis suggests that PP is

• *N. Salleh is with Department of Computer Science, International Islamic University Malaysia, P.O. Box 10, 50728,Kuala Lumpur, Malaysia.*
• *Salleh and E. Mendes are with Computer Science Department, University of Auckland, Private Bag 92019, Auckland, New Zealand. (E-mail: nsal017@ec.auckland.ac.nz, emilia@cs.auckland.ac.nz)*
• *J. Grundy is with the Faculty of Information & Communication Technologies, Swinburne University of Technology, PO Box 218, Hawthorn, Victoria, Australia (E-mail: jgrundy@swin.edu.au)*

Manuscript submitted 29 October 2008, revised 13 July 2009, 23 December 2009, 16 March 2010. Accepted 30th April 2010.

more effective than solo programming when quality and the duration to complete the tasks are the concern, but PP overall requires more effort (i.e. more person-hours). However, it is likely that participants' expertise and task complexity might have affected the accuracy of their findings. This SLR is related to ours, but is different in terms of its purpose and population. Our SLR investigated the potential of PP as a pedagogical tool, specifically focusing on existing evidence regarding PP's effectiveness in the context of higher education institutions.

Also in 2007, Dybå and Dingsøyr [20] carried out a SLR of Agile Software Development empirical studies examining benefits, limitations, and strength of evidence for agile methods. PP was not the focus of this SLR that found a low strength of evidence supporting agile techniques.

As PP inherently involves a social interaction between two people, investigating compatibility aspects is, in our view, very important. Previous studies reported that students who experienced PP with an incompatible partner disliked the collaborative work [38],[64]. For example, Muller et al. [51] show that the performance of a pair is correlated with how comfortable the pairs feel during a pair session ("feel-good" factor). Since students' performance may be largely affected by the pair's compatibility, it seems relevant and applicable to examine compatibility factors of paired students and its effect on learning. Our goal is not only to contribute to the body of knowledge of PP but also to improve the use of PP as an effective pedagogical tool.

In order to realize how PP can significantly contribute as an effective pedagogical tool, a proper investigation of its implementation needs to be carried out. Chaparro et al. [9] suggest that the potential to effectively use PP is *highly connected with the compatibility factors relative to the paired subjects*. Thus, one important aspect is to understand the underlying factors that contribute to a successful pairing formation i.e. factors that make pairs highly compatible. Our research aims to improve the practice of PP as a pedagogical tool in CS/SE education by investigating pair compatibility and its effect on PP's effectiveness. We applied a *systematic literature review* (see Section 2) in assessing existing PP literature. The key contribution of this paper is the findings from our SLR of empirical studies of PP in higher education settings.

TABLE 1
SUMMARY OF PICOC

Population	CS/SE students in higher education
Intervention	Pair programming
Comparison	None
Outcomes	PP's effectiveness
Context	Review(s) of any empirical studies of pair programming within the domain of CS/SE in higher education. No restrictions on the type of empirical study (e.g. case study) apply.

We present our SLR results by integrating evidence into patterns that can be used to understand the current *state-of-the-art* of research in PP when applied to a higher education context. We believe this can better inform educators wanting to incorporate PP into a CS/SE curriculum. Additionally, conflicting findings from the analysis are presented and gaps in the existing body of knowledge are highlighted. These suggest key areas of focus for future PP research. Section 2 describes the method we used in our SLR. Section 3 reports the results of our SLR based on the synthesis of evidence. Section 4 presents a discussion of our key findings, implications, threats to the validity of this review, and future work. Section 5 presents conclusions from the review.

2 THE REVIEW METHOD

Introduction

A SLR is defined as a process of identifying, assessing, and interpreting all available research evidence with the purpose to provide answers for specific research questions [35]. It is a tool that aims to produce a scientific summary of the evidence in a particular area, in contrast to "traditional" narrative review [56]. We followed the procedures of Kitchenham and Charters [35].

Research Questions

Table 1 shows the PICOC (*Population*, *Intervention*, *Comparison*, *Outcomes*, and *Context*) structure of our research questions. In our SLR, we included all empirical studies that investigated PP within a higher education setting, regardless of whether or not they compare PP to solo students. Therefore, we could not include a specific comparison in our PICOC.

The primary focus of our SLR was to understand and identify the factors that influence the effectiveness of the PP practice for CS/SE in higher education. While the primary reason for using PP in industry is to gain benefits in terms of economic advantage (i.e. time to market, development effort, quality etc.) [12], [19], the type of outcomes that can benefit students' learning is what motivates educators [46]. We organized the measurement of PP's effectiveness into four broad categories: academic performance, technical productivity, program/design quality, and satisfaction [46]. Therefore, our SLR aimed to answer the following primary research question (RQ):

Primary Question: What evidence is there of PP studies conducted in higher education settings that investigated PP's effectiveness and/or pair compatibility for CS/SE education?

Our SLR also aimed to answer the following secondary sub-questions:

Sub-Question 1: What evidence is there regarding compatibility factors that affect pair compatibility and/or PP's effectiveness as a CS/SE pedagogical tool and which pairing configurations are considered as most effective?

Sub-Question 2: How was PP's effectiveness measured in PP studies and how effective has PP been when used within higher education settings?

Sub-Question 3: How was quality measured in the PP studies that used software quality as a measure of effectiveness[1]?

[1] The choice to focus on quality was due to the fact that most studies we already knew about measured PP's effectiveness using quality metrics

Identification of Relevant Literature

The strategy we used to construct the search strings was as follows [35], [48]:

- Derive major terms used in the review questions (i.e. based on the population, intervention, outcome, and context);
- List the keywords mentioned in the articles (primary studies) we already knew about;
- Search for synonyms and alternative words. We have also consulted a subject librarian to seek further advice in the proper use of the terms;
- Use the Boolean OR to incorporate alternative spellings and synonyms;
- Use the Boolean AND to link the major terms from population, intervention, and outcome.

The complete search string initially used for the searching of the literature was as follows:

(student OR undergraduate) AND (pair programming OR pair-programming) AND (experiment OR measurement OR evaluation OR assessment) AND (effective OR efficient OR successful)

Petticrew and Robert [56] highlight that the two major issues in conducting a SLR search are the sensitivity and specificity of the search. The sensitivity refers to a search that retrieves a high number of relevant studies. Specificity causes the search to retrieve a minimum number of irrelevant studies. In our preliminary search, we retrieved a very small number of articles when using the complete search string defined above. For instance, IEEEXplore, Inspec, and ProQuest each retrieved only five, three, and four articles respectively. Therefore, we sought the opinion of a subject librarian regarding the appropriate use of our search string and her advice was that we should use a much simpler string than the one defined in the protocol to enable the retrieval of more results. We used the keywords "pair programming" OR "pair-programming" which resulted in a higher number of studies retrieved from various online databases. The primary search process involved the use of 12 online databases: ACM Digital library, Current Contents, EBSCOhost, IEEEXplore, ISI Web of Science, INSPEC, ISI Proceedings, ProQuest, Sage Full Text Collections, ScienceDirect, SpringerLink, and Scopus. The selection of online databases was based on our knowledge of databases that index PP primary studies we were aware of and the list of available online databases subscribed by the University of Auckland's library under the "Computer Science" subject category. Khan et al. [34] recommend searching multiple databases to cater for as many citations as possible to avoid bias to the review. Thus, we also searched from the *citeseer* website using similar keywords (i.e. "pair programming" OR "pair-programming"); from the *Agile alliance* website we looked for articles under two categories: "pair programming" and "Extreme programming"; and on-line *Google scholar* was used to search for full text of articles. Our experience in literature search supports the suggestion by Kitchenham et al. [35] that it is important for SE researchers to identify a list of relevant online databases to facilitate the search process.

Upon completion of the primary search phase, the identification of relevant literature continued with the secondary search phase. During this search phase, all the references in the papers identified from the primary sources were reviewed. If a paper was found to be suitable, it was added to the existing list of studies qualified for the synthesis.

Selection of Studies

Our inclusion criteria aimed to only include PP empirical studies that targeted CS/SE education and that used PP as a practice defined by the XP creators in 1999 [4]. As such, the literature search only covered studies published within the period of 1999 to 2007. The detailed inclusion criteria comprised (i) studies that investigated factors affecting the effectiveness of PP for CS/SE students; and (ii) studies that measured the effectiveness of PP for CS/SE students.

The main exclusion criterion comprised PP papers not targeted at CS/SE education. In addition the following criteria were also applied: (i) papers presenting claims by the author(s) with no supporting evidence; (ii) papers about Agile/XP describing development practices other than PP, such as test-first programming, refactoring etc; (iii) papers that only described tools (i.e. software or hardware) that could support PP; (iv) papers involving students but outside higher education; (v) papers that solely investigated distributed PP; (vi) papers not written in English.

Data Extraction and Study Quality Assessment

To facilitate the data extraction process a form was designed[2], used to gather evidence relating to our research questions and to measure the quality of the primary studies. When designing the studies' quality checklist we reused some of the questions proposed in the literature [39], [56], [62], [17], [24], [27]. Our checklist comprised seven general questions (see Table 2) to measure the quality of both quantitative and qualitative studies according to the following ratio scale: Yes = 1 point; No = 0 points; Partially = 0.5 point. The resulting total quality score for each study ranged between 0 (very poor) and 7 (very good).

One of the authors (Salleh) was responsible for reading and completing the extraction form for each of the primary studies. In order to validate the data extraction process, a random sample comprising 20% of the total number of primary studies had their data extracted by the first and second authors and then compared in a review meeting. Whenever the data extracted differed, where differences never surpassed more than 10-15%, such differences were discussed until consensus was reached. We did not measure inter-rater agreement since our review aimed to reach an absolute consensus on the sample used [36]. For the remaining 80% primary studies we hoped that lessons learnt from the review meeting would minimize the bias with their data extraction. If information in a study was unclear, we contacted author(s) for clarification.

[2] The data extraction form is available at http://www.cs.auckland.ac.nz/~norsaremah/Form.pdf

TABLE 2
STUDY QUALITY CHECKLIST

Item	Answer
1. Was the article refereed? [39]	Yes/No
2. Were the aim(s) of the study clearly stated? [17], [24]	Yes/No/ Partially
3. Were the study participants or observational units adequately described? For example, students' programming experience, year of study etc. [56], [27]	Yes/No/ Partially
4. Were the data collections carried out very well? For example, discussion of procedures used for collection, and how the study setting may have influenced the data collected [56], [62], [24], [27]	Yes/ No/ Partially
5. Were potential confounders adequately controlled for in the analysis? [24]	Yes/No/ Partially
6. Were the approach to and formulation of the analysis well conveyed? For example, description of the form of the original data, rationale for choice of method/tool/package [62], [24], [27]	Yes/No/ Partially
7. Were the findings credible? For example, the study was methodologically explained so that we can trust the findings; findings/conclusions are resonant with other knowledge and experience [62], [56], [27]	Yes/No/ Partially

2 RESULTS

Introduction

In this section we present the synthesis of evidence of our SLR beginning with the analysis from the literature search results. During the selection process, the Scopus database was chosen as the baseline database due to its reputation as the largest abstract and citation database [21]. In addition each article retrieved from the other databases was compared with the existing list of papers accumulated from Scopus' screening process in order to avoid duplication. The initial phase of our search process identified 379 empirical studies using the *"pair programming OR pair-programming"* search term. Of these only 153 were potentially relevant based on the screening of titles and abstracts. Each of these studies was filtered according to the inclusion and exclusion criteria before being accepted for the synthesis of evidence. If titles and abstracts were not sufficient to identify the relevance of a paper, full articles were used. We also carefully checked if there were any duplicate studies if very similar studies were published in more than one paper. Inclusion of duplicate studies would inevitably bias the result of the synthesis [34].

