Educ Inf Technol
DOI 10.1007/s10639-013-9285-5

# Knowledge sharing through pair programming in learning environments: An empirical study

**R. K. Kavitha · M. S. Irfan Ahmed**



**Abstract** Agile software development is an iterative and incremental methodology, where solutions evolve from self-organizing, cross-functional teams. Pair programming is a type of agile software development technique where two programmers work together with one computer for developing software. This paper reports the results of the pair programming exercise which was carried out among fifty six post graduate students of Master of Computer Applications (MCA) who are engaged in developing small applications as a part of their Visual Programming laboratory course at Kumaraguru College of Technology (KCT) during the academic year 2012–2013. The basic intent of the study is to explore the possibilities of adopting pair programming as a pedagogical tool in higher educational setting. The study investigates the possibilities of deploying purposeful pair programming modules to facilitate knowledge sharing in regular laboratory sessions. The research findings suggest that pair programming can be a useful approach to teach computer programming in higher education to facilitate effective knowledge sharing among the students.

**Keywords** Agile software development · Pair programming · Knowledge sharing · Collaborative learning

## 1 Introduction

Agile software development promises to deliver increased productivity, quality and project success rate in software development projects by synergizing considerable tacit knowledge. Agile methods promote proactive planning and also encourage rapid

R. K. Kavitha (✉)
Department of Computer Applications, Kumaraguru College of Technology, Coimbatore, India
e-mail: rk_kavitha@ymail.com

M. S. I. Ahmed
Department of Computer Applications, Sri Krishna College of Engineering and Technology, Coimbatore, India
e-mail: msirfan@gmail.com

Published online: 09 October 2013

and flexible response to change by providing collaborative environments. Of all the various agile software, Extreme Programming (Beck 2000) is one among the widely recognized software development methodology. Extreme Programming focuses on disseminating knowledge through collaborative practices such as pair programming, planning game and retrospectives. The basic idea behind these practices is to share work and expertise across teams. Pair Programming is an extreme programming practice where two programmers mutually collaborate at the same workstation to acquire added up knowledge and experience on a daily basis. It is assumed that while programming in pairs, people try harder to code more effectively, which in due course helps to develop more confidence.

### 1.1 Pair Programming [PP]

Pair programming is practiced mostly by those institutions focusing on software development. The programmers collaborate as pairs by sharing a single computer working with the same design, algorithm, code, or test etc., (Williams and Kessler 2003). One member of the pair, known as the driver types at the computer or writes down a design. The other member who assumes the role of the navigator observes the work of the driver to ensure objectivity, logic and process flexibility (Cockburn and Williams 2002). Anecdotal evidence suggests that two programmers working in collaboration can be twice more effective in terms of speed and the possibility of finding solution when compared to two programmers working individually on their own. The effectiveness of the pair programming depends on effective and frequent brainstorming among the pair (Faja 2011). In addition, it also facilitates frequent exchange of roles between pairs. Moreover, constant code reviews, which are an inherent part of pair programming, minimize defects to a significant extent, thus resulting in the development of error-free software.

### 1.2 Academic perspective of knowledge management in computer programming laboratory sessions

An agile methodology like Extreme Programming (XP) which relies heavily upon sharing knowledge and information for holistic development of organizations is being widely used across various industries in the recent past. The knowledge management model helps the developers to share technical knowledge across the whole organization. On the contrary, dearth in information sharing has led many organizations to fail time and again. Thus, this new methodology has given a new impetus to teaching and learning software development related subjects.

The strength of knowledge sharing was felt when the students of the Master of Computer Applications (MCA) program struggled a lot initially while developing applications, owing to their different educational backgrounds. Pair programming complimented the learning process commendably within a short period of time (Williams and Kessler 2003). It also helped students to gain real time practical experience of software development through knowledge sharing and collaboration. While programming in pairs, both the partners discussed and worked on the given problem, thus sharing their experience and knowledge (Williams and Kessler 2000). Rotation of pairs led to a greater sharing of knowledge, quicker learning and more efficient use of mutual skill sets.

## 2 Existing studies on Pair Programming

Several experiments on the effects of pair versus solo programming have been reported in literature. Muller (2005) has shown that pair programmers are as cost-effective as solo programmers in writing code with fewer errors. According to Lui & Chan (2006), pair programming promotes not only quality programming skills, but also enhances responsibility, mentoring, teamwork and increased sense of passion. Jari Vanhanen and Harri Korpi (2007) have presented their experiences of using PP extensively in an industrial project. Their results shows that test-driven development and design in pairs minimized defects and improved both quality and knowledge transfer, thus proving their suitability for complex tasks.

