# Using a Collaborative Learning Technique as a Pedagogic Intervention for the Effective Teaching and Learning of a Programming Course

**Dr Desmond W. Govender**

*University of KwaZulu-Natal, Edgewood Campus, South Africa, Email: govenderd50@ukzn.ac.za*

**Mr. T.P Govender**

*Lecturer: Department of Information Technology, Faculty of Accounting and Informatics,Durban University of Technology, Address: 41-43 ML Sultan Road, Durban, 4001, KwaZulu-Natal, South Africa Email: prinaving@dut.ac.za*

**Doi:10.5901/mjss.2014.v5n20p1077**

***Abstract***

*Educators are faced with ever increasing challenges when teaching programming. The increase in the number of operating systems brings with it challenges for programmers because of the change in programming paradigms, programming languages and software suites that are dependent on most recent developments in technology and more so operating systems. The issue for educators maybe to move away from teacher centred teaching and learning to student centred learning. Pair programming is a technique that offers educators an opportunity to further enhance student centred learning. This study conducted an empirical study of "pair programming" in the teaching and learning of an introductory programming course in computer science with input from educators and learners. The purpose was to determine how a collaborative learning technique can be used as a pedagogic intervention for effective teaching and learning of a programming course. The study attempted to determine the impact of collaborative pair programming on students and whether Information Technology educators can use pair programming as a teaching strategy. There was a pre-test for students to secure data on how students attempted programming tasks. Thereafter the pair-programming technique was implemented and a post-test was administered to determine the effectiveness of the intervention strategy. The research findings indicated that the educators and learners had a positive attitude towards the use of pair programming to support teaching and learning and learners were convinced that they would become better programmers in the future.*

***Keywords:*** *Pair Programming, Teaching Programming, Collaborative teaching strategy, Teaching and Learning.*

## 1. Introduction

One of the ultimate outcomes of information technology (IT) education, is to ensure good pedagogical quality and gainful employment by the IT student in the software industry. Every day there are an infinite number of ideas and software projects that industry and academia can conjure up; however, there must be an association between what academia conjures up and what happens in the software industry. Software projects researched by Bryant, Romero, and du Boulay (2006) encouraged PP amongst software developers. Williams, Robert, Cunningham, and Jeffries (2000) , validated anecdotal and qualitative assumptions that pair programming (PP) software products can be produced in less time, with higher quality than programs done by a single individual.

Over the years debates have persisted over the correct approach to teach Computer Science programming and the programming languages that should be used. According to Roy and Haridi (2003), the most popular approach to teach programming is in a single paradigm embodied in a single language. Cooper, Dann and Pausch, (2003) advocate an objects-first approach, while Howe, Thornton, and Weide (2004) consider the object-oriented (OO) and the component-first approaches to be most influential. Govender (2006) indicated in her research that there "exists tension between procedural paradigm and OO paradigm". Ismail, Ngah, and Umar (2010), however, maintain that an OO approach to programming is "not a good starting-point for introducing students to the basic concepts of programming".

While the debate rages on as to the challenges in higher post-school education and the insufficient levels of resources, examination results continue to diminish and dropout rates among first-year students increase (South Africa, 2012). It is for these reasons that we try to ascertain what techniques would assist in improving learners' ability to program and thus throughput rates.

## 2. Literature Review

Programming is a challenging task and programming courses are generally considered problematic by many learners (Govender, 2006; Macgregor, 2007; Havenga & Mentz, 2009). Cooperative learning, and more importantly PP, has

opened up new possibilities for learning programming. Collaborative learning lies at the centre of human development, motivated by the desire for education that seeks to construct knowledge for the learning society (Education, 2003). Learning and the construction of knowledge have of late brought about new concepts, for example the introduction of mobile online learning, which in turn have given rise to changes in educational objectives.

The effective use of PP in support of classroom teaching by integrating technology with an appropriately considered pedagogical approach was central to teaching and learning programming in this study. Research supports the premise that collaboration is an effective pedagogy for introductory programming (McDowell, Werner, Bullock, & Fernald, 2002); Cliburn (2003); DeClue (2003). The collaborative learning research literature identifies cooperative behaviour as students discussing problems together and correcting any misconceptions or mistakes, and it has been identified as a one of the five critical attributes common to successful collaborative learning (Davidson, 1994).