Based on the primary searches, 73 studies (48% of 153 studies) were accepted for the synthesis of evidence after a detailed assessment of abstracts and full text, and exclusion of duplicates (see Fig. 1). The secondary search phase further identified another five studies; however, after their detailed assessment, only one was found relevant for the SLR. Therefore, in total, 74 studies were included for the synthesis of evidence (see Appendix A for the list of included studies). Based on the research classification by Wohlin et al. [74] and Creswell [16], an analysis of the type of research approach used in these studies is shown in Fig 2. Formal experiments were found to be the most popular research approach used (59%).

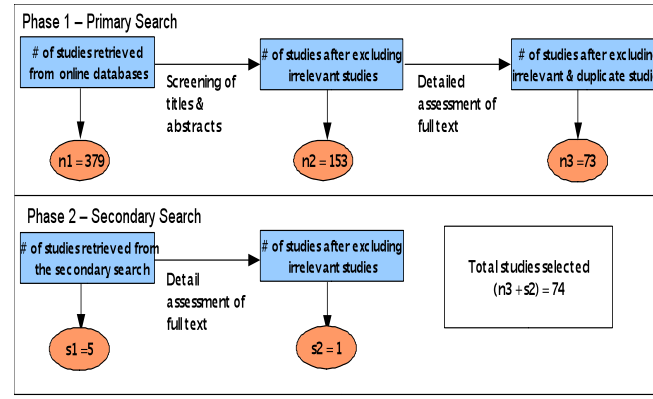

Fig. 1. Identifying Relevant Literature

Table 3 shows the quality scores for all primary studies. Most achieved above average quality: 20 studies (27%) and 36 studies (49%) were deemed very good and good quality respectively. One study attained very poor quality; it did not detail its research methodology and we could not ensure its results were reliable and useful as evidence. This study was removed from the analysis phase. Thus, in the end only 73 studies were included in the analysis of evidence.

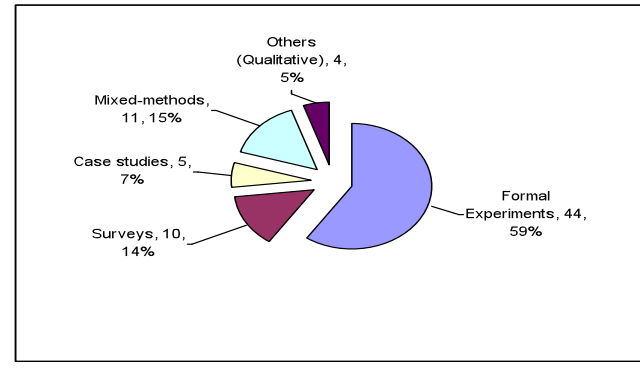

Fig 2. Studies by Research approach

TABLE 3:
QUALITY SCORES

Quality Scale	Very Poor (<2)	Poor (2 – < 3)	Fair (3 – <5)	Good (5 – <= 6)	Very Good (>6)	Total
Number of studies	1	0	17	36	20	74
Percentage (%)	1%	0%	23%	49%	27%	100%

In the following section we present the results for the SLR's main research question and three sub-questions. Each study is identified as S*m*, where *m* represents the study's number (see Appendix A for the list of studies used in this SLR).

Research Question

Question: "What evidence is there of PP studies conducted in higher education settings that investigated PP's effectiveness and/or pair compatibility for CS/SE education?"

The SLR identified 73 PP studies conducted in higher education settings that investigated the use of PP by undergraduate and graduate CS/SE students. The context of

investigation varied via the comparison of PP to other practices, such as solo programming, side-by-side programming, peer-review inspection and application of the practice to design tasks (i.e. pair designing). Studies investigating PP's effectiveness also covered other aspects of PP such as pair formation.

The SLR's ultimate goal was to understand how PP affects students' learning outcomes in order to improve academic achievement, technical productivity, program quality, and learning satisfaction. Of the 73 studies analysed, 17 (23%) investigated factors believed to have a bearing on pair compatibility and PP's effectiveness. Seventy (96%) investigated PP's effectiveness using a quantitative or qualitative approach and 32 (44%) investigated quality aspects as a measure of PP's effectiveness. The following sub-questions detail the SLR's synthesis of evidence.

Sub-Question 1 – Compatibility factors

"What evidence is there regarding compatibility factors that affect pair compatibility and/or PP's effectiveness as a CS/SE pedagogical tool, and which pairing configurations are considered as most effective?"

Compatibility factors are factors believed to influence the affinity of students when working in pairs. Altogether 14 factors were identified by a total of 17 studies which investigated how these factors affected or correlated with PP's effectiveness and/or pair compatibility. Table 4 lists the compatibility factors, studies that looked into each factor and whether a factor had a positive, negative, no effect, or mixed effect. The summary of findings used to answer this research question is available at:

www.cs.auckland.ac.nz/~norsaremah/summary.pdf

Table 4 shows that personality type and actual skill level were the two factors most commonly investigated in PP studies. In terms of personality, the two studies with positive findings reported that paired students of different personality types performed better when compared with paired students of similar type [S50],[S73]. While most studies that investigated the effects of personality type did not produce significant findings there was agreement that paired students of different personalities tended to perform better than pairs of similar personalities [S28],[S32],[S63].

Six out of the nine PP studies that investigated personality type employed the Myers-Briggs Type Indicator (MBTI) [52] as a personality assessment method [S13],[S28],[S29],[S32],[S63], [S73]. Only one study applied NEO-PI [15] to investigate the relationship between programmers' personality and PP's effectiveness [S23]. The study found that the personality of an individual programmer does not have a significant effect on PP's effectiveness, but this may not be the case when looking at the combination of personalities in a single pair. Other than MBTI and NEO-PI, the Keirsey Temperament Sorter [33] and Revised Eysenck Personality Questionnaire (EPQ-R) were used in two studies to measure the personality [S50] and temperament types [S74] of pair developers. Sfetsos et al [S50] report that pairs of mixed-personalities and temperaments achieve better scores than pairs of similar personality. On the contrary, Gevaert [S74] found no significant correlation between personality type and PP's effectiveness.

Seven out of the ten PP studies regarded paired skill level as one of the determinant factors of PP's effectiveness [S8],[S11],[S15],[S28],[S29],[S58],[S63]. The two categories of skill level used were actual and perceived skill. The actual skill level was determined based on programming experience, academic background, and students' academic performance. Perceived skill level was measured subjectively according to the skill of a student's partner relative to their own perceived skill (i.e. "better", "about the same", or "weaker"). The consensus from these studies is that PP works best when the pair has a similar skill level. However, two correlation studies show contradictory findings on the association between students' skill level and PP's effectiveness [S42],[S68]. Muller and Padberg [S42] report there is no correlation between the two variables and Madeyski [S68] refutes this finding.

TABLE 4:
LIST OF FACTORS INVESTIGATED IN PP STUDIES

No.	Factor	Total studies	Significant positive effect	Significant negative effect	No significant effect	Mixed effect
1	Personality	9	S50, S73	-	S13, S23, S29 S32 S74	S28, S63
2	Actual skill level	10	S8, S11, S15, S28, S29, S58, S63, S68, S74	*S11, *S58	S42	-
3	Perceived skill level	1	S14, S28, S29, S63	-	-	-
4	Self-esteem	3	-	-	S8, S29, S63	-
5	Gender	2	S29	-	S73	-
6	Ethnicity	1	S29	-	-	-
7	Learning style	2	-	-	S32, S63	-
8	Work ethic	2	S63	-	S32	-
9	Time management ability	2	-	-	S63	-
10	"Feelgood" factor	2	S42, S68	-	-	-
11	Confidence Level	2	S22, S54	-	-	-
12	Type of role	1	-	-	S14	-
13	Type of tasks	1	S14	-	-	-
14	Communication skills	1	-	-	S73	

*S11 and S58 both reported that skill levels had both positive and negative effect on the PP's effectiveness. Pairs consisting of similar skills can benefit students; pairs consisting of very different skill levels seem ineffective.

The two studies that investigated the effect of gender differences on pair compatibility produced contradictory findings [S29],[S73]: Choi [S73] reports that gender is not a significant factor to influence pair compatibility

whereas Katira et al. [S29] found gender is a factor likely to determine pair compatibility. S29's findings suggest that pairing students of different gender would lead to incompatible pairs and that pairing female students would very likely result in a compatible pair. Three studies that investigated the effect of self-esteem discovered that paired students' self-esteem did not influence pair compatibility [S28],[S29],[S63].

Katira et al. [S29] investigated ethnicity as a compatibility factor by classifying students as either belonging to a majority or minority ethnical group. Their results show that students from minority ethnical groups are more likely to pair with students who are also from minority ethnical groups, but not necessarily the same group. In this study, the effects on pair compatibility when pairing students belonging to the same ethnicity group were not investigated. The study does not report the results of pair compatibility on male students and majority ethnic groups due to its focus on the issue of low representation of minority and female students in CS.

The two studies that investigated the effect of Felder-Silverman learning style reported that learning style did not significantly affect pair compatibility or the perception of students towards pairing [S32],[S63]**.** In terms of work ethic, Williams et al. [S63] report that pairing students of similar work ethic enhances pair compatibility, and Layman [S32] reports that students' perception towards pairing is not affected by their work ethic. Williams et al. [S63] also investigated students' time management ability and found it has no effect on pair compatibility.

In 2004, Muller and Padberg [S42] coined the term "feel-good" which refers to how comfortable pairs feel during the PP session. They report that the feel-good factor is correlated with a pair's performance. Madeyski [S68] had similar findings where a positive correlation between the feel-good factor and pair performance (quality of software) was found.

Very few PP studies have investigated confidence, communication level, type of role and tasks. Thomas et al. [S54] report that performance increased when pairing students of similar confidence. Nevertheless, students who consider themselves as "code warriors" (i.e. high confidence level students) prefer to work alone and enjoy PP less. This contradicts the findings reported by Hanks [S22]. Chapparo et al. [S14] show task type significantly affects PP's perceived effectiveness: paired students prefer program comprehension, re-factoring, and coding to debugging tasks. Choi [S73] reports communication skills have no impact on pair compatibility.

A two-phased study conducted between 2002 and 2005, investigated factors believed to influence pair compatibility. Table 1 (see Appendix B) summarizes the findings. In these studies [S28],[S29],[S63], experiments involved undergraduate and graduate CS students in three courses: Introduction to Programming (CS1), Sw. Eng. (SE), and OO Languages and Systems. Our analysis showed some divergence in the findings. For instance, results were contradictory between CS1 and SE courses when pairing students according to different personality types, similar actual skill level, and self-esteem [S63]. The perceived skill level was the most influencing factor in determining pair compatibility. These studies however did not provide evidence stressing pair compatibility as an important criterion determining PP's effectiveness.