Tanja Bipp and Andreas Lepper (2008) presented the results of an extensive and substantial case study on pair programming, which was carried out in software development courses at the University of Dortmund, Germany. Thirteen software development teams with 100 students took part in the experiments. The groups were divided into two sets with different working conditions. In one set, the group members worked on their projects in pairs. According to the study, the paired teams produced nearly as much code as the teams of individual workers within the same period. In addition, the code produced by the paired teams was easier to read and understand, thus facilitating easy detection and correction of errors.

Norsaremah Salleh (2008) reviewed 66 studies and identified certain psycho-social factors such as compatibility, personality and gender issues which affect the effectiveness of pair programming among students. V. Venkatesan and A. Sankar (2010) studied the effects of pair programming on knowledge transfer and derivation of a sense of work accomplishment in the student context. The study had two-fold objectives: (1) to analyze if task complexity has an effect on the effort invested in solo and pair programming and to check if pair programming improves knowledge transfer and enjoyment of work. The result of this experiment illustrated a significant difference both in the time line and effort to perform the task precisely in solo and pair programming. Norsaremah Salleh, Emilia Mendes, and John C. Grundy (2011) presented the effectiveness and relative evidence to adopt pair programming (PP) as a pedagogical tool in higher education CS/SE courses. The common criterion used by them to gauge PP's effectiveness was the time spent on programming. The results of their study showed that the overall students' skill level was the most significant factor that influences the effectiveness of PP. In addition, students expressed a higher level of content and satisfaction during pair programming when compared to working solo. The studies reported in literature so far, mostly involved experiments carried out for a shorter period of time. However, the current study carried out by the researcher presents the results of the outcome of a controlled experiment carried out for longer time duration of 6 months. The present study explores the knowledge sharing aspect of PP and also records the positive and negative aspects of PP, as perceived by the students.

## 3 Proposed work

Based on the researcher's experiences and insights drawn from existing literature, it was proposed to study the effects and experiences of the pair programming concept.

Initially, the factors including the usefulness of pair programming, willingness to adopt the same in the near future, contribution to taking initiatives, proactive learning and knowledge improvement were taken into consideration. The objectives of the study are:

- To test the association between the perceived usefulness of pair programming and the possibility of its adoption in future
- To test the association between proactive learning and level of satisfaction with pair programming
- To test the difference of opinion with respect to the extent and level of knowledge gained among groups
- To test the association between contradiction among pairs and the sense of accomplishment experienced by the pair programmers

## 3.1 Experimental methodology and context

In order to facilitate the students learning process in Computer Applications course, it was decided to use pair programming as a teaching pedagogy and investigate its effectiveness on students overall learning process. The students were asked to work in pairs for the entire duration of their laboratory course.

## 3.2 Survey

The survey is a technique of gathering data through questioning respondents, who are expected to provide the desired information. A formal list of questionnaire was prepared and the responses were analyzed using standard statistical techniques. Two questionnaires with close-ended questions with 5 point rating scale were designed. Initially, the students were directed to fill an entry questionnaire to assess the students' level of exposure to programming tasks, partner preferences etc. The worksheet also contained open-ended questions which would grant the choice for the students to provide their own answers in an unprompted manner thus yielding qualitative data. A close-ended questionnaire was administered to the students at the end of each session to measure the parameters related to knowledge sharing, productivity, debugging and pair programming experiences.

### *3.2.1 Student respondents*

The pair programming experiment was carried out in Visual Programming Laboratory course conducted during the fourth semester of the MCA Program. The credit for the course weighted 40 % among the other laboratory courses carried out in that semester and as per the curriculum, it doesn't have an associated theory component. The students were expected to use Microsoft Visual Studio to implement visual c++ programs owing to its good user interface. As it was assured that they do not have any prior knowledge of the software, they were asked to explore the various functionalities and to develop small applications on their own.

Fifty six second year MCA students participated in the study and it was carried out in a controlled experimental setup. Four students had expressed their willingness to

do solo programming. To begin with, an introductory presentation on pair programming was given to the students to familiarize them with the concept. The students were given the choice of choosing their pairs. In sequel, the basic features of the visual studio environment were demonstrated by the faculty. The students were given sufficient time to explore the concepts of the prescribed software by referring to e-books and lab manuals. The students who participated in the study were made to work in pairs in regular laboratory sessions. Care was taken to ensure that the pairs interchanged among themselves and shifted roles frequently to enable knowledge sharing among themselves. All the laboratory sessions were coordinated and closely monitored by two faculty members to ensure accurate recording of data.