Vandegrift (2004) states that "PP is a form of collaborative learning in which groups consisting of only two members – a driver and a navigator – work together on the same computer to complete the same project". Hanks, McDowell, Draper & Krnjajic (2004) further elaborated that "Each member also has individual responsibilities and roles to perform". Preston (2005) observed students, who worked in pairs, and found that providing feedback did not appear to be common practice in PP; this author therefore concluded that feedback should be included in cooperative behaviour and performance of the roles of navigator and driver. PP is an application of collaborative learning (Preston, 2005).

Williams, Wiebe, Yang, and Miller (2002) however, loosely define PP and collaborative programming interchangeably: “Pair or collaborative programming is where two programmers develop software side by side at one computer”.

There has been a decade of research into PP and its usefulness and effectiveness in both academic and industrial settings (Hannay, Dybå, Arisholm, & Sjøberg, 2009 ;Salleh, Mendes, Grundy, & Burch 2010). Beck (1999), however, claims that the history of PP stretches back to punch cards. Researchers at Microsoft Begel and Nagappan et al. (2008) found that most research focused on academic environments, and there were limited studies about PP in industry. This study has focused on academia, i.e. secondary and tertiary institutions of learning; however, academics must always bear in mind that the ultimate client of programming is the software industry.

As far back as 1978 there was a conference on programming languages. The Conference on History of Programming Languages ,Wexelblat (1978) described the 13 computer programming languages present at the time. This study does not focus on programming languages, but rather on a strategy that a teacher can adopt to teach a particular programming language. Although the history of PP stretches back to punch cards, it emerged as a viable approach to software development in the early 1990s when it was noted by Williams et al (2000) “as one of the 12 key practices promoted by extreme programming”. Swamidurai and Umphress (2012) claim that PP “has been widely accepted as an alternative to traditional individual programming”.

In the past popular paradigms included imperative programming, functional programming, logic programming and concurrent imperative programming (Van Roy & Haridi 2002). Empowering learners to become self-sufficient has always been and will always be a primary objective of education, and PP emerges among a number of recent innovations in the field of teaching and learning programming that have the potential to assist in this. Pair programming is often used in professional software development communities and appears in commercial training environments as well as in some undergraduate and high school classrooms (Nagappan et al., 2003). Chong and Hurlbutt (2007) noted that PP is a common professional practice in the workplace. When learners programme in isolation they might not experience collaborative solving of problems, communicating their understanding verbally, and resolving disagreements with their peers, all vital skills in the world of work. If one has to correctly assume that one of the primary aims of an education is gainful employment, then introducing PP at institutions of learning is preparation for the world of work. Williams et al. (2000) stated that among the benefits of PP are increased self-confidence and an interest in IT. Breed et al. (2013) research indicated that meta-cognitive skills used during PP can result in increased knowledge productivity.

Ismail, Ngah and Umar (2010) state that the main cause of difficulty in understanding programming and coding is the ‘inactive involvement’ of students during programming tutorials. Nosek (1998) found that PP outperformed individual programmers and that code from individual programmers had more errors than code programmed by PP. A similar result was found in this study. Braught, Eby, and Wahls (2008) also found that students with low standardised test scores in Mathematics showed significant improvements in individual programming skill when enrolled in classes that use PP and it was also found that drop-out rates decreased and academic achievement was enhanced. Similarly, Nagappan et al. (2003) showed that a benefit of PP was improved course completion rates. Furthermore, McDowell et al,(2002) noted that improving student experiences in introductory classes also increased the likelihood that students will continue taking more advanced classes in computer programming.