The second part of sub-question 1 investigated the most effective ways of pairing formation from the viewpoint of pair compatibility or pair effectiveness. We presented our evidence based on the ranking of the number of studies with corroborating findings relating to pairing formation (see Table 2 in Appendix B). The *actual skill level* was ranked highest (seven studies) [S8],[S11],[S15],[S28],[S29],[S58],[S63] followed by *perceived skill level* (four studies) [S14],[S28],[S29],[S63]: the skill level between the partners should be similar in order to achieve greater pair compatibility or pair effectiveness. Next was *personality type* where two studies report that students should be paired with a partner of different personality [S50], [S73].

In terms of quality assessment, the average quality score obtained for the studies used to answer sub-question 1 was 5.1, with the highest quality score being 6.5. Of 17 studies, we rated 12 as having good quality of experimental design and analysis.

Sub-Question 2 – Measure of Effectiveness

"How was PP's effectiveness measured in PP studies and how effective has PP been when used within higher education settings?"

PP's effectiveness was measured using various factors, organized in four categories: technical productivity, program/design quality, academic performance, and satisfaction. Technical productivity, measured by 31 (44%) of the 70 studies was the most common method used to assess PP's effectiveness, followed by program/design quality (30 studies, 43%). A subset of 16 studies (23%) evaluated PP's effectiveness based on students' academic performance in final exams, mid-terms, assignments, projects, and course grades. Besides the objective measurements, PP's effectiveness was evaluated subjectively in 22 studies (31%) using students' perceived satisfaction experiencing PP sessions (see Table 3 in Appendix B).

Of the 31 'technical productivity' studies, 19 [S4],[S7],[S9],[S19],[S25],[S30],[S31],[S33], [S38],[S42],[S44],[S46],[S47],[S49],[S51],[S52],[S53],[S60],[S65] used "time spent" as a measure of PP's effectiveness. Of these, 11 studies [S4],[S7], [S9], [S30], [S31], [S33], [S49], [S51], [S52], [S53], [S60] report that paired students complete tasks in shorter duration than solo students. However, 7 studies report that PP incurs additional cost or requires more effort (in person hours) because it takes two on a task [S25],[S65],[S60],[S52],[S53],[S46],[S47]. Some studies do not report the total effort as they included only the time taken to solve the task.

PP studies that measured PP's effectiveness using quality attributes (30 studies) focused on either internal or external code quality; lines of code and the number of test cases passed were the two methods most commonly employed. PP's effectiveness was investigated subjectively by means of students' perception of their satisfaction using PP. Findings showed a positive attitude towards

working collaboratively with another student. Scheduling conflicts and incompatible partners were the major problems highlighted [S24], [S32],[S66],[S69],[S56],[S16].

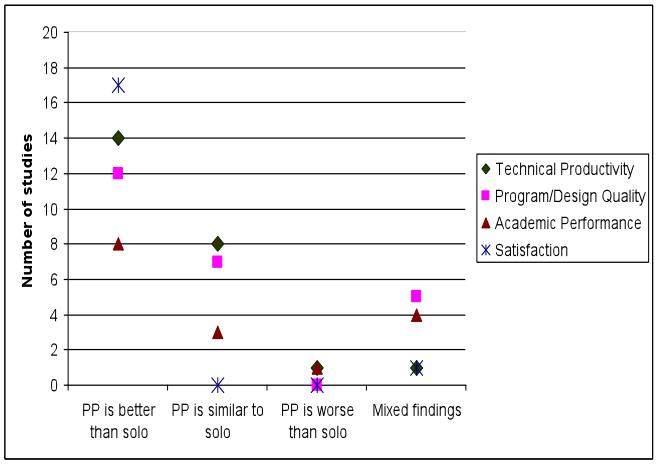

Fig. 3. Studies' findings on PP effectiveness

Of the 45 studies that compared PP to solo programming, 31 report that PP led to an improved performance in technical productivity and satisfaction. Although the findings regarding PP's effectiveness in program/design quality and academic performance varied considerably (see Table 4 in Appendix B), the majority of studies (8 out of 16) report a significant positive effect of PP towards academic performance.

Fig. 3 suggests that PP was a more effective technique compared with solo programming. In terms of satisfaction almost all studies reported similar findings with students presenting greater satisfaction and enjoyment when using PP. Pickard et al. [57] suggest that only studies that have comparable quantitative measures are eligible to be included in a meta-analysis. We found that PP's effectiveness was measured using various types of metrics. Thus in order to perform a meta-analysis we would need to select a specific subset (e.g. final exam score, success rate of paired and solo students) to measure the effects on academic performance.

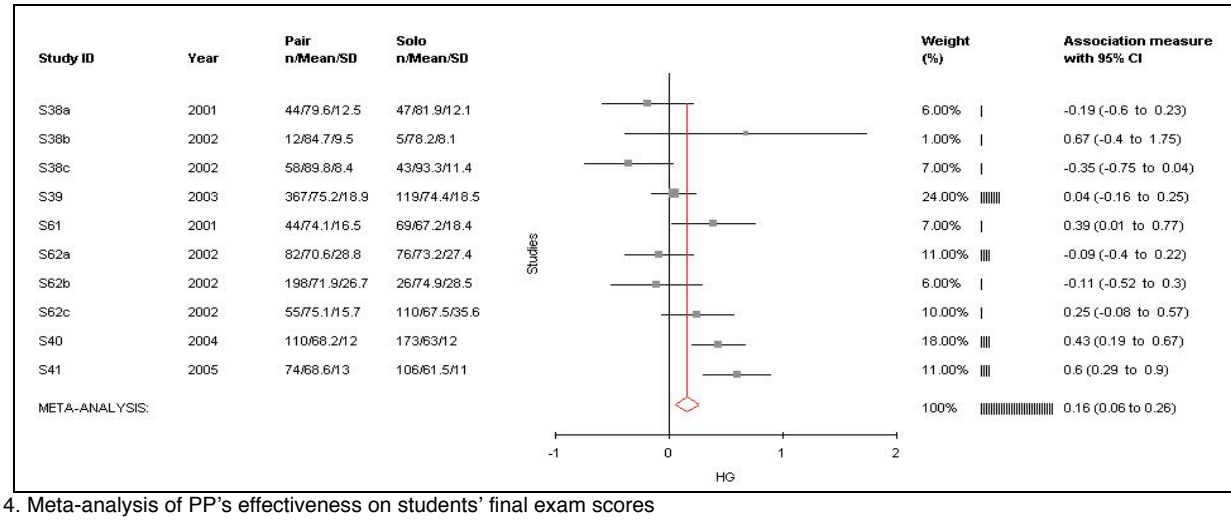

Fig. 4. Meta-analysis of PP's effectiveness on students' final exam scores

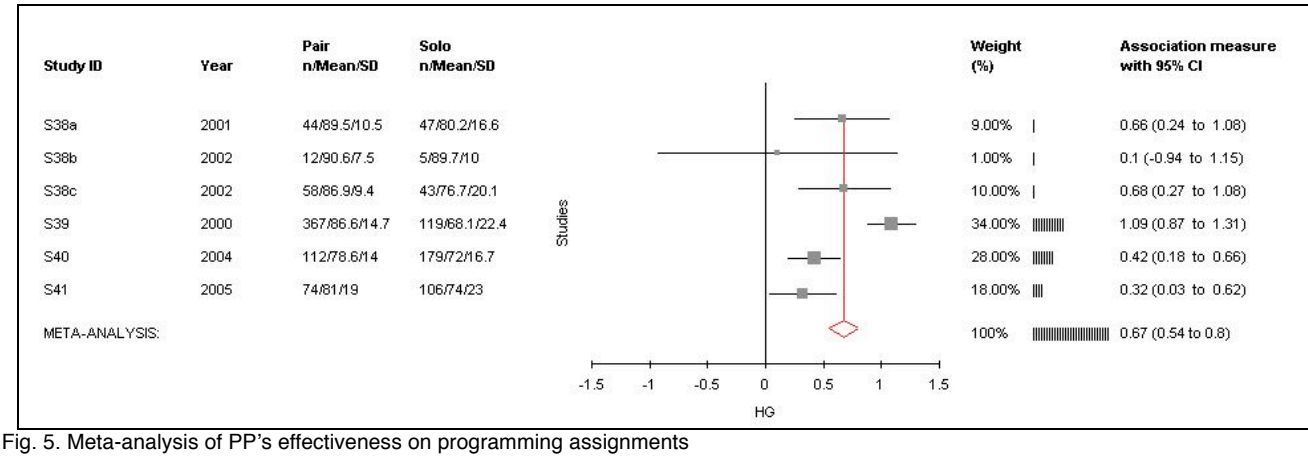

Fig. 5. Meta-analysis of PP's effectiveness on programming assignments

In our SLR, only six studies that reported their statistical results were applicable for a meta-analysis [S38],[S39],[S40],[S41],[S61],[S62]. We used their data to conduct two meta-analyses: one of PP's effectiveness on final exam scores of paired and solo students (MA1), and another of PP's effectiveness on assignments' scores (MA2). MA1 showed a standardized effect size of 0.16, calculated using Hedges's g statistic. Here, we used the standardized mean difference under the fixed effects model as the effect size measure. Effect size was calcu-

lated based on the difference between two means (final exam scores of paired and solo students) divided by the pooled standard deviation, adjusted for small sample bias [31].

The forest plot in Fig. 4 shows MA1 results. The small box indicates the point estimates of effect size in a single study whereas the horizontal line that crosses each study represents the confidence interval for a study's estimate. The diamond at the bottom of the plot represents the pooled effect or the average effect size after pooling all studies. The pooled result from this meta-analysis suggests that the effects of PP were small (i.e. effect size of 0.16) in terms of its practical significance or meaningfulness in improving students' performance in final exams, compared with solo programming. We employed the effect size category from Kampens et al. [31]: small (effect size of 0.000 – 0.376), medium (effect size of 0.378 – 1) or large (effect size of 1.002 – 3.40). Note that some of the studies reported their statistical results for several experiments conducted throughout various academic semesters so we treated each as a separate study in the meta-analysis (e.g. the three experiments in McDowell et al. [S38] are denoted as S38a, S38b, and S38c, respectively).

The second meta-analysis, MA2, showed a medium effect size (see Fig. 5). The pooled effect size of 0.67 suggests that PP was beneficial and effective in helping students get better scores in assignments. We used the software MIX version 1.7 [2], [3] for performing both meta-analyses and generating the forest plots, a good tool for meta-analysis according to Bax et al. [2].

Sub-Question 3 – Measure of Quality

"How was quality measured in the PP studies that used software quality as a measure of effectiveness?"

PP is reported to benefit users by improving software design quality [12]. Of the 73 studies, 32 (44%) investigated the quality of the work produced by paired students, and employed various quality metrics, arranged into four different categories: Internal code quality [44], External code quality [44], Standard Quality Model, and General category (see Table 5 in Appendix B).