### 3.3 The controlled experiment

In order to carry out the pair programming exercise in a systemic way, a process framework was designed (Fig. 1) with appropriate user interfaces which enabled the student respondents and the assessors to record the data precisely. Once the students were found to acquire the requisite understanding about Pair programming, they were allowed to access online software Pair Programming Information System [PPIS] which forms a part of the framework, where the questionnaire responses were stored in appropriate databases. The experiments were well structured and conducted upon with ten problem descriptions executed across separate lab sessions of three hour duration each. The student respondents were directed to analyze and write appropriate codes for the given problem. Simultaneously they were also requested to record their individual experiences for the respective sessions.

A worksheet was used to extract and record the software learning experiences of those students working in pairs. Subsequently, the details related to knowledge

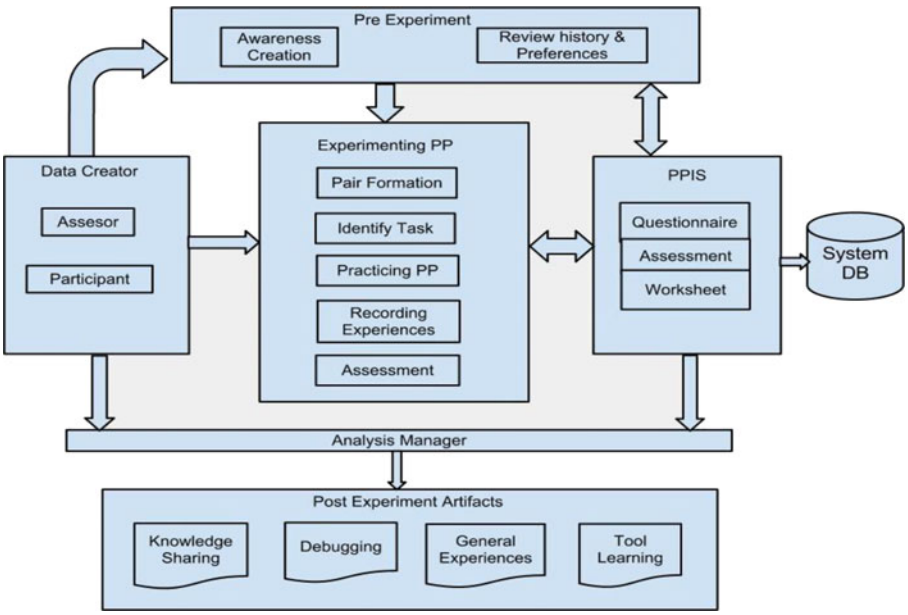


**Fig. 1** Pair programming process framework

sharing collected. On completion, the students were asked to fill an exit questionnaire, which was designed to collect their views both on the positive and negative aspects of pair programming, knowledge sharing and tool learning. Each experiment was assessed and the comments were recorded in the prescribed software. Further, the model and end-semester exam assessment scores were also analyzed to see if pair programming had an effect on improving students' performance. The data thus collected was analyzed using appropriate tools and the results of the analysis are presented in the following sections.

## 4 Analysis and interpretation of collected data

Analysis of the collected data was carried out using statistical predictive analytics software and data mining tools.

### 4.1 Students perceptions on Pair Programming

From Table 1, it can be inferred that pair programming enables sharing knowledge to a significant extent and that students are willing to adopt pair programming for future needs. The pair support rendered was also observed to be relatively high in the laboratory sessions (Fig. 2). It was also observed that during PP sessions, the students were more vigilant and exhibited an increased initiative and proactive attitude towards learning, leaving very little for chance.

Frequent contradictions were observed among the pairs while choosing the appropriate solution for a problem (Fig. 3). Distractions mainly occurred in the form of casual discussions, not connected to the work on hand. The participants also expressed that pair programming did not make them feel inhibited at any given time (Table 2).