## 3. Problem Orientation

The rationale for this study was that students perceived programming to be difficult (Koorsse, Calitz, & Cilliers, 2010) coupled with the increased dropout rate among programming students and decreased throughput rate of programming qualifications (MacGregor, 2007). The University of South Africa (UNISA) is a distance, correspondence university and in 2007 it announced that it would spend nearly R50 million in intervention strategies to support its programmes (MacGregor, 2007). Peer-to-peer learning was one intervention strategy conducted in an informal setting at regional learning centres, aimed at improving the pass rate and reducing the student dropout rate. Kinnunen and Malmi (2006) indicated that many institutions report dropout rates of 20-40% or even higher in introductory programming courses. Furthermore, there is overwhelming evidence (Cooper et al., 2003; Govender 2006; Havenga & Mentz, 2009; Kolling & Rosenberg, 2001; Okur 2006) that supported the fact that both students and educators considered programming difficult to learn and teach, and that an intervention strategy was required to improve students' pass rates. This bears testimony to the fact that educators and academics are seriously concerned about the high student dropout rate and dwindling numbers of new students enrolling for Computer Science programming courses. The declining graduation numbers in the Computer Science discipline has also been documented by several authors (Howles, 2009; McKinney & Denton 2004; Ventura & Ramamurthy 2004).

## 4. Purpose of the Study

The purpose of this study was to determine how a collaborative learning technique could be used as a pedagogic intervention for effective teaching and learning of an introductory programming course at secondary schools and institutions of higher learning within KwaZulu-Natal. The study aimed to determine the impact of collaborative PP on students, and whether IT educators could use PP as a teaching strategy. The 2 research questions that were addressed were:

1. What are learners' experiences of solving programming tasks?
2. How does PP enhance problem solving in programming?

## 5. Design and Methodology

This study used a mixed-methods approach, where both qualitative and quantitative methods were used to address the research questions. Duffy (1987) summarised the relationship between quantitative and qualitative research: "quantitative research is used to evaluate objective data consisting of numbers, while qualitative research deals with subjective data that are produced by the minds of respondents." Qualitative research methods involve collecting textual or verbal data and observation of people followed by careful description and analysis (Boeree, 2008). For this study the data from the interviews and observations were analysed using qualitative research methods. Quantitative methods were employed in analysis of data collected from the questionnaires. Development of the questionnaire was guided by an extensive review of the literature. The questionnaire included questions eliciting the basic views of respondents towards computer programming in the classroom, their experiences of cooperative learning strategies and actual support for PP.

### *5.1 Sampling and data collection*

The participants were Grade 11 and Grade 12 IT learners from secondary schools, and IT students from a university of technology and from a university. In order to obtain data that would further the aims and strengthen this study, it was decided that it was necessary to use more than one data collection method. The study used interviews, questionnaires and observations. The researcher observed the learners programming in pairs, and learners complete the pre-test and post-test questionnaires. Educators were interviewed before and after the intervention. The observations, interviews and questionnaires enabled the researcher to provide answers to the research questions.

### *5.2 Data analysis and interpretation*

Data analysis involved organising, analysing and interpreting data (McMillan & Schumacher, 1993). Data from the questionnaires and observation sheets were captured on a spreadsheet as numerical data, which facilitated statistical representation in percentages and graphs. The data from the interviews were transcribed, and once the transcripts were completed the researcher looked for themes and categories that were associated with the theoretical framework, guided

by the research questions.

## 6. Results and Discussion

In terms of pre-test questionnaires, 222 were administered and collected. All participants were also required to complete the post-test questionnaire and to rate themselves as computer programmers after the intervention strategy, One hundred and ninety one (191) post-test questionnaires were collected. The overall reliability score for the ordinal section that comprises this construct was 0.667, which is close to the acceptable value of 0.70. This implied that the respondents scored the construct consistently.

With regard to gender, nearly two-thirds (62.6%) of the sample was male, however, the issue of gender was outside the scope of this study. Approximately half of the students were at first-year level. Even though IT students at third-year level completed the questionnaires, they were in effect introductory programmers who had no programming experience. A little more than two-thirds (68.5%) indicated that they did not have any programming experience, while approximately one-third (31.5%) of the respondents indicated that they had previous programming experience. Half of this grouping was at first-year university level. In total, half of the respondents who did not have programming experience were at university. Those in second and third year who indicated that they had no previous programming experience had referred to programming experience at school.

Of the school learners, 18.3% indicated that they had not had programming experience, and the equivalent value for the university student was 50.2%. In total, 68.5% of respondents indicated that they had not been exposed to programming prior to doing a computer subject. Java appeared to be the most common language across the sample population. On average almost 74% of respondents were exposed to Java.