Internal code quality was divided into two sub-groups: program size and Object Oriented (OO) design quality. Three PP studies applied program size (e.g. LOC) as a quality metric and found that shorter programs led to higher quality and more maintainable software [S25], [S47], [S21]. However, Vanhanen and Lassenius [S57] argue that LOC is not a reliable metric because fewer lines of code does not guarantee better quality. Thus rather than using LOC as an indicator they analyzed design quality based on a method's size and complexity metrics. There was no significant difference in performance between pair and solo students when effectiveness was measured using program size. In terms of OO design quality, program quality is rated higher for pair programmers when design quality is measured at the class level (i.e. depth of inheritance, coupling and cohesion level) [S5]. However, there is no significant difference between paired and solo students in OO design quality at the method and package levels [S21], [S35], [S57]. Hanks et al. [S21] mention that the mixture in the studies' findings was due to the various levels of task complexities. Vanhanen and Lassenius [S57] report that differences in design quality between pair and solo groups depends on the metric used and may have been affected by the size of the system analyzed.

External code quality (ECQ) was investigated by 16 (22%) of the 73 primary studies. Of these, nine (56%) report that the ECQ produced by paired students is significantly better compared with soloists'. Only one study measured the quality of design diagrams (e.g. Data Flow Diagrams, Relational Databases, Functional Interface Diagram) using the ISO IEC 9126 quality model, presenting mixed findings about the impact of pair work on the quality of design products [S1]. No study measured the quality of design artifacts using UML diagrams.

The general category, comprising expert opinion and academic performance measures such as programming score or project grade, was applied in 14 of the 73 studies (19%). Studies that relied upon expert opinion measured quality using criteria such as the significance of identifiers, how well-organized methods were, use of appropriate indentation and whitespace [S21], functionality and style [S38], output correctness, required documentation, correct use of objects and interface design [S13], and number of defects in specification, expression, and algorithm [S45]. In 5 out of 7 studies, program's quality produced by pairs was superior to the program's quality produced by solo students when quality was measured using course assignment's score or project's grade [S2],[S9],[S30],[S39],[S59]. Four out of 7 studies that employed professional judges (expert opinion) to evaluate the quality of work produced by pair and solo programmers' reported that PP had a positive effect on the quality of work [S5],[S16],[S38],[S73].

3 Discussion

Pair Compatibility Issues

Some studies used a mixture of subjects (undergraduate and graduate students) as a representative sample population [S28],[S29],[S63]. Thus, experience and academic background may have varied widely. Nature of courses, instructors, and instruments used may have also affected the studies' outcome. For instance, the instruments used in two studies that measured confidence level [S22],[S54] were different and this may have contributed to the contradictory findings.

The two compatibility factors investigated most were skill level and personality type. Our synthesis suggested that pairing works effectively when pairing students according to their skill level, supporting previous work by Comrey and Staats [13], who found that group productivity is highly correlated with the ability or competency level of group members.

Regarding the effect of personality type on pair compatibility, studies' findings are mixed. We believe that these mixed results could have been caused by the diversity of

types of instruments used, studies' duration, and the nature of the tasks carried out. Bowers et al. [6], and Mohammed and Angell [50] show that the relationship between personality composition and team performance are highly dependent on the type of task, which supports our view.

There was very little evidence of studies looking at gender and ethnicity issues in relation to improving PP's effectiveness as a pedagogical tool. Two studies investigated whether gender affected PP's effectiveness [S29],[S73]; however their findings are contradictory. Perhaps one of the reasons of such contradictory findings was the duration of each study: the experiments carried out by Choi [S73] had a shorter duration (90 minutes) compared to the three experiments conducted by Katira et al. [S29], each lasting for a full semester.

We found a lack of the studies investigating PP's effectiveness focused on software modeling or methodologies. As our search terms included "programming" this was perhaps not an unexpected result. However, some studies applied PP to a software design phase to investigate whether pair programming was effective at enforcing or diffusing designs' knowledge among the project team members [S3],[S6],[S8],[S9]. Pairing was beneficial in terms of knowledge transfer among pair designers suggesting that PP should not be restricted to coding-related tasks.

Evidence on PP's effectiveness

Of the 70 studies, 19 (27%) measured pair productivity using the time spent in completing the tasks, where most findings (11 studies) indicated that pair programmers effectively completed the assigned tasks in a shorter time. One of the more significant findings from this review was that students perceived greater satisfaction and enjoyment from using PP.

The relatively small overall effect on final exam scores shown in our meta-analyses indicates that PP did not directly improve students' course grades; but the medium effect size on students' assignment scores suggests that PP was useful on assignments. Thus, evidence suggests that PP is an effective pedagogical tool that not only benefits students in terms of learning, but also increases their satisfaction and enjoyment. These findings corroborated the results of a meta-analysis on small group and individual learning with technology by Lou et al. [42], who found that students learning in pairs resulted in better cognitive and affective outcomes.

The cognitive theories of cooperative learning research emphasize two major benefits of students working together. First, the interaction that occurs while working together helps students increase their "mastery of critical concepts" [61]. When peers engage in a discussion, cognitive conflicts and reasoning are more likely to happen, and this type of interaction helps improve students' achievement. Second, the ability to elaborate or explain will consequently help students in retaining knowledge. PP exhibits these elements of interaction and elaboration.

A review of research in education shows that cooperative learning can be beneficial in accelerating students' achievement when the emphasis is placed upon the *group's goals* and *individual accountability* factors [60]. By default, PP incorporates those factors and students are also accountable for their own individual achievement in exams.

Measuring Quality

The work produced by paired students was of high quality when measured using expert opinion and academic performance. Thirty-two studies investigated PP's quality aspects, and results in general report significant findings showing that quality of design/code developed by paired students is considerably superior to soloists, corroborating meta-analysis results reported by Dyba et al. [19].

Although results were in general supportive of PP, the effects of PP towards internal code quality seem to be unclear/contradictory. Most studies either provide a mixture of findings or report that PP had no impact on the internal code quality (Table 5, Appendix B). For example, Madeyski [S35] reports that package dependencies in an OO design were not significantly affected by the pair or solo development. Since no other evidence was, as far as we know, available regarding this issue, a replication study needs to be carried out to support or refute this. We believe that the unclear evidence as to whether PP improves internal code quality can be attributed to several reasons such as the types of tasks, level of task complexity, size of the analyzed system, and studies' context.

Steiner's theories emphasized that the potential performance of a group is very much dependent on the type of task at hand, and whether the group members have adequate resources (i.e. skills, tools, and effort) in order to carry out the task [63]. Our SLR showed that the tasks given to paired students varied from simple programming assignments to complex J2EE distributed applications. We believe that given this range in task complexity internal code quality is likely to be affected by application size and the choice of metrics used to measure the quality of code design. Vanhanen and Lassenius [65] comment that measuring code design quality can be unreliable due to the varying amount of functionality in different applications. Our review supports their findings and we suggest that measuring quality based on external metrics (i.e. test cases passed, number of defects, etc.) would be a better mechanism to evaluate code quality. Finally, while the majority of studies investigated code quality, only three looked at the quality of design documents using a ISO model [S1] and/or design scores [S9], [S30].

Implications for Research

Our SLR found that personality was one of the most common factors investigated in PP studies. However, the results from existing studies are inconsistent in terms of the effects of personality towards PP's effectiveness. Existing literature in psychology shows that students' personality traits play an important role to predict their academic success and are also considered as one of the critical success factors in determining teamwork success [8], [22]. In one of the meta-analytic studies, Bowers et al. [6] investigated whether homogeneous personality teams

outperformed heterogeneous personality teams; their findings show a partial support for the latter. Because these studies were conducted mostly in the psychology domain, further research should be done in other fields too (e.g. CS/SE) to investigate whether personality composition can affect PP's effectiveness as a pedagogical tool. In addition, the issue of whether homogeneity or heterogeneity of personality is good for PP is not yet clearly understood. We also identified that most PP studies investigated personality type using the Myer-Briggs Type Indicator. We suggest further research should be undertaken using other credible personality measurement frameworks such as the Five-Factor Model [45].

We also observed that in many of the PP experiments confounding effects were not controlled, leading to results that could very likely be biased [37]. For example, the validity of some of the results might have been confounded by the method of pair formation. For instance, instead of randomly assigning students to treatment and control, some studies let the students decide whether to pair or not [S20],[S28],[S63]. This means it is possible that most of the students who paired were enthusiastic about using PP, thus biasing the results. In order to improve the quality of empirical research, researchers can refer to available guidelines for conducting empirical research in SE [37] and for reporting controlled experiments in SE [30].

Our SLR showed only 17 studies (23%) investigated factors that may affect PP's effectiveness, including pair compatibility. However, there was no clear relationship determined between pair compatibility and PP's effectiveness. Some studies investigated the perceived compatibility of students towards their partners but no evidence was available on whether pair compatibility improved PP's effectiveness. Research in psychology has investigated the effects of interpersonal compatibility on group productivity using Schutz's FIRO theory. Results suggest that the productivity of compatible groups was greater than that of incompatible groups [40]. We suggest that the association of these factors be investigated in future PP studies.

Most of the PP studies we reviewed (85%) required students to engage in tasks only related to coding or application development, thus suggesting that PP had been rarely employed in courses where students were exposed to software design/modeling tasks. This clearly indicates that further research needs to be conducted to investigate whether PP can be an effective pedagogical tool to learn CS/SE in topics other than coding. There is also a need to increase the number of studies investigating factors potentially affecting PP's effectiveness in order to aggregate results.

Implications for CS/SE Educators

One of the key repercussions for CS/SE educators relates to how to implement PP. The results of this SLR suggest that the most effective pairing configuration is to pair students of similar competency level using as a basis their exam scores/GRE/GPA or programming experience. We suggest that educators who are willing to practice PP in their classroom should pair students according to their skill or competency level to achieve greater pair compatibility.

The SLR suggests that students perceive higher satisfaction when working in pairs. According to the Vygotskian theory known as "zone of proximal development" [66], students are capable of achieving higher intellectual level when collaborating with other students rather than working alone. Students who pair programmed were satisfied with the pairing experience mainly because PP helped them increase their knowledge and gain greater confidence, besides improving their social interaction skills. PP can also assist instructors and educators in reducing their own workload as there will be a smaller number of assignments or projects to be graded.

Threats to the Validity of the Results

Several factors need to be taken into account when generalizing the results of this SLR. During the process of identifying the relevant literature we only considered as primary sources articles published electronically, thus neglecting studies that might have appeared in conference proceedings or journals that were not published online. This was particularly applicable to material published before 1987. However, since the PP practice, as considered in our SLR, was proposed in 1999 [4], we feel it is unlikely that PP studies are not available online. Furthermore, we used an extensive list of search databases and included in our search all the databases we were aware of where PP primary sources had been published.