**Table 1** Student's perception with respective to the positive aspects of pair programming

| Variables | Mean | Std. Deviation |
|---|---|---|
| Usefulness | 4.20 | .70 |
| Overall productivity | 4.07 | .69 |
| Sense of responsibility | 4.02 | .55 |
| Pair support | 4.33 | .67 |
| Knowledge improvement | 3.95 | .61 |
| Knowledge sharing | 4.41 | .69 |
| Roles switching | 4.02 | .85 |
| Levels of awareness | 3.72 | .69 |
| Increased productivity | 4.05 | .65 |
| Degree of learning | 3.75 | .81 |
| Levels of Satisfaction | 4.16 | .75 |
| Possibilities of second opinion | 3.95 | .75 |
| Work quality | 4.25 | .65 |
| Sense of accomplishment | 4.09 | .64 |
| Possibility of future Adoption | 4.34 | .78 |

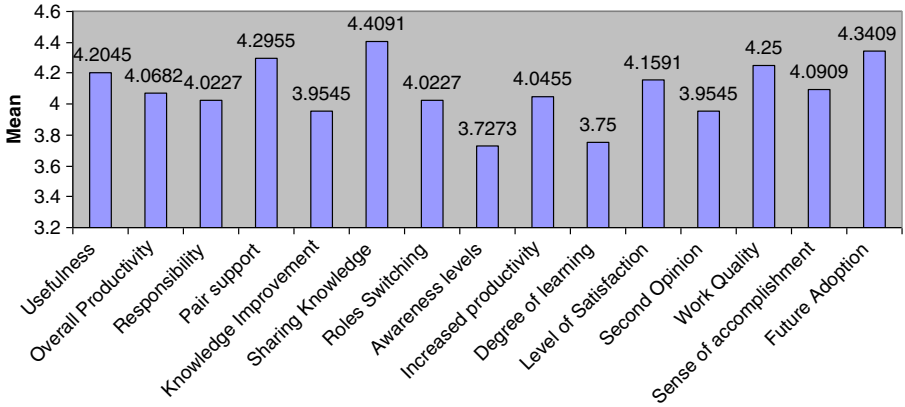


**Fig. 2** Students' perception on positive aspects of pair programming

From Table 3, it can be inferred that the total debugging time of the pair programming groups out of the total development time varies from 0 to 80 %. On an average, 22.29 % of total development was spent on debugging. It was observed that the total number of conversations among the pairs ranged between 10 and 130. On an average, around 40 conversations were recorded between them, out of which 8 were related to problem understanding, 6 were for tool usage and 11 and 12 for coding and debugging respectively. However, it is to be noted that the remaining 3 conversations were not related to the task in hand. While observing the knowledge-related aspects from the table, it can be found that 64 % of the knowledge was shared among the pairs on an average. The mean knowledge gained through pair programming was found to be 54 %.

The participants opined that pair programming helped them to master the use of software, provided the given software has a good user interface. Most of them seem to indicate that PP drastically reduced the time consumed by the lab experiment, when compared to learning by one's own trails.

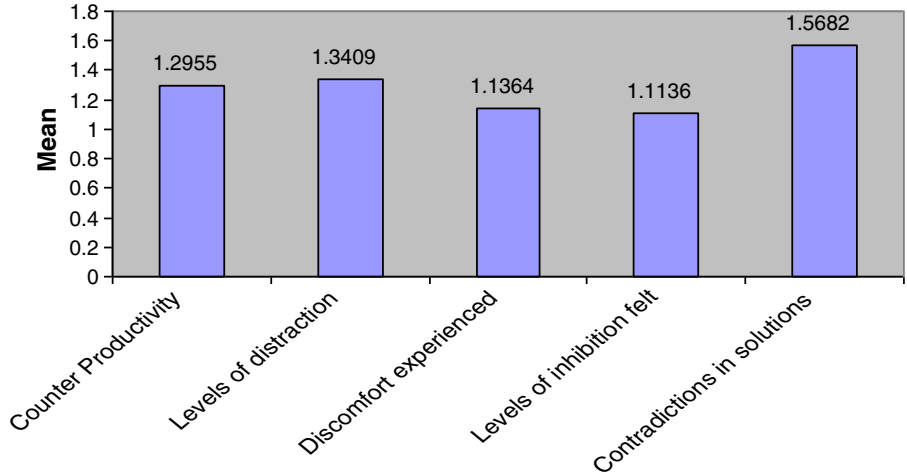


**Fig. 3** Student's perception about the negative aspects of pair programming

**Table 2** Student's perception with respect to the negative aspects of pair programming

| Variables | Mean | Std. Deviation |
|---|---|---|
| Chances of counter productivity | 1.29 | .73 |
| Levels of distraction | 1.34 | .64 |
| Discomfort experienced while Switching roles | 1.13 | .41 |
| Levels of inhibition felt | 1.11 | .32 |
| Contradictions in solutions | 1.57 | .89 |

The results of correlation are exhibited in Table 4. It can be seen that the highest correlation of 0.424 was observed for the usefulness of pair programming and possibility of future adoption and it was found to be significant. The next highest correlation was found for the variable usefulness of pair programming and the degree of learning that occurs with a correlation value of 0.419 and a significance value of 0.003. The contradictions in solutions do not seem to have any correlation with the other variables under the study such as possibility of future adoption, sense of accomplishment, level of satisfaction, degree of learning and usefulness of pair programming.