Initially, in the pre-test, a little more than half (56.1%) of the respondents indicated that they believed that they were good programmers. A small percentage (3.6%) rated themselves as being advanced. The remaining respondents indicated that they were experiencing difficulty. In the post-test more than two-thirds of the respondents indicated that they believed that they were good programmers (68.6%). A small percentage (8.3%) rated themselves as being advanced and the balance were still experiencing some difficulty.

The pre-test data indicated that only about one-fifth (19.9%) of respondents preferred working alone. More school learners preferred to work alone (24.1%) than university students (18.4%). Overall similar numbers of respondents indicated that they would be comfortable with working in a group or with a partner. More university students preferred working in a group, whilst more school learners preferred working with a partner. Again, in the post-test only about one-fifth (21.6%) of the respondents preferred working alone. Almost 80% of the learners appeared to be enjoying programming. This augers well for the overall performance on the course and correlates to the 'likeability factor' of the course.

At least 61.1% of the respondents in the pre-test indicated that they would consult with the teacher and their classmates; however, when posed with the identical question in the post-test, only 40.1% of respondents indicated that they would request the educator's assistance. It is evident that PP reduced educator workload and consultation time as per the literature.

Only 1.1% of respondents indicated that 'team work is encouraged'. As IT educators, who require students to program individually during the teaching and learning process, should we not take our cue from the software industry which requires programmers to program in pairs/teams, and hence boost our IT learning outcomes and pass rates?

### *6.1 Thematic analysis*

The themes emerged from the pre- and post-test questionnaires, classroom observations and interviews with the educators, and were as follows: self-rating of programming, programming experience, and likeability/enjoy ability of programming. The educator's role referred to the processes of implementing, maintaining and ultimately terminating the PP strategy in the classroom and the student's role referred to the self-management of paired programming and how it influenced the successful adoption of the PP strategy.

#### *6.1.1 Self-rating of programming*

All participants were asked to rate themselves as computer programmers. Table 1 indicates the self-rating of the respondents in the pre-test.

**Table 1 :** Self-rating of respondents, pre-test (%)

| | Group | | |
|---|---|---|---|
| | School learners | University students | Overall |
| Advanced | 3.4 | 3.7 | 3.6 |
| Good | 54.2 | 56.8 | 56.1 |
| Struggling | 42.4 | 39.5 | 40.3 |

A little more than half of the respondents indicated that they believed that they were good programmers (56.1%). A small percentage (3.6%) rated themselves as being advanced. The remaining (40.3%) indicated that they were experiencing difficulties with programming. They felt that programming was complicated and that they had insufficient knowledge; this was, amongst others, the main cause for experiencing difficulties in programming. Table 2 indicates the self-rating of the respondents as a computer programmer from the post-test results.

**Table 2:** Self-rating of respondents, post-test (%)

| | Group | | |
|---|---|---|---|
| | School learners | University students | Total |
| Advanced | 0.0 | 9.6 | 8.3 |
| Good | 73.1 | 68.0 | 68.6 |
| Struggling | 26.9 | 22.5 | 23.0 |

More than two-thirds of the respondents indicated that they believed that they were good programmers (68.6%); in the pre-test an average of 55% did so, so this indicates a definite increase. A small percentage (8.3%) rated themselves as being advanced. The remaining respondents (23%) indicated that they were experiencing difficulties with programming, which is a decrease from the pre-test figure of 40.3%. When comparing whether respondents achieved their programming solutions by working on their own or within a paired partnership, a similar result was achieved. Almost 60% of respondents indicated that they often achieved their programming solutions when working on their own; similarly, in the post-test almost 58% indicated that they did so whilst working with a partner. Table 3 indicates the advantages of using a paired partner to program.