Another threat relates to the issue of handling the review. The first author was responsible for developing the protocol and carrying out the major tasks involved in each of the SLR stages, which may have unwittingly had some influence on the SLR results. However, the other authors provided detailed feedback during all the stages part of the SLR (e.g. protocol's preparation, primary studies' selection, data extraction's quality assurance, compiling of results), which we believe should have minimized, if not removed, any possible bias in the results presented herein. In addition, we followed very closely the recommendations suggested in the SLR guidelines [35] in order to avoid bias. Publication bias is also considered as a common issue in SLRs [35]. In dealing with publication bias, we make used the following strategy: (i) develop and continuously refine the SLR protocol, in particular during the search process; and (ii) include searching of grey literature such as theses, dissertations, and technical reports so that the search process covers as many studies as possible.

Future work

As part of our future work, we are currently conducting an experiment investigating personality and gender aspects towards PP's effectiveness where students are involved in software design-related tasks. Since existing PP studies heavily relied upon MBTI to measure personality, and MBTI has been widely criticized as a good personality framework to be employed [58],[75], we are using IPIP-NEO as an instrument to measure personality.

IPIP-NEO was proposed based on the Five-Factor Model personality, which is currently considered the predominant taxonomy of personality by personality-focused psychologists [7],[26],[14],[22]. In addition, we are also looking into understanding the relationship between pair compatibility and its effect on pair effectiveness.

4 Conclusions

This paper described a SLR targeted at empirical studies of PP's effectiveness and/or pair compatibility conducted in higher education settings. A total of 73 primary studies were used in our SLR, from which 14 compatibility factors potentially affecting PP's effectiveness were identified. Of these, personality type, actual and perceived skill level were the three factors investigated the most in PP studies. However, the effects of personality type towards pair compatibility and/or PP's effectiveness were inconclusive. PP studies that investigated actual and perceived skill levels achieved a consensus suggesting that students prefer to pair with someone of similar skills to themselves. Results also showed that a pair works well when both students have similar abilities and motivation to succeed in a course.

Evidence showed that various metrics were employed to measure PP's effectiveness, classified into technical productivity, program/design quality, academic performance, and satisfaction. We also found out that the metric used most often to measure pair productivity was the time spent in completing the tasks. Paired students usually completed the assigned tasks in shorter duration than solo students. Almost all studies' findings reported that students' satisfaction was higher when using PP compared with working individually. In terms of academic performance, the results of our meta-analysis of PP's effectiveness indicated that PP had no significant advantage in improving students' performance in final exams over solo programming (effect size = 0.16). However, in the second meta-analysis, the pooled results suggested that PP was effective in helping students obtaining better scores in their assignments (effect size = 0.67).

There were numerous methods employed when considering quality as a measure of PP's effectiveness. Based on our review, research on quality aspects was classified into *internal* and *external code quality*, *standard quality model*, and *general* categories. Of all categories, external code quality and general category were the two researched the most. Findings indicated that when quality was measured according to academic performance and expert opinion, students who pair programmed produced a better quality program compared to students who programmed alone. However, when the quality of the work produced by pair and solo students was measured using metrics at the internal code level, results were contradictory.

We also discussed a number of implications of the SLR results for research and practice, including the need to replicate PP studies in areas where findings were inconsistent, or to conduct studies in areas where there is scarcity of or no evidence regarding the effect of certain compatibility factors towards PP's effectiveness as a pedagogical tool.

The results of the SLR suggest that PP was rarely employed in courses where students were exposed to software design/modeling tasks, thus we believe that this is a fruitful area for future work. The review results suggest that paired students achieve productivity similar or better than solo students; and indicate that implementing PP in the classroom or lab does not lead to any detrimental effect on students' academic performance. This is in line with research evidence in education that clearly supports collaborative learning as an effective instructional method in higher education [54].

Acknowledgment

This research is partially funded by the Ministry of Higher Education Malaysia. The authors would like to thank the reviewers and associate editor for the insightful comments and suggestions made to this paper, and Liz Hardley for the advice and assistance given during the literature search process.

References

[1] M. Ally, F. Darroch, and M. Toleman, "A Framework for Understanding the Factors Influencing Pair Programming Success," *Extreme Programming and Agile Processes in Software Engin.*, LNCS 3556, Springer, pp. 82-91, 2005.

[2] L. Bax, L.-M. Yu, N. Ikeda, H. Tsuruta, and K.G. Moons, "Development and Validation of MIX: Comprehensive Free Software for Meta-Analysis of Causal Research Data," *BMC Medical Research Methodology*, vol. 6, 2006.

[3] L. Bax, L.-M. Yu, N. Ikeda, H. Tsuruta, and K.G. Moons. (2006, October). Comprehensive Free Software for Meta-Analysis of Causal Research Data .Version 1.7. , *BMC Medical Research Methodology* [Online]. 6(50). http://www.biomedcentral.com/1471-2288/6/50

[4] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.

[5] S.B. Berenson, K.M. Slaten, L. Williams, and C.-w. Ho, "Voices of Women in a Software Engineering Course: Reflections on collaboration," *J. of Educ. Resources in Computing*, vol. 4, no.1, 2004.

[6] C.A. Bowers, J.A. Pharmer, and E. Salas, "When Member Homogeneity is Needed in Work Teams: A Meta-Analysis," *Small Group Research*, vol. 31, pp. 305 - 327, 2000.

[7] G. Burch, and N. Anderson, "Personality as a Predictor of Work-related Behavior and Performance: Recent Advances and Directions for Future Research," *International Review of Industrial and Organizational Psychology*, G. P. Hodgkinson and J. K. Ford, eds., pp. 261-305: John Wiley & Sons Ltd., 2008.

[8] V.V. Busato, F.J. Prins, J.J. Elshout & C. Hamaker, Intellectual ability, Learning Style, Personality, Achievement Motivation and Academic Success of Psychology Students in Higher Education. *Personality and Individual Differences,* vol. 29, no. 6, pp. 1057-1068, 2000.

[9] E.A. Chaparro, "Factors Affecting the Perceived Effectiveness of Pair Programming in Higher Education," in *17th Workshop of the Psychology of Programming Interest Group, Sussex University*, pp. 5-18, 2005.

[10] S.R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE Trans. on Software Engin.*, vol. 20, pp. 476-493, 1994.

[11] D.C. Cliburn, "Experiences with Pair programming at a Small College," *J. of Computing Sciences in Colleges*, vol. 19, no. 1, pp. 20-29, 2003.

[12] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," in *Proc. 2nd Int'l Conf. of Extreme Programming and Flexible Processes in Software Engin.*, 2001.

[13] A.L. Comrey and C.K. Staats, Group Performance in a Cognitive Task. Journal of Applied Psychology, vol. 39, pp. 354-356, 1955.

[14] M.A. Conard, "Aptitude is Not Enough: How Personality and Behavior Predict Academic Performance," *J. of Research in Personality*, vol. 40, pp. 449–346, 2006.

[15] P.T. Costa and R.R. McCrae, *The NEO Personality Inventory Manual*. Odessa, Florida: Psychological Assessment Resources, 1985.

[16] J.W. Creswell, *Research Design Qualitative, Quantitative and Mixed Method Approaches*: SAGE Publications, 2003.

[17] I.K. Crombie, *The Pocket Guide to Appraisal*: BMJ Books, 1996.

[18] T.H. DeClue, "Pair Programming and Pair Trading: Effects on Learning and Motivation in a CS2 Courses," *J. of Computing Sciences in Colleges*, vol. 18, no.5, pp. 49-56, 2003.

[19] T. Dyba, E. Arisholm, D.I.L. Sjoberg, J.E. Hannay, and F. Shull, "Are Two Heads Better Than One? On the Effectiveness of Pair Programming," *IEEE Software,* vol. 24, no.5, pp. 12 – 15, 2007.

[20] T. Dyba and T. Dingsoyr, "Empirical Studies of Agile Software Development: A Systematic Review," *Inform. and Software Technol.*, Elsevier, vol. 50, pp. 833-859, 2008.

[21] B.V. Elsevier (2008). Scopus Overview: What is it? [Online]. Available: http://www.info.scopus.com/about/

[22] T. Farsides and R. Woodfield, "Individual Differences and Undergraduate Academic Success: The Roles of Personality, Intelligence, and Application," *Personality and Individual Differences,* vol. 34, pp. 1225-1243, 2003.

[23] R.M. Felder and L.K. Silverman, "Learning and Teaching Styles in Engineering Education," *Engin. Educ.,* vol. 78, pp. 674 - 681, 1988.

[24] A. Fink, *Conducting Research Literature Reviews. From the Internet to Paper*: Sage Publication, Inc., 2005.

[25] D.R. Forsyth, Group Dynamics, 4th Ed., Thomson Wadsworth, 2006.

[26] A. Furnham, "The Big Five Vs the Big Four: The Relationship between Myers-Briggs Type Indicator (MBTI) and NEO-PI five factor model of personality," *Personality and Individual Differences,* vol. 21, pp. 303-307, 1996.

[27] T. Greenhalgh, *How to read a paper: The basics of evidence-based medicine*: BMJ Books, 2000.

[28] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic, "Program Quality with Pair Programming in CS1," ACM. SIGCSE Bulletin, vol. 36, pp. 176-180, 2004.

[29] E.V. Howard, "Attitudes on Using Pair-Programming," J. of Educ. Technol. Syst., vol. 35, pp. 89-103, 2006.

[30] A. Jedlitschka, and D. Pfahl, "Reporting Guidelines for Controlled Experiments in Software Engineering," in Int'l Symposium on Empirical Software Engineering, pp. 95-104, 2005.

[31] V.B. Kampenes, T. Dyba, J.E. Hannay, and D.I.K. Sjoberg, "A Systematic Review of Effect Size in Software Engineering," *Inform. and Software Technol.*, vol. 49, pp. 1073-1086, 2007.

[32] N. Katira, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer, "On Understanding Compatibility of Student Pair Programmers," *ACM. SIGCSE Bulletin*, vol. 36, pp. 7-11, 2004.

[33] D. Keirsey and M. Bates, *Please Understand Me.* California: Del Mar, Prometheus Book Company, 1984.

[34] K.S. Khan, R. Kunz, J. Kleijnen, and G. Antes, *Systematic Reviews to Support Evidence-based Medicine*. London: The Royal Society of Medicine Press Ltd., 2003.

[35] B.A. Kitchenham and S. Charters, *Procedures for Performing Systematic Literature Reviews in Software Engineering*, EBSE Tech. Report version 2.3, EBSE-2007-01, Software Engin. Group, Keele Univ., Univ. of Durham, UK, 2007.

[36] B.A. Kitchenham, E. Mendes, and G.H. Travassos, "Cross versus Within-Company Cost Estimation Studies: A Systematic Review," *IEEE Trans. on Software Engin.,* vol. 33, pp. 316-329, 2007.

[37] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. on Software Engin.*, vol. 28, pp. 721–734, 2002.

[38] L. Layman, "Changing Students' Perceptions: an Analysis of the Supplementary Benefits of Collaborative Software Development," in *Proc. 19th Conf. on Software Engin. Educ. & Training,* IEEE CS Press, pp. 159-166, 2006.

[39] P.D. Leedy and J.E. Ormrod, *Practical Research Planning and Design*, 8th ed.: Pearson Merril Prentice Hall, 2005.

