The effect of incorporating cooperative learning principles in pair programming for student teachers

E. Mentz, J.L. van der Walt and L. Goosen

Based on their quantitative and qualitative investigations, the authors conclude that pair programming as a strategy for teaching student teachers could be made more effective through the incorporation of principles associated with cooperative learning. They substantiate this claim by referring to a literature study about the advantages and disadvantages of pair programming as a teaching-learning strategy, by then discussing five principles of cooperative learning, and by presenting the findings of their empirical study. Second year student teachers taking a Delphi programming module participated in an experiment conducted over a two year period. In 2005, the participants did computer programming in pairs without the application of principles associated with cooperative learning. In 2006, a similar group of participants also programmed in pairs, but in their case, certain principles associated with cooperative learning were incorporated in the strategy followed by the facilitator. According to a comparison of the module examination marks, the 2006 group outperformed the 2005 group. This finding was confirmed by qualitative investigations.

Keywords: pair programming; collaborative learning; cooperative learning; positive interdependence; computer programming

Introduction and background

Based on our experience as educators and particularly as lecturers in Information Technology (IT), we surmised that although pair programming had been proved to be an effective strategy for educating student teachers in a first programming course in our discipline (Bevan, Werner & McDowell, 2002; Williams, Wiebe, Yang, Ferzli & Miller 2002; Nagappan et al. 2003; Hanks, McDowell, Draper, & Krnjajic, 2004; Atli, 2006), its effectiveness could be improved by incorporating certain principles of cooperative learning. (The term ‘‘student teachers’’ refers to students having just completed high school/Grade 12, and enrolled in a four year initial training course (B.Ed degree) to become professional secondary or high school teachers.)

We used three research strategies to test our surmise. We began our investigation with a literature survey of pair programming (a form of collaborative learning) and of cooperative learning. We trawled the available literature to discover principles of cooperative learning that we could successfully incorporate into pair programming. The next step was to quantitatively compare the academic results of two groups of student teachers, one of which had been exposed to pair programming only, while the other had been exposed to pair programming into which also certain principles of cooperative learning had been incorporated. The third step of our verification process was a qualitative study.

We found that, provided that certain validity threats are taken into consideration, the incorporation of principles of cooperative learning to pair programming indeed had a positive effect on the academic achievement of the student teachers that had been exposed to them. The rest of this article will now be devoted to an explanation of the diverse set of evidence on which we base this claim.

Theoretical and conceptual framework

Pair programming, its essentials, pros and cons

Pair programming 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 (VandeGrift, 2004). Each member also has individual responsibilities and roles to perform (Hanks et al., 2004). Although research has proven the benefits of pair programming in learning programming skills (Mendes, Al-Fakhri & Luxton-Reilly, 2006; Werner, Hanks & McDowell, 2004; Williams, McDowell, Nagappan, Fernald & Werner, 2003), it has a few potential shortcomings, such as the danger of the academically stronger member of the pair doing all or most of the work (Cliburn, 2003; Williams et al., 2002; McDowell, Hanks & Werner, 2003), the possibility of a student remaining in the same role throughout and therefore failing to learn the skills associated with the other role, students reversing roles without approval of the instructor (Williams et al., 2002), the problem of intra-pair conflict resulting in inadequate collaboration (Williams et al., 2002), the possibility of students preferring to work on their own, either because intra-pair pressure prevents them from cheating (Williams & Upchurch, 2001) or because they insist that their personal achievement will be better (Waite, Jackson, Diwan & Leonardi, 2004), and the lack of time to meet with a partner (VandeGrift, 2004).

Solutions have been offered for these and related problems. Williams and Upchurch (2001) and Williams et al. (2002) suggested, for instance, that students should be required to report on their own contributions to their project as well as on those of their partners. Williams et al. (2002) recommended a strong supervisory role for the instructor/lecturer/facilitator, also for the purpose of ensuring that the members of a pair take turns at being either driver or navigator (Howard, 2006–2007; see Chong and Hurlbutt, 2007 about the effects of collaborators not necessarily hewing to the separate roles of driver and navigator). Le Jeune (2003) contended that individual assessment should serve as the ultimate source of each member's personal accountability. Peer evaluation can be invaluable in this process (Cliburn, 2003).