**Table 3:** Advantages of using a paired partner to program

| Statement | Rating | % |
|---|---|---|
| We achieve our programming solution | Often | 58 |
| When I write programs with a fellow student the programs are of a higher quality | Often | 54 |
| Our programs usually have syntax, run time and logic errors | Seldom had | 54 |
| We usually obtain a programming solution within the allocated time period | Often | 47 |
| Working as a pair we share each other's frustration when we are unable to successfully compile our program | Always | 49 |
| We have the necessary problem-solving skills to find a solution to a programming task | Often | 56 |

Of particular importance was the statistic that 54% of the respondents felt that their programs created using PP were of a higher quality and had fewer syntax and logical errors. During the post-test respondents indicated that they could share their frustrations when faced with a non-compiling program or inability to grasp a particular programming concept; in contrast, when programming on their own respondents only had their educator to re-explain the programming concept or to find the error in their non-compiling error-prone program.

*6.1.2 Programming experience*

A little more than two-thirds (68.5%) of respondents indicated that they did not have a programming background, while approximately one-third (31.5%) indicated that they had previous programming experience. This grouping included those in their first year at university. In total, half of the respondents who did not have programming experience were at university. Those in second and third year who indicated that they had no previous programming experience referred to experience gained at school. Of the school learners, 18.3% indicated that they did not have programming experience; the equivalent value for the university students was 50.2%. In total 68.5% indicated that they had not been exposed to

programming prior to doing a computer subject.

On average, 34% stated that they had no programming experience. This statistic had the following effects on PP:

- The more experienced programmers tended to dominate the PP experience.
- Educators found it difficult to pair programmers because of differences in programming abilities and experience.
- In PP the weaker, slower and more inexperienced programmers did benefit from working with more experienced programmers.

*6.1.3 Likeability of programming*

Figure 1 shows percentage responses to 'Do you like to solve programming problems by working alone or within a group or with a partner?'

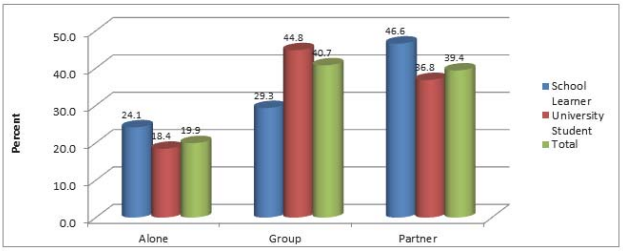


**Figure 1:** Pre-Test : Percentages of respondents who preferred working alone, in a group or with a partner.

Overall, only about a fifth (19.9%) of respondents preferred working alone, with more school learners who preferred to work alone (24.1%) than university students (18.4%). Overall, similar numbers of respondents indicated that they would be comfortable with working in a group or with a partner. More university students preferred working in a group, whilst more school learners preferred working with a partner. Table 4 indicates some of the most popular reasons given for working with a partner.

**Table 4:** Reasons given for working with a partner

| | School learners | University students | Total |
|---|---|---|---|
| Different views provided by others | 27.3 | 37.7 | 35.0 |
| Mistakes are easy to detect | 9.1 | 10.7 | 10.3 |
| Improves understanding and communication | 9.1 | 8.2 | 8.4 |
| Share knowledge and problem-solving skills | 9.1 | 14.5 | 13.1 |

Figure 2 indicates percentage of post-test responses to the question "Do you like to solve programming problems by working alone or within a group or with a partner"?

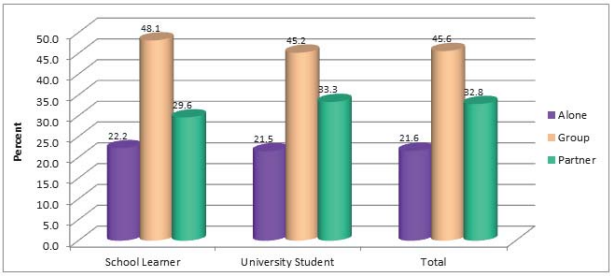


**Figure 2**: Post-test – 'Do you like to solve programming problems by working alone or within a group or with a partner?'

Only about one-fifth (21.6%) of respondents preferred working alone overall. The most popular reasons given for working with a partner were that they could "Share different views and ideas, and that mistakes were easy to detect". One student summarised his experiences with group work versus PP in this rather apt quotation: "Easier to work with partner, groups can become chaotic".

*6.1.4 "Enjoy ability" of programming*

Responses from the pre-test as to whether the respondents enjoyed programming indicate that almost 89.1% did so. Reasons given for this enjoyment indicated that 35.1% found programming fun and that it challenged their thinking, while 14.2% said that it created new ideas and skills. Table 5 indicates the level of agreement for respondents enjoying programming with a partner in the post-test.