[40] W.W. Liddel, and J. John W. Slocum, "The Effects of Individual-Role Compatibility upon Group Performance: An Extension of Schutz's FIRO Theory," *The Academy of Management Journal,* vol. 19, no. 3, pp. 413-426, 1976.

[41] J.A. Livermore, "What Elements of XP are being Adopted by Industry Practitioners?," in *Proc. of the IEEE, Southeast Conf.*, pp. 149-152, 2006.

[42] Y. Lou, P.C. Abrami, and S. d'Apollonia, "Small Group and Individual Learning with Technology: A Meta-Analysis," *Review of Educational Research,* vol. 71, no. 3, pp. 449-521, 2001.

[43] R.C. Martin, "OO Design Quality Metrics: An Analysis of Dependencies," *Position Paper, Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA'04,* 1994.

[44] S. McConnel, *Code Complete: a Practical Handbook of Software Construction*: Microsoft Press, 1993.

[45] R.R. McCrae, R.R. and O.P. John, An Introduction to the Five-Factor model and its application. *J. of Personality*, vol. 60, pp. 175-215, 1992.

[46] C. McDowell, L. Werner, H.E. Bullock, and J. Fernald, "The Impact of Pair Programming on Student Performance, Perception and Persistence," in *25th Int'l Conf. on Software Engin. (ICSE'03)*, pp. 602-607, 2003.

[47] E. Mendes, L. Al-Fakhri, and A. Luxton-Reilly, "Investigating Pair-programming in a 2nd-year Software Development and Design Computer Science Course," in ACM. *SIGCSE Bulletin,* pp. 296-300, 2005.

[48] E. Mendes, "A Systematic Review of Web Engineering Research," in Int'l Symp. on Empirical Software Engin., pp. 498–507, 2005.

[49] E. Mendes, L. Al-Fakhri, and A. Luxton-Reilly, "A Replicated Experiment of Pair-programming in a 2nd-year Software Development and Design Computer Science Course," in *Proc. 11th Annual SIGCSE Conf. on Innovation and Technol. in Comput. Science Educ., ITiCSE06,* ACM Press, pp. 108-112, 2006.

[50] S. Mohammed and L.C. Angell, "Personality Heterogeneity in Teams: Which Differences Make a Difference for Team Performance?," *Small Group Research,* vol. 34, pp. 651-677, 2003.

[51] M.M. Muller and F. Padberg, "An Empirical Study about the Feelgood Factor in Pair Programming," *Proc. 10th Int'l Symp. on Software Metrics,* IEEE Comput. Soc., pp. 151-158, 2004.

[52] I. Myers-Briggs, M.H. McCaulley, N.L. Quenk, and A. Hammer, *MBTI Manual (A Guide to the Development and use of the Myers Briggs Type Indicator)*, 3rd ed edition ed. vol. Consulting Psychologists Press, 1998.

[53] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the CS1 Experience with Pair Programming," *ACM. SIGCSE Bulletin*, vol. 35, pp. 359-62, 2003.

[54] A.I. Nevin, K.A. Smith, and A. Udvari-Solner, "Cooperative Group Learning and Higher Education," *Creativity and Collaborative Learning: A Practical Guide to Empowering Students and Teachers*, J. S. Thousand, R. A. Villa and A. I. Nevin, eds., Maryland: Paul H. Brookes Publishing Co., Inc., 1994.

[55] J.T. Nosek, "The Case for Collaborative Programming," *Commun. of the ACM*, vol. 41, pp. 105-108, 1998.

[56] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences A Practical Guide*: Blackwell Publishing, 2006.

[57] L.M. Pickard, B.A. Kitchenham, and P.W. Jones, "Combining Empirical Results in Software Engineering," *Inform. and Software Technol.*, vol. 40, pp. 811-821, 1998.

[58] D.J. Pittenger, "Measuring the MBTI...and coming up short.," *J. of Career Planning and Employment*, pp. 48-53, 1993.

[59] K.M. Slaten, M. Droujkova, S.B. Berenson, L. Williams, and L. Layman, "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction," in *Proc. Agile 2005,* IEEE Comput. Soc., pp. 323-330, 2005.

[60] R.E. Slavin, "Cooperative Learning," *Review of Educational Research,* vol. 50, no. 2, pp. 315-342, 1980.

[61] R.E. Slavin, *Cooperative Learning: Theory, Research and Practice,* Englewood Cliffs, N.J.: Prentice Hall, 1990.

[62] L. Spencer, J. Ritchie, J. Lewis, and L. Dillon, "Quality in Qualitative Evaluation: A Framework for Assessing Research Evidence," *Government Chief Social Researcher's Office*, London, 2003.

[63] L.D. Steiner, *Group Process and Productivity*, Academic Press, 1972.

[64] L. Thomas, M. Ratcliffe, and A. Robertson, "Code Warriors and Code-a-phobes: a Study in Attitude and Pair Programming," *ACM. SIGCSE Bulletin*, vol. 35, pp. 363-7, 2003.

[65] J. Vanhanen and C. Lassenius, "Effects of Pair Programming at the Development Team Level: an Experiment," *Proc. Int'l Symp. Software Engin., ISESE 05*, IEEE CS Press, pp. 336-345, 2005.
[66] L.S. Vygotsky, Mind in Society: The development of higher psychological Processes, Cambridge: MIT Press, 1978.
[67] L.L. Werner, B. Hanks, and C. McDowell, "Pair-Programming Helps Female Computer Science Students," J. of Educational Resources in Computing, vol. 4, 2004.
[68] L.A. Williams, "The Collaborative Software Process", Ph.D. Dissertation, Univ. of Utah, 2000.
[69] L.A. Williams and R.R. Kessler, "The Effects of "pair-pressure" and "pair-learning" on Software Engineering Education," *13th Conf. on Software Engin. Educ. and Training,* IEEE CS Press, pp. 59-65, 2000.
[70] L. Williams and R.R. Kessler, *Pair Programming Iluminated*. Boston, MA, USA: Addision-Wesley Longman Publishing Co., Inc., 2002.
[71] L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair Programming," *IEEE Software*, vol. 17, pp. 19-25, 2000.
[72] L. Williams, L. Layman, J. Osborne, and N. Katira, "Examining the Compatibility of Student Pair Programmers," in *Proc. AGILE 2006 Conf., AGILE'06,* IEEE Comput. Soc., 2006.
[73] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner, "Building Pair Programming Knowledge Through a Family of Experiments," in *Proc. 2003 Int'l Symp. on Empirical Software Engin., ISESE 2003,*.IEEE Comput. Soc., pp. 143-152, 2003.
[74] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction*. Boston: Kluwer Academic Publisher, 2000.
[75] R. Zemke, "Second Thoughts about the MBTI," *Training*, vol. 29, no. 4, pp. 43-47, 1992.

APPENDIX A

List of Included Studies

[S1] H. Al-Kilidar, P. Parkin, A. Aurum, and R. Jeffery, "Evaluation of Effects of Pair Work on Quality of Designs," in Proc. 2005 Australian Software Eng. Conf. IEEE Comput. Soc., pp. 78-87, 2005.
[S2] V. Balijepally, "Task Complexity and Effectiveness of Pair Programming: An Experimental Study," Ph.D. dissertation, The University of Texas at Arlington, Texas, USA, 2006.
[S3] E. Bellini, G. Canfora, F. Garcia, M. Piattini, and C.A. Visaggio, "Pair Designing as Practice for Enforcing and Diffusing Design Knowledge," J. of Software Maintenance and Evolution-Research and Practice, vol. 17, pp. 401-423, 2005.
[S4] S.B. Berenson, K.M. Slaten, L. Williams, and C.-w. Ho, "Voices of Women in a Software Engineering Course: Reflections on Collaboration," J. of Educational Resources in Computing, vol. 4, no.1, article no. 3, 2004.
[S5] T. Bipp, A. Lepper, and D. Schmedding, "Pair Programming in Software Development Teams An Empirical Study of its Benefits," Information and Software Technology, vol. 50, pp. 231-240, 2008.
[S6] G. Canfora, A. Cimitile, and C.A. Visaggio, "Working in Pairs as a Means for Design Knowledge Building: An Empirical Study," in 12th IEEE Int'l Workshop on Program Comprehension (IWPC'04), pp. 62-68, 2004.
[S7] G. Canfora, A. Cimitile, and C.A. Visaggio, "Empirical Study on the Productivity of the Pair Programming," Extreme Programming and Agile Processes in Software Eng. Proc. 6th Int'l Conf. XP 2005, LNCS 3556, Springer, pp. 92-99, 2005.
[S8] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. Aaron Visaggio, "Confirming the Influence of Educational Background in Pair-Design Knowledge Through Experiments," in ACM Symp. on Applied Computing (SAC'05), New Mexico, USA, pp. 1478-1484, 2005.
[S9] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C.A. Visaggio, "Performances of Pair Designing on Software Evolution: A Controlled Experiment," in 10th European Conf. on Software Maintenance and Reengineering. IEEE Comput. Soc., pp. 195-202, 2006.
[S10] H. Srikanth, L. Williams, E. Wiebe, C. Miller, and S. Balik, "On Pair Rotation in the Computer Science Course," in 17th Conf. on Software Eng. Educ. and Training (CSEET'04), pp. 144-149, 2004.
[S11] C. Lan and B. Ramesh, "An exploratory study on the effects of pair programming," in 8th Int'l Conf. on Empirical Assessment in Software Eng., EASE 2004 Workshop - 26th Int'l Conf. on Software Eng. IEE., pp. 21-28, 2004.
[S12] J.C. Carver, L. Henderson, L. He, J. Hodges, and D. Reese, "Increased Retention of Early Computer Science and Software Engineering Students Using Pair Programming," in 20th Conf. on Software Eng. Educ. & Training, pp. 115-122, 2007.
[S13] J. Chao and G. Atli, "Critical Personality Traits in Successful Pair Programming," in AGILE'06: IEEE Comput. Soc., pp. 89-93, 2006.
[S14] E.A. Chaparro, A. Yuksel, P. Romero, and S. Bryant "Factors Affecting the Perceived Effectiveness of Pair Programming in Higher Education," in 17th Workshop of the Psychology of Programming Interest Group, Sussex University, pp. 5-18, 2005.
[S15] D.C. Cliburn, "Experiences with Pair Programming at a Small College," J. of Computing Sciences in Colleges, vol. 19, pp. 20-29, 2003.
[S16] T.H. DeClue, "Pair Programming and Pair Trading: Effects on Learning and Motivation in a CS2 Courses," J. of Computing Sciences in Colleges, vol. 18, no.5, pp. 49-56, 2003.
[S17] M.A. Domino, R.W. Collins, H.A.R., and C.F. Cohen, "Conflict in Collaborative Software Development," in Proc. 2003 Conf. on Comput. Personnel Research: Freedom in Philadelphia-leveraging differences and diversity in the IT workforce, pp. 44-51, 2003.
[S18] M.A. Domino, R. W. Collins, and A.R. Hevner, "Controlled Experimentation on Adaptations of Pair Programming," Information Technology and Management, vol. 8, pp. 297-312, 2007.
[S19] S.F. Freeman, B.K. Jaeger, and J.C. Brougham, "Pair programming: More Learning and Less Anxiety in a First Programming Course," in ASEE Annual Conf. Proc., pp. 8885-8893, 2003.
[S20] E.F. Gehringer, "A Pair-Programming Experiment in a Non-Programming Courses," in 18th Annual ACM SIGPLAN Conf. on Object Oriented Programming Systems, Languages and Applications-OOPSLA'03, California, pp. 187-190, 2003.
[S21] B. Hanks, C. McDowell, D. Draper, and M. Krnjajic, "Program Quality with Pair Programming in CS1," in ACM. SIGCSE Bulletin, vol. 36, pp. 176-180, 2004.
[S22] B. Hanks, "Student Attitudes Toward Pair Programming," in 11th Annual Proc. SIGCSE Conf. on Innovation and Technology in Comput. Science Educ. (ITiCSE06), pp. 113-117, 2006.
[S23] S. Heiberg, U. Puus, P. Salumaa, and A. Seeba, "Pair-Programming Effect on Developers Productivity," Proc. 4th Int'l Conf. XP 2003, LNCS 2675, Springer, pp. 215-224, 2003.
[S24] C.-w. Ho, "Examining Impact of Pair Programming on Female Students," Dept. of Comput. Science, North Carolina State University, Raleigh, TR-2004-20, 2004.
[S25] M. Ciolkowski and M. Schlemmer, "Experiencing with a Case Study on Pair Programming," in Proc. First Int'l Workshop Empirical Studies in Software Engin., Finland, 2002.
[S26] S. D. James and J. C. Hansen, "Student-based Pair Programming: an Examination," in 6th World Multiconf. on Systemics, Cybernetics and Informatics, Orlando, vol. 8, pp. 485-489, 2002.
[S27] A. Janes, B. Russo, P. Zuliani, and G. Succi, "An Empirical Analysis on the Discontinuous Use of Pair Programming," Extreme Programming and Agile Processes in Software Engin. – Proc. 4th Int'l Conf. XP 2003, LNCS 2675, Springer, pp. 205-214, 2003.
[S28] N. Katira, L. Williams, E. Wiebe, C. Miller, S. Balik, and E. Gehringer, "On Understanding Compatibility of Student Pair Programmers," in ACM. SIGCSE Bulletin, vol. 36, pp. 7-11, 2004.
[S29] N. Katira, L. Williams, and J. Osborne, "Towards Increasing the Compatibility of Student Pair Programmers," in 27th Int'l Conf. on Software Engin., ICSE'05 St Louis, Missouri, IEEE Comput. Soc., pp. 625-626, 2005.
[S30] S. Kuppuswami and K. Vivekanandan, "The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments," Informatics in Education, vol. 3, pp. 251-266, 2004.
[S31] L. Layman, L. Williams, J. Osborne, S. Berenson, K. Slaten, and M. Vouk, "How and Why Collaborative Software Development