We concluded that all of these solutions seemed to form part and parcel of cooperative learning (see Johnson & Johnson, 2006). We could, however, not find any evidence of research attempting to combine all these principles (or cooperative supporting structures) in a single strategy and to empirically determine whether their application would lead to enhanced achievement in computer programming. Also, as far as we could ascertain, not much research

had been done to relate the strategy of pair programming as a form of collaborative learning to the principles of cooperative learning. In our opinion, as indicated above, pair programming could be made more effective by incorporating into it the principles of cooperative learning.

Cooperative learning

The problems mentioned above in connection with pair programming as well as the solutions offered to problems that might arise from its application are not unfamiliar to scholars in the field of cooperative learning. Veenman, Van Benthum, Bootsma, Van Dieren and Van der Kemp (2002) and Johnson and Johnson (2006), for instance, encapsulated such solutions in the form of five principles of cooperative learning, and advised instructors/lecturers/facilitators to take cognisance of them to ensure that all the members of a group achieved the desired outcomes. Although the groups in pair programming consist only of two members each, we surmised along with Preston (2006) that the incorporation of these five principles of cooperative learning into pair programming might lead to more effective achievement of outcomes. We therefore incorporated the following five principles of cooperative learning into pair programming.

Five principles of cooperative learning

Positive interdependence. Both members of a pair, driver and navigator, should understand that one of them cannot succeed unless they both do (Johnson & Johnson, 2006). The facilitator should ensure that they take turns at being driver and navigator (Howard, 2006–2007). Goal setting, incentives or bonuses for excellent work, collective responsibility, shared resources, mutual support and encouragement are required for the achievement of success (Veenman et al., 2002). Both partners should understand that their work will be assessed from time to time, and they should be given insight into how the assessment will work.

Individual accountability. Measures should be in place to ensure that both partners participate equitably (Gross Davis, 1999) and contribute towards achieving the expected outcomes. Performance assessment of each partner's work will ensure that both will contribute to the effort. The results should be given to both partners to reflect upon (Johnson & Johnson, 2006). By doing this, the facilitator ensures that the partners keep each other accountable. Accountability can be reinforced by the facilitator requesting any partner to demonstrate and explain their program to the rest of the class, and this can be followed by peer assessment. Bonus points given for excellent reporting can serve as a further incentive and promote awareness of responsibility (also see Johnson & Johnson, 2006). For individual assessment each member of a pair can also be required to write a program similar to that written in the pair. A student's personal accountability can be enhanced if the results of each member's individual assessment were given to the pair to reflect upon. If all these measures are in place, competition as well as blaming the other can be avoided (Johnson & Johnson, 2006).

Face-to-face interaction. The partners (physically) work together, encourage and facilitate each other's efforts to achieve success (see Johnson & Johnson, 2006). This results in providing assistance to the partner, in exchanging resources, and in challenging conclusions, reasoning and strategies. They should also be guided to understand how to act in the roles of driver and navigator respectively, how to assume responsibility, how to share roles, set targets for themselves as individuals and as a pair, divide tasks between them, and communicate with each other (McWhaw, Schnackenberg, Sclater & Abrami, 2003). This, in turn, will increase a willingness to ask for assistance, lead to an increase in confidence to suggest ways of improving a programming code and of solving problems. It will also result in reduced stress. To achieve all of this, each partner should understand exactly what is expected of them as individuals and as a pair, and they should master the skills required to achieve the objectives of the project. To facilitate this, the lecturer should provide time slots during which pair programming can take place and be observed by her, such as during scheduled contact sessions or close laboratory sessions (Preston, 2005).

Development of good social skills. The partners should communicate clearly and regularly with each other; they should develop interpersonal skills, learn to trust each other, and to resolve conflict amicably (Johnson & Johnson, 2006). They should learn to accept criticism graciously and to criticise in an acceptable manner, to test ideas (Williams & Kessler, 2000), to formulate a problem and discover appropriate strategies for overcoming obstacles, to encourage, and to compliment the partner on work well done. To ensure the optimal development of these skills, the partners should occasionally reverse the roles of driver and navigator (Williams et al., 2002). As their social skills improve, so does the enjoyment factor increase and the partners become more motivated for the task.

Group processing. The partners should periodically reflect on how well they are functioning and how they plan to improve their achievements (Johnson & Johnson, 2006). The facilitator should give them time for such reflection. During pair programming activities, the facilitator should also be constantly available for consultation, guidance and assessment.

After having incorporated these five principles of cooperative learning into pair programming