**Table 5.** Respondents who indicated that they enjoyed programming with a partner in the post-test (%)

| | Group | | Total |
|---|---|---|---|
| | School learners | University students | |
| Yes | 84.6 | 87.1 | 86.8 |
| No | 15.4 | 12.9 | 13.2 |

The figures indicate that there were similar levels of agreement between the learners and the students. Overall, the level of agreement with the statement was high. Some of the popular reasons given for enjoying programming with a partner are indicated in Table 6.

**Table 6:** Popular reasons given for enjoying PP (%)

| | Total |
|---|---|
| Assist each other | 24.7 |
| Share knowledge | 16.5 |
| Different opinions are helpful | 7.7 |
| Learning experience is easier | 7.1 |
| Share work, which makes it easier to find a solution | 7.1 |

Other reasons that were also cited were that they enjoyed hearing different styles of coding programs, enjoyed teamwork, learned from each other, and found that it improved understanding and was fun and enjoyable. This is in agreement with the literature reviewed.

The two predominant reasons involved assisting each other (24.7%) and sharing of knowledge (16.5%). It was observed that because of the fact that students enjoyed programming, they were prepared to spend more time on a programming task and this eventually led to their obtaining a solution to it.

## 7. Conclusion

The objective of this study was to investigate the use of PP to support teaching and learning of an introductory programming course, and to provide learners and educators at schools and tertiary institutions with a strategy that they could adopt when teaching and learning introductory programming courses. The findings which emanated from the study are now summarised according to each of the research questions.

Research question one**:** What are learners' experiences of solving programming tasks?

The pre-test and post-test questionnaires were used to determine learners' attitudes towards programming. Learners were asked in the questionnaire whether they perceived PP to be useful to them, and responses showed strong agreement. The overall perception from the questionnaire was that learners felt that PP was a useful learning technique to assist with programming concepts. The majority of learners perceived that the use of PP improved their programming abilities. The data showed that learners felt that PP was a positive strategy and would unreservedly embrace its use.The positive responses to the use of PP and data from the pre- and post-test suggest that overall learners and educators were confident to implement the concepts of PP. Responses to the questionnaires and interviews showed that overall PP was easy to implement and likely to be more widely adopted by learners and educators. A majority of the learners indicated the PP was one the strategies that they would adopt when confronted by difficulties in programming tasks. The

learners' attitudes towards the paired partner positively and significantly affected their intention to use PP strategies. The study revealed that use of PP is not a current practice at secondary school level, or in introductory programming courses at university.

The findings recommended that educators could implement PP and that it would enhance learning of programming concepts. This confirmed that PP would be suitable to support learning. The post-test elicited data from learners about how programming with a partner contributed to their learning. Findings suggested that learners associated higher levels of enjoyment with programming with a partner than with programming in solitary. Observational data suggested that the male learners preferred and were more likely to select and program with an attractive female programmer. Post test data suggested that learners were extremely confident that availability of resources, paired partners, educator support enhanced their learning of programming concepts. Interview data from the educators complemented these findings. It was noted from educators that educator resources, knowledge of implementing PP, and class sizes enhanced the educators teaching experience.

Research question Two: How does PP enhance problem solving in OO programming?

One of the findings was that the learners enjoyed discussing problem strategies with their paired partner instead of with their educators. Observations and data from the post-test showed evidence of discussions that took place between the learners in a PP environment. Learners mentioned that they consulted with the educator only after they experienced difficulties that they could not solve as a pair. Often they were able to solve the problem amongst themselves. This is reassuring, because some learners felt afraid of asking questions in class but are more comfortable communicating to a paired partner.