### 4.2 Association between the perceived usefulness of Pair Programming and the possibility of its adoption in future

Chi square test was used to find the association between perceived usefulness of pair programming and its possibility of adoption in future. The details of the contingency table and the chi square results are presented below in Table 5. The hypothesis formulated for the same is:

> $H_0$: There is no association between perceived usefulness of pair programming and possibility of its adoption in future.

**Table 3** Worksheet data

| | Minimum | Maximum | Mean | Std. Deviation |
|---|---|---|---|---|
| Total debugging time | .00 | 80.00 | 22.29 | 17.99 |
| Total conversations | 10.00 | 130.00 | 39.68 | 31.63 |
| Frequency of problem understanding related conversations | .00 | 20.00 | 7.55 | 5.69 |
| Frequency of tool usage related conversations | .00 | 20.00 | 6.14 | 4.47 |
| Frequency of coding related conversations | 2.00 | 60.00 | 10.66 | 12.38 |
| Frequency of debugging related conversations | .00 | 40.00 | 12.27 | 11.09 |
| Extent of knowledge shared | 30.00 | 95.00 | 64.09 | 17.40 |
| Extent of knowledge gained | 10.00 | 90.00 | 54.09 | 23.53 |
| Software Usage | 3.00 | 5.00 | 4.39 | .62 |
| Reduction in time consumption | 4.00 | 5.00 | 4.7045 | .46 |

**Table 4** Correlation between variables

| | Usefulness | Learning | Level of Satisfaction | Experience | Future Adoption | Contradiction in Solution |
|---|---|---|---|---|---|---|
| Usefulness | 1 | .419** | .381* | .268 | .424** | .254 |
| | | 0.003 | .011 | .078 | .004 | .096 |
| Learning | | 1 | .337* | .358* | .139 | −.088 |
| | | | .025 | .017 | .369 | .571 |
| Level of Satisfaction | | | 1 | .407** | .346* | −.173 |
| | | | | .006 | .021 | .262 |
| Experience | | | | 1 | .404** | −.172 |
| | | | | | .007 | .263 |
| Future Adoption | | | | | 1 | −.117 |
| | | | | | | .448 |
| Contradiction in Solution | | | | | | 1 |

** Correlation is significant at 0.01 level

* Correlation is significant at 0.05 level

The results of the chi square test indicate that the significance value is less than 0.05, thus rejecting the null hypothesis and supporting the alternative hypothesis. It can safely be assumed that there is an association between the usefulness of pair programming and the possibility of adoption in future.

## 4.3 Association between the proactive learning and level of satisfaction with pair programming

The null hypothesis for this proposition is as follows:

$H_{0:}$ There is no association between level of satisfaction and proactive learning.

**Table 5** Testing for association between perceived usefulness of pair programming and its adoption in future

| Usefulness | Possibility of future adoption | |
|---|---|---|
| | Low | High |
| Low | 19 | 11 |
| High | 4 | 10 |
| Total | 23 | 21 |

| | Value | df | Asymp. Sig. (2-sided) |
|---|---|---|---|
| Pearson Chi-Square | 4.623[a] | 1 | .032 |

Since the significance value is less than 0.05 (Table 6), the null hypothesis can be rejected. It can also be concluded that there appears to be an association between level of satisfaction and proactive learning.

### 4.4 Testing for differences in the extent of knowledge gained

It was also planned to test if there is any difference in the extent of knowledge gained between student groups which believe there is an improvement and those who believed otherwise. *T*-Test was used to test and identify this attitudinal difference between the students who underwent the experiment. However the results of the *T*-Test as presented in Table 7 seem to indicate that knowledge sharing has taken place during the experiment to a considerable extent. The data on variable improvement in knowledge was measured and obtained through dividing the groups into two by administering dichotomous questions. The data for knowledge gained was calculated as a percentage value. The null hypothesis for this proposition is as follows:

> $H_0$: There is no significant difference in the percentage of knowledge gained between the students who perceive there was knowledge improvement and who perceive there isn't any improvement.