Impacts the Software Engineering Course," in Proc. 35th Annual Conf. Frontiers in Educ., FIE '05, pp. T4C-9-T4C-14, 2005.

[S32] L. Layman, "Changing Students' Perceptions: an Analysis of the Supplementary Benefits of Collaborative Software Development," in Proc. IEEE 19th Conf. on Software Engin. Educ. & Training, pp. 159-166, 2006.

[S33] K.M. Lui and K.C.C. Chan, "Pair Programming Productivity: Novice-novice vs. Expert-expert," Int'l J. of Human-Computer Studies, vol. 64, pp. 915-925, 2006.

[S34] L. Madeyski, "Preliminary Analysis of the Effects of Pair Programming and Test-driven Development on the External Code Quality," in Software Engin. Evolution and Emerging Technologies, K. Zieliński and T. Szmuc., Eds. IOS Press, pp. 113–123, 2005.

[S35] L. Madeyski, "The Impact of Pair Programming and Test-driven Development on Package Dependencies in Object-oriented Design - an Experiment," Product-Focused Software Process Improvement, Proc. 7th Int'l Conf. PROFES 2006, Springer, pp. 278-289, 2006.

[S36] L. Madeyski, "On the Effects of Pair Programming on Thoroughness and Fault-Finding Effectiveness of Unit Tests," in Product-Focused Software Process Improvement PROFES 2007, Springer-Verlag, Berlin, pp. 207-221, 2007.

[S37] C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The Effects of Pair-programming on Performance in an Introductory Programming Course," in ACM. SIGCSE Bulletin. vol. 34, pp. 38-42, 2002.

[S38] C. McDowell, B. Hanks, and L. Werner, "Experimenting with Pair Programming in the Classroom," ACM. SIGCSE Bulletin, vol. 35, pp. 60-64, 2003.

[S39] C. McDowell, L. Werner, H.E. Bullock, and J. Fernald, "The Impact of Pair Programming on Student Performance, Perception and Persistence," in 25th Int'l Conf. on Software Engin., ICSE'03, pp. 602-607, 2003.

[S40] E. Mendes, L. Al-Fakhri, and A. Luxton-Reilly, "Investigating Pair-programming in a 2nd-year Software Development and Design Computer Science Course," in SIGCSE Bulletin, pp. 296-300, 2005.

[S41] E. Mendes, L. Al-Fakhri, and A. Luxton-Reilly, "A Replicated Experiment of Pair-programming in a 2nd-year Software Development and Design Computer Science Course," in Proc. 2006 11th Annual SIGCSE Conf. on Innovation and Technology in Comput. Science Educ., ITiCSE06, pp. 108-112, 2006.

[S42] M.M. Muller and F. Padberg, "An Empirical Study about the Feelgood Factor in Pair Programming," Proc. 10th Int'l Symp. on Software Metrics, IEEE Comput. Soc., pp. 151-158, 2004.

[S43] M.M. Muller, "Are Reviews an Alternative to Pair Programming?," Empirical Software Engin., vol. 9, pp. 335-351, 2004.

[S44] M.M. Muller, "Two Controlled Experiments Concerning the Comparison of Pair Programming to Peer Review," J. of Syst. and Software, vol. 78, pp. 166-179, 2005.

[S45] M.M. Muller, "Do Programmer Pairs Make Different Mistakes than Solo Programmers?," J. of Syst. and Software, vol. 80, no. 9, pp. 1460-1471, 2006.

[S46] M.M. Muller, "A Preliminary Study on the Impact of a Pair Design Phase on Pair Programming and Solo Programming," Inform. and Software Technology, vol. 48, no. 5, pp. 335-344, 2006.

[S47] J. Nawrocki and A. Wojciechowski, "Experimental Evaluation of Pair Programming," in Proc. European Software Control and Metrics Conf. (ESCOM 01), pp. 269-276, 2001.

[S48] T.C. Ahren, "Work in Progress - Effect of Instructional Design and Pair Programming on Student Performance in an Introductory Programming Course," in Proc. 35th Annual Conf. Frontiers in Educ., FIE '05, pp. F3E-11-12, 2005.

[S49] M. Phongpaibul and B. Boehm, "An Empirical Comparison Between Pair Development and Software Inspection in Thailand," in Proc. 2006, 5th ACM-IEEE Int' Symp. on Empirical Software Engin, ACM Press, pp. 85-94, 2006.

[S50] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, "Investigating the Impact of Personality Types on Communication and Collaboration-viability in Pair Programming - an Empirical Study," Extreme Programming and Agile Processes in Software Engin. – Proc. 7th Int'l Conf. XP 2006, LNCS 4044, Springer, pp. 43-52, 2006.

[S51] K.M. Slaten, M. Droujkova, S. B. Berenson, L. Williams, and L. Layman, "Undergraduate Student Perceptions of Pair Programming and Agile Software Methodologies: Verifying a Model of Social Interaction," in Proc. Agile 2005, IEEE Comput. Soc., pp. 323-330, 2005.

[S52] X. Shaochun and V. Rajlich, "Pair Programming in Graduate Software Engineering Course Projects," in Proc. 35th Annual Conf. Frontiers in Educ., FIE '05, pp. F1G-7-F1G-12, 2005.

[S53] X. Shaochun and V. Rajlich, "Empirical Validation of Test-driven Pair Programming in Game Development," 5th IEEE/ACIS Int'l Conf. on Comput. Inform. Science - In Conjunction with 1st IEEE/ ACIS Workshop on Component-Based Software Engineer Architecture and Reuse, IEEE Comput. Soc., pp. 500-505, 2006.

[S54] L. Thomas, M. Ratcliffe, and A. Robertson, "Code Warriors and Code-a-phobes: a Study in Attitude and Pair Programming," ACM. SIGCSE Bulletin, vol. 35, pp. 363-367, 2003.

[S55] J.E. Tomayko, "A Comparison of Pair Programming to Inspections for Software Defect Reduction," Comput. Science Educ., vol. 12, no. 3, pp. 213-222, 2002.

[S56] T. VanDeGrift, "Coupling Pair Programming and Writing: Learning About Students' Perceptions and Processes," in ACM. SIGCSE Bulletin, pp. 2-6, 2004.

[S57] J. Vanhanen and C. Lassenius, "Effects of Pair Programming at the Development Team Level: an Experiment," Proc. Int'l Symp. Software Engin., ISESE 05, IEEE CS Press, pp. 336-345, 2005.

[S58] T. Van Toll Iii, R. Lee, and T. Ahlswede, "Evaluating the Usefulness of Pair Programming in a Classroom Setting," in 6th IEEE/ACIS Int'l Conf. on Comput. and Inform. Science, ICIS 2007, pp. 302-308, 2007.

[S59] L.A. Williams and R.R. Kessler, "The Effects of "pair-pressure" and "pair-learning" on Software Engineering Education," 13th Conf. on Software Engin. Educ. and Training, IEEE Comput. Soc., pp. 59-65, 2000.

[S60] L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair Programming," IEEE Software, vol. 17, no. 4, pp. 19-25, 2000.

[S61] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller, "In Support of Pair Programming in the Introductory Computer Science Course," Comput. Science Educ., vol. 12, pp. 197-212, 2002.

[S62] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner, "Building Pair Programming Knowledge Through a Family of Experiments," in Proc. 2003 Int'l Symp. on Empirical Software Engin., IEEE Comput. Soc., pp. 143-152, 2003.

[S63] L. Williams, L. Layman, J. Osborne, and N. Katira, "Examining the Compatibility of Student Pair Programmers," in Proc. AGILE 2006 Conf., AGILE'06, IEEE Comput. Soc., 2006.

[S64] D. Winkler and S. Biffl, "An Empirical Study on Design Quality Improvement from Best-practice Inspection and Pair Programming," Proc. Product-Focused Software Process Improvement, LNCS 4034, Springer, pp. 319-333, 2006.

[S65] J.R. Nawrocki, M. Jasinski, L. Olek, and B. Lange, "Pair Programming vs. Side-by-side Programming," in Proc. 12th European Conf. on Software Process Improvement, EuroSPI 2005, LNCS 3792, Springer, pp. 28-38, 2005.

[S66] E. V. Howard, "Attitudes on Using Pair-programming," J. of Educ. Technology Syst., vol. 35, no. 1, pp. 89-103, 2006.

[S67] M. Mujeeb-u-Rehman, Y. Xiaohu, D. Jinxiang, and M. Abdul Ghafoor, "Heterogeneous and Homogenous Pairs in Pair Programming: an Empirical Analysis," in Canadian Conf. on Electrical and Computer Engin., CCECE/CCGEI, pp. 1116-1119, 2006.

[S68] L. Madeyski, "Is External Code Quality Correlated with Programming Experience or Feelgood Factor?," Proc. 7th Int'l Conf. XP 2006, LNCS 4044, Springer, pp. 65-74, 2006.

[S69] B. Simon and B. Hanks, "First Year Students' Impressions of Pair Programming in CS1," in 3rd Int'l Computing Educ. Research Workshop, ICER'07, pp. 73-86, 2007.

[S70] L. Williams, L. Layman, K.M. Slaten, S. B. Berenson, and C. Seaman, "On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering," in Proc. Int'l Conf. on Software Engin., pp. 677-686, 2007.

[S71] B. Hanks, "Problems Encountered by Novice Pair Programmers," in 3rd Int'l Computing Educ. Research Workshop, pp. 159-164, 2007.

[S72] A.T. Williams, "Pair formation in CS1: Self-selection vs Random pairing," Ph.D. Dissertation, Pace University, New York, USA, 2007.

[S73] K.S. Choi, "A Discovery and Analysis of Influencing Factors of Pair Programming," Ph.D. dissertation, Dept. of Information Systems, New Jersey Institute of Technology, New Jersey, USA, 2004.

[S74] H. Gevaert, "Pair Programming Unearthed," M.Sc. thesis, University of Manitoba, Canada, 2007.

APPENDIX B

TABLE 1
COMPATIBILITY OF STUDENT PAIR PROGRAMMERS

	Were the pairs compatible? (Yes/No)								
Courses	**CS1**			**SE**			**OO**		
Factors/Study	[S28]	[S29	[S63]	[S28]	[S29]	[S63]	[S28]	[S29]	[S63]
Personality Type	Yes	-	No	No	No	Yes*	-	No	No
Perceived Skill	Yes	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Actual Skill	No	-	No	No	Yes	Yes*	Yes	Yes	No
Self-esteem	No	-	No	-	-	Yes*	-	-	No
Gender	-	-	-	-	Yes	-	-	Yes	-
Ethnicity	-	-	-	-	Yes	-	-	Yes	-
Work ethic	-	-	-	-	-	Yes	-	-	-
Time Management skill	-	-	-	-	No	-	-	-	-
Learning Style	-	-	-	-	-	Yes*	-	-	-

*(Note: * Indicates partial support)*

TABLE 2
SUMMARY OF EFFECTIVE PAIRING FORMATION

Compatibility factor	**Study(s)**	**Pairing formation**	**Findings**
Actual skill	S8	Similar educational background	The academic background of pair's component affects the knowledge built. Coupling two different academic backgrounds does not seem to improve the performance.
	S11, S15, S29, S28, S63	Similar mid-term/GPA/course grades	Pairs of similar or not very different level of competency were effective (S11). 92.3% students responded that PP made them work better with others. Their exam scores were higher compared with those from previous semesters (S15). S29, S28, S63 identified that students had a preference to pair with a student of similar actual skill (based on SAT/GRE/GPA scores)
	S58	Similar programming skills	PP works best when the programmers are of slightly different skill level, but the gap should not be too broad.
Perceived skill	S14	Paired students with matching skills	Skill level appeared to have a strong influence in the success of PP session. The skill level gap between the partners should not be too broad.
	S28, S29	Similar perceived skill or technical competence level.	Compatibility was highly affected by the perceived skills of a student's partner.
	S63	Similar or higher skill level.	Students preferred to pair with a partner they perceived to be of similar or higher skill level.
Personality type	S73	Different personality type.	Paired students with diverse personality performed significantly better than the pairs of similar personality in terms of code productivity & code design (S73)
	S50	Mixed- personalities and temperaments.	Pairs of mixed-personalities and temperaments showed better performance and collaboration-viability. They achieve better points on assignments and shorter time to complete the tasks.
Gender	S29	Pairs of female students.	Pairs of female students will likely result in a compatible pair. Paired students of mixed gender reported to be less likely compatible.
Ethnicity	S29	Pairs of minority students.	S29 reported that a pair with only minority students is more likely to be compatible. Paired students with the same gender were also reported to be more comfortable working with each other.
Work ethic	S63	Similar work ethic.	Students preferred to work with someone who had similar intention to success in the course as themselves.

TABLE 3
CATEGORIES OF METRICS TO MEASURE PP'S EFFECTIVENESS

Categories	**Metrics used to measure effectiveness**	**Significant +ve effect**	**Sig. –ve effect**	**No significant effect**	**Mixed findings**
Technical productivity 31 studies (44%)	Time Spent	S7, S9, S30, S31, S33, S49, S52, S53, S60, S4, S51	S65	S19, S25, S38, S42, S44, S46, S47	-
	Knowledge & Skill transfer	S3, S6, S8, S26, S27	-	-	-
	Task performance & Code accuracy	S18, S50	-	S43, S74	-
	Number of solution that satisfy test cases	-	-	S23, S57	-
	Number & types of problem	-	-	-	S71

Program/design quality 30 studies (43%)	Design/Project scores	S2, S9, S30, S39, S59	-	-	S62
	LOC		-	S47, S25	S57
	OO Design Quality	S5,	-	S35, S25	S21
	Number of test cases passed/failed	S17, S33, S53 S60	-	S44 S49	-
	Expert opinion	S38, S52, S73	-	S13	S45
	Number of defects	S55	-	S47	S64
	NATP	S68	-	S34, S46	-
	Code coverage thoroughness	-	-	S36	-
	Standard Quality model	-	-	-	S1,
Academic Performance 16 studies (23%)	Assignment scores	S38, S39, S40, S41	-	-	S37
	Final exam scores	S48, S39, S40, S41	-	S19, S54	S37, S38
	Midterm scores	S48,	-	-	-
	Quiz scores	-	-	S19	-
	Project scores	S59	-	S20	-
	Test scores	S30, S40, S41	-	-	-
	Course grade	S15, S39, S40, S41	S24	S19	S61, S62
	Course completion rate	S12, S39	-	-	-
	Retention rate	-	-	-	S37
Satisfaction 22 studies (31%)	Pair formation, increased knowledge & confidence, positive attitude about collaboration, enjoyment, social interaction	S2, S10, S14, S16, S18, S19, S20, S21, S22, S31, S32, S40, S41, S52, S56, S57, S59, S62, S66, S70, S72, S74	-	-	S69

TABLE 4
PP'S EFFECTIVENESS (PP VS SOLO)

Comparison	Technical Productivity	Program quality	Academic Performance	Satisfaction
1) PP is better than solo	S3, S4, S6, S7, S8, S9, S18, S30, S31, S33, S51, S52, S53, S60	S2, S5, S9,S17, S30, S33, S38, S39, S53, S55, S60, S68	S12, S15, S30, S39, S40, S41, S48, S59	S2, S14, S16, S18, S19, S20, S21, S31, S40, S41, S52, S57, S62, S66, S70, S72, S74
2) PP is similar to solo	S19, S23, S25, S43, S46, S47, S57, S74	S25, S34, S35, S36, S44, S46, S47	S19, S20, S54	-
3) PP is worse than solo	S65	-	S24	-
4) Mixed-findings	S71	S1,S21, S45, S57,S64	S37, S38, S61, S62	S69

TABLE 5
SUMMARY OF QUALITY METRICS USED

Metrics' Category:	Quality Metrics(s)		Ranking*	Sig. +ve	Sig. –ve	No effect	Mixed
Internal Code Quality	**Program size**	1) Lines of code (LOC)	2	S53	-	S47	S21
		2) Non comment lines of code (NCLOC)	2	-	-	S25, S57	S21
		3) Comment Ratio (CR)	4	-	-	S25	
		4) Number of methods	4	-	-		S21
	Object Oriented Design Quality	1) Method Level (McCabe's cyclomatic complexity, and number of parameters passed to the method)	3	-	-	S57**	S21
		2) Class Level (Chidamber & Kemerer's metric suit)	4	S5	-	-	-
		3) Package level (Martin's package level dependency metrics)	4	-	-	S35	-
		4) Coupling Factor (CF)	4	-	-	S25	-
External Code Quality	1) Number of test cases passed/failed		1	S33, S53, S60, S17	-	S44, S49,	-
	2) Number of features correctly implemented		4	S21	-		-
	3) Number of incomplete requirements		4	-	-	S49	-
	4) Number of defects/errors		2	S47, S55	-		S57, S64
	5) Completion of change request		4	S52	-		-
	6) Thoroughness and fault finding effectiveness		4	-	-	S36	-
	7) Number of acceptance test passed (NATP)		2	S68	-	S34, S46	-
Standard Quality Model	1) ISO/IEC 9126 Quality Factors		4	-	-	-	S1
General	1) Programming score, project score/grade		1	S2, S9, S30, S39, S59	-	S20,	S62
	2) Expert opinions		1	S5, S16, S38, S73	-	S13	S21, S45

(*) The ranking showed the quality metrics used the most in PP studies (in ascending order)
(**) Some of the studies used more than one metric to measure the quality of a program/design.

Norsaremah Salleh is currently studying her PhD at the Department of Computer Science, University of Auckland, New Zealand. Her study is sponsored by the Ministry of Higher Education Malaysia. She holds a lecturing position at the Department of Computer Science, International Islamic University Malaysia. She obtained her MSc. in Computer Science from the Universiti Teknologi Malaysia. Before joining academic, she worked nearly five years as analyst programmer in the manufacturing industry. Her research interests are empirical studies in Software Engineering, evidence based research, CS/sE education, and object-oriented software development.

Emilia Mendes is Associate Professor in Computer Science at the University of Auckland (New Zealand). She has active research interests in the areas of Empirical Web & Software Engineering, Evidence-based research, Hypermedia, Computer Science & Software Engineering education, in which areas she has published widely and over 130 refereed publications, which include two books (one edited (2005) and one authored (2007)). A/Prof. Mendes is on the editorial board of the International Journal of Web Engineering and Technology, the Journal of Web Engineering, the Journal of Software Measurement, the International Journal of Software Engineering and Its Applications, the Empirical Software Engineering Journal, the Advances in Software Engineering Journal and the Software Quality Journal. She worked in the software industry for ten years before obtaining in 1999 her PhD in Computer Science from the University of Southampton (UK), and moving to Auckland (NZ).

John C. Grundy is a member of the IEEE and the IEEE Computer Society. He holds the BSc(Hons), MSc and PhD degrees, all in Computer Science and all from the University of Auckland, New Zealand. Previously he was Lecturer and Senior Lecturer at the University of Waikato, New Zealand, and Professor of Software Engineering and Head of Electrical & Computer Engineering at the University of Auckland, New Zealand. He is currently Professor of Software Engineering and Head, Computer Science and Software Engineering, at the Swinburne University of Technology, Melbourne, Australia. He is an Associate Editor of IEEE Transactions on Software Engineering, Automated Software Engineering journal, and IEEE Software. His current interests include domain-specific visual languages, model-driven engineering, large scale systems engineering, and software engineering education.