The following benefits were derived from the learners' interaction with PP: improved communication (collaboration); interactive participation amongst peers; enhanced accessibility of resources; a supportive and non-favourable setting; and enhancement of collaborative learning. The advantage of the pairing is that the learner can discuss their experiences with their paired partner before consulting their educator, enhancing independent learning and problem solving. Learners found it easy to identify with a paired partner from a similar linguistic group because of similarity in cultural background, which also resulted in improved problem solving. Participating in the PP scenario also allowed for interactive participation. Learners were actively involved in the PP session, whether it was correcting their paired partner's mistake, accessing a secondary resource, or simply quietly observing the programming skills of an experienced partner. Since PP allowed learners to switch roles, it enhanced the collaboration between paired partners. The findings of this study showed that the learners found PP an easy, efficient and enjoyable way to learn problem-solving techniques.

Previously most knowledge was gained through the 'all knowledgeable' educator who 'spoon-fed' an ignorant learner. Today the incredible power of the Internet coupled with advancements in handheld PC tablets is creating challenges to educators and learners alike. Information is virtually at their fingertips and IT educators must find novel teaching strategies to keep the programming students of today interested and eager to solve programming tasks. One such strategy is PP. PP has a place in the IT classroom. PP provides an innovative way to complement the traditional learner-educator interaction. The use of PP will create opportunities for collaboration and independent learning, and meet the needs of all learners in various stages of their learning. PP provides a collaborative learning environment and a teaching strategy to cater for the programming needs of IT learners and educators of today, encompassing a multi-faceted (holistic) view of learning. Introduction of the use of PP in our schools and tertiary institutions to support teaching and learning of programming will not only create new possibilities for our learners to engage in new ways of learning, but also provide them with job-related software development skills currently being used in the software development industry. At a simplistic level it may even provide educators with another teaching strategy that can be adopted.

## 8. Acknowledgements

The data for this study was collected and analysed by Mr. P. Govender a master's student of Dr. D.W. Govender.

## References

Beck, K. (1999). Extreme Programming Explained: Embrace Change, Reading, PA: : Addison-Wesley.

Boeree, G.C. (2008). Personality Theories: An Introduction. Retrieved April 12, 2008, from http://webspace.ship.edu/cgboer/prsonalityintroduction.html

Braught, G., Eby, L.M. & Wahls, T. (2008). The effects of pair-programming on individual programming skill. Paper presented at the ACM SIGCSE Bulletin.

Breed, B., Mentz, E., Havenga, M., Govender, I., Govender, D., Dignum, F., & Dignum, V. (2013). Views of the Use of Self-directed