Since the significance value is greater than 0.05, the null hypothesis was accepted over the perception of knowledge improvement and percentage of knowledge gained. Data analysis shows even if the students have felt that there is less improvement in terms of knowledge, they had gained considerable knowledge through pair programming. Thus, it can be safely concluded that pair programming has significantly improved knowledge sharing among the pairs.

### 4.5 Association between contradiction among pairs and sense of accomplishment

To find out the association between contradiction among pairs and sense of accomplishment, chi square test was used (Table 8). The details of the contingency table and

**Table 6** Testing for association between proactive learning and level of satisfaction

| Level of satisfaction | Proactive learning | |
|---|---|---|
| | Low | High |
| Low | 9 | 19 |
| High | 1 | 15 |
| Total | 10 | 34 |

| | Value | df | Asymp. Sig. (2-sided) |
|---|---|---|---|
| Pearson Chi-Square | 3.887[a] | 1 | .049 |
| Continuity correction | 2.552 | 1 | .110 |
| Likelihood Ratio | 4.518 | 1 | .034 |
| Linear-by-Linear Association | 3.799 | 1 | .051 |

**Table 7** Testing for difference in the extent of knowledge gained

| | Improvement in knowledge | Mean | Std. Deviation |
|---|---|---|---|
| Knowledge gained | No Improvement | 41.11 | 26.31 |
| | Improvement | 57.43 | 21.94 |

| | T-test for Equality of Means | | |
|---|---|---|---|
| | t | df | Sig. (2-tailed) |
| Knowledge gained | -1.714 | 11.034 | .115 |

the chi square results are presented below. The hypothesis formulated for the same is as follows:

> $H_0$: There is no association between contradiction among pairs and sense of accomplishment.

The results of the chi square testing indicated that the significance value is greater than 0.05. Thus, the null hypothesis can be accepted and it can be concluded that the experience of sense of accomplishment has no connection with the occurrence of contradictions. While choosing a solution to the problem the students seem to experience a sense of accomplishment irrespective of the presence or absence of contradictions.

### 4.6 Discovering the rules of association from the data set

Mining association rules search for interesting relationships among items in a given data set. Association rules x=>y express that the occurrence of x has a positive influence on the occurrence of y (Han and Kamber 2006). The association modeling helps not only in

**Table 8** Testing for association between contradiction among pairs and sense of accomplishment

| | | Contradiction among pairs | | |
|---|---|---|---|---|
| | | Low | High | Total |
| Sense of accomplishment | Low | 22 | 12 | 34 |
| | High | 8 | 2 | 10 |
| | Total | 30 | 14 | 44 |

| | Value | df | Asymp. Sig. (2-sided) |
|---|---|---|---|
| Pearson Chi-Square | .833[a] | 1 | .361 |
| Continuity correction | .277 | 1 | .598 |
| Likelihood Ratio | .886 | 1 | .346 |
| Linear-by-Linear Association | .814 | 1 | .367 |

predicting those items that are likely to co-occur, but also the strength of the relationship between these items. The association between the following attributes, namely usefulness of pair programming [Q1], knowledge improvement [Q5] and possibility of future adoption [Q15] were explored by applying the a priori association mining algorithm (Fig. 4). As a part of the association method, FP-Growth algorithm was applied to the data set. The graphical view of the association rules are also shown in Fig. 5.

A high level of confidence observed implies that the association rules shown above assist in arriving at certain decisions and justifying the same. From the rules derived, it can be inferred that the aspects of knowledge improvement through pair programming and the possibility of its adoption in future appear to have a positive influence on the usefulness of pair programming, with a high level of confidence at 0.97.

### 4.7 Performance of student respondents

The student respondents who participated in these experiments were selected upon a base criteria of the Cumulative Grade Point Average (CGPA) scores obtained during their previous semesters of the course and were categorized into high, average and low performers. The students were continuously monitored and assessed through two internal tests and one end semester examination with a viva component. The experiment trials conducted included problems with higher levels of difficulty compared to those that they would normally solve in regular laboratory sessions. A student respondent was assigned with two problems individually for each test component. Finally the assessors evaluated the program output and throughput for 80 and 20 marks respectively. The criteria for evaluating program output encompassed the time frame, logical sequence, requisite output and problem understanding. The evaluation was also complimented by a viva session that weighed 20 % of total marks to test the efficiency of the student respondents to explain about the tool features, programming concepts and algorithms which they have used earlier. The test results revealed that the student respondents had gained a significant knowledge about tool usage and programming tactics. In addition, it was also noted that even the presumed mediocre performers were able to complete the programmes with minimum hassles. To benchmark the performance outcomes of those students who underwent pair programming, the results of another laboratory course which adopted solo programming conducted during the same period were compared (Fig. 6). The results had indicated clearly that pair programming experiments had

```
[Q15] --> [Q1, Q5] (confidence: 0.800)
[Q15] --> [Q5] (confidence: 0.825)
[Q1] --> [Q5] (confidence: 0.829)
[Q1, Q15] --> [Q5] (confidence: 0.842)
[Q5] --> [Q1, Q15] (confidence: 0.914)
[Q1] --> [Q15] (confidence: 0.927)
[Q1, Q5] --> [Q15] (confidence: 0.941)
[Q5] --> [Q15] (confidence: 0.943)
[Q15] --> [Q1] (confidence: 0.950)
[Q15, Q5] --> [Q1] (confidence: 0.970)
[Q5] --> [Q1] (confidence: 0.971)
```

**Fig. 4** Association rules derived from the exit questionnaire data set

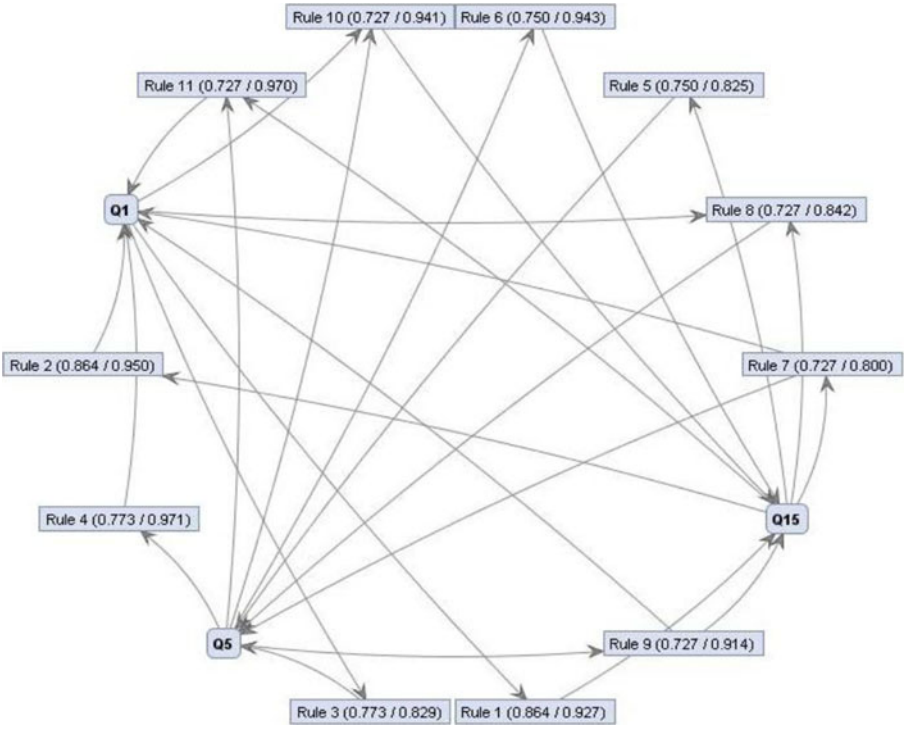


**Fig. 5** Graphical view of the association rules derived from the exit questionnaire data set

enhanced the performance level of students. To substantiate, it was recorded that even the average and mediocre students felt that pair programming sessions came handy for their better level of performance output. Few students who were involved in solo programming also expressed their willingness to adopt pair programming in future.

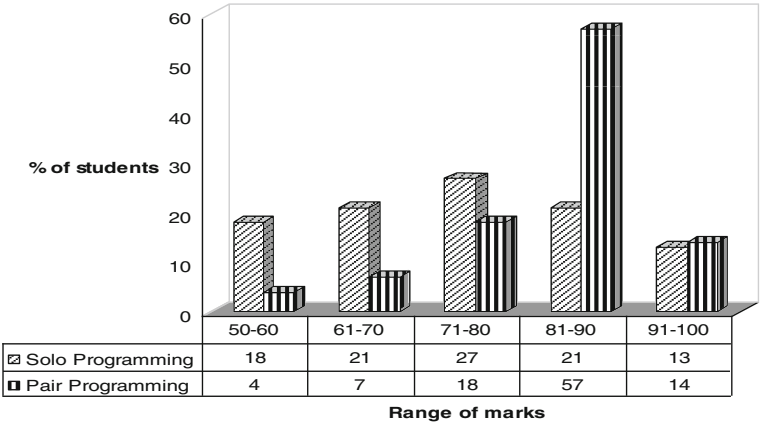


**Fig. 6** Student's performance—Solo and Pair Programming

## 5 Conclusions and scope for future work

The significance of collaborative skills has been realized more than ever in teaching and learning programming courses during recent times. Keeping this in mind, a study of pair programming in the context of facilitating knowledge sharing has been carried out. This paper presents the results of the preliminary work of implementing pair programming as a learning pedagogy for second year post graduate computer applications students. Earlier results seem to show a considerable degree of promise and they further indicate that pair programming contains a huge potential to enhance the development of both programming practice and collaborative skills within the context of a programming laboratory course. These early indicators have provided impetus to expand the benefits of an enhanced collaborative learning.

A process framework for pair programming has been developed. The experiment was conducted for the entire duration of a laboratory course and the data collected was analyzed using statistical and data mining tools. The empirical evidence shows that pair programming helps in effective knowledge sharing among the pairs. Further, this approach seems to help students in easy and quick learning of that software's with a good user interface. The examination scores of the laboratory course were also found to be commendable with around 70 % of students scoring above 80 % of marks. Majority of the students have expressed that while practicing in pairs they did experience a sense of reward and accomplishment. However, few students who worked in paired teams also expressed their concern with the team composition. As an extension of the present work, the researchers are currently working on developing software tools that would help in the evaluation and analysis of the experimental sessions. The research on pair programming can also be planned to analyze pair compatibility and styles.

**Acknowledgments** The authors would like to thank the Management of Kumaraguru College of Technology, Coimbatore, India for providing the necessary support to conduct the experiment and collect the necessary data for research. We would like to thank all the students who participated in the study and faculty friends who supported the study. This work was funded by the All India Council for Technical Education (AICTE) Ref. No. 8023/RID/RPS-61/2011-12 (Pvt).

## References

Beck, K. (2000). Extreme programming explained, Addison-Wesley.

Bipp, T., Lepper, A., Schmedding, D. (2008). Pair programming in software development teams-an empirical study of its benefits, Science Direct Information and Software Technology, 231–240.

Cockburn, A., & Williams, L. (2002). *"The costs and benefits of pair programming" in extreme programming examined*. Boston: Addison-Wesley.

Faja, S. (2011). Pair programming as a team based learning activity: a review of research. *Issues in Information Systems, 12*(2), 207–216.

Han, J., Kamber, M. (2006). Data mining: concepts and techniques, 2nd edition. USA: Morgan Kaufmann Publishers.

Lui, K. M., & Chan, K. C. C. (2006). Pair programming productivity: novice-novice vs. expert-expert. *International Journal of Human-Computer Studies, 64*(9), 915–925.

Muller, M. M. (2005). Two controlled experiments concerning the comparison of pair programming to peer review. *The Journal of Systems and Software, 78*(2), 166–179.

Salleh, N. (2008). *"A systematic review of pair programming research –initial results", Proceedings of NZCSRSC 2008* (pp. 151–158). New Zealand: Christchurch.

Salleh, N., Mendes, E., & Grundy, J. C. (2011). Empirical studies of pair programming for CS/SE teaching in higher education: a systematic literature review. *IEEE Transactions on Software Engineering, 37*(4), 509–525.

Vanhanen, J., Korpi, H. (2007) "Experiences of using pair programming in an agile project", Proceedings of the 40th Hawaii International Conference on System Sciences IEEE Computer Society, 1530–1605.

Venkatesan, V., & Sankar, A. (2010). Adoption of pair programming in the academic environment with different degree of complexity in students perspective– an empirical study. *International Journal of Engineering Science and Technology, 2*(9), 4791–4800.

Williams, L. A., Kessler, R. R. (2000). All I ever needed to know about pair programming I learned in kindergarten, Communications of the ACM. New York: Association for Computing Machinery (ACM).

Williams L., Kessler R. R. (2003). Pair programming illuminated. London: Addison-Wesley.