Metacognitive Questioning during Pair Programming in Economically Deprived Rural Schools. African Journal of Research in Mathematics, Science and Technology Education, 17(3), 206-219.
Bryant, S., Romero, P., & du Boulay, B. (2006). The collaborative nature of pair programming Extreme programming and agile processes in software engineering (pp. 53-64): Springer.
Chong, J., & Hurlbutt, T. (2007). The social dynamics of pair programming. Paper presented at the Software Engineering, 2007. ICSE 2007. 29th International Conference on.
Cliburn, D.C. (2003). Experiences with pair programming at a small college. Journal of Computing Sciences in Colleges, 19(1), 20-29.
Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. Paper presented at the ACM SIGCSE Bulletin.
Davidson, N. (1994). Cooperative and collaborative learning: An integrative perspective. Creativity and collaborative learning: A practical guide to empowering students and teachers, 13-30.
DeClue, T.H. (2003). Pair programming and pair trading: effects on learning and motivation in a CS2 course. Journal of Computing Sciences in Colleges, 18(5), 49-56.
Duffy, M. (1987). Methodological triangulation: a vehicle for merging quantitative and qualitative research methods. Journal of Nursing Scholarship, 19(3), 130-133.
Education, Department of. (2003). Draft White Paper on e-Education: Transforming Learning and Teaching through Information and Communication Technologies (ICTs). Government Gazette(26734), 1-44.
Govender , I. (2006). Learning to program,learning to each programming:pre-and-inservice teachers' experiences of an object-oriented language. (PHD), Unisa, Pretoria.
Hanks, B., McDowell, C. Draper, D. & Krnjajic,M. (2004, 28-30 June ). Program quality with pair programming in CS1. Paper presented at the 9th Annual Conference on Innovation and Technology in Computer Science Education.
Hannay, J., Dybå, T., Arisholm, E., & Sjøberg, D. (2009). The effectiveness of pair programming: A meta-analysis. Information and Software Technology, 51(7), 1110-1122.
Havenga, M., & Mentz, E. (2009). The school subject Information Technology: A South African perspective. Paper presented at the South African Computer Lecturers Assocation Conference (SACLA), Mpekweni Beach Resort, South Africa.
Howe, E., Thornton, M., & Weide, B. W. (2004). Components-first approaches to CS1/CS2: principles and practice. Paper presented at the ACM SIGCSE Bulletin.
Howles, T. (2009). A study of the attrition and the use of student learning communities in the computer science introductory programming sequence. Computer science education, 19(1), 1-13.
Ismail, M. N., Ngah, N. A., & Umar, I., N. (2010). Instructional Strategy in the teaching of computer programming:a need assessment analyses. TOJET: The Turkish Online Journal of Educational Technology, 9(2).
Kinnunen, P. & Malmi, L. (2006). Why students drop out CS1 course? . Paper presented at the International Workshop on Computing Education Research, New York.
Kolling, M. & Rosenberg, J. (2001). Guidelines for Teaching Object Orientation with Java. Paper presented at the 6th conference on Information Technology in Computer Science Education, Canterbury.
Koorsse, M., Calitz, A.P. & Cilliers, C.C. (2010). Programming in South African Schools: The Inside Story. Paper presented at the South African Computer Lecturers Association (SACLA), University of Pretoria.
Macgregor, K. (2007). South Africa:Student drop-out rates alarming. 0003. 2010, from http://www.universityworldnews.com/article.php?story=20071025102245380&mode=print [Accessed: 2009-05-29]
McDowell, C., Werner, L., Bullock, H., & Fernald, L. (2002). The Effects of Pair-Programming on Performance in an Introductory Programming Course. Paper presented at the SIGCSE'02 . Covington, Kentucky, USA.
McKinney, D. & Denton, L.F. (2004). Houston, we have a problem: There's a leak in the CS1 affective oxygen tank. ACM SIGCSE Inroads Bulletin, 36(1), 236-239.
McMillan , J.H. & Schumacher, S. (1993). Research in Education. A Conceptual Introduction (third ed.). New York: Harper Collins College Publishers.
Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. & Balik, S. (2003). Improving the CS1 experience with pair programming. Paper presented at the ACM SIGCSE Bulletin.
Nosek , T. (1998). The Case for Collaborative Programming. Communications of the ACM, 105-108.
Preston , D. (2005). Pair programming as a model of collaborative learning: a review of the research. Journal of Computing Sciences in colleges, 20(4), 39-45.
Roy , P., & Haridi, S. (2003). Teaching Programming Broadly and Deeply: The Kernel Language Approach. In L. Cassel & R. Reis (Eds.), Informatics Curricula and Teaching Methods (Vol. 117, pp. 53-62): Springer US.
Salleh , N., Mendes, E., Grundy, J. & Burch, J. (2010). An empirical study of the effects of conscientiousness in pair programming using the five-factor personality model. Paper presented at the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1.
South Africa. (2012). The Green Paper for Post School Education and Training. South Africa:Retrieved from http://www.dhet.gov.za/LinkClick.aspx?fileticket=w0qJyEiFVYQ%3d&tabid=189&mid=483
Swamidurai, R. & Umphress, D. (2012). Collaborative-Adversarial Pair Programming. ISRN Software Engineering, 2012, 11. doi: 10.5402/2012/516184
Van Roy , P, & Haridi , S. (2002). Concepts,Techniques, and Models of Computer Programming.

Vandegrift, T. (2004, 3-7 March 2004). Coupling pair programming and writing : Learning about students'perceptions and processes. . Paper presented at the 35th SIGCSE Technical Symposium on Computer Science Education

Ventura, P. & Ramamurthy, B. (2004). Wanted: CS1 students. No experience required. ACM SIGCSE Inroads Bulletin, 36(1), 240-244.

Wexelblat, R. L. (1978). History of programming languages I: ACM.

Williams , L.K., Robert R, Cunningham , W., & Jeffries , R. (2000). Strengthening the case for pair programming. Software, IEEE, 17(4), 19-25.

Williams , L., Wiebe , E., Yang, K. F. M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. Computer Science Education, 12(3), 197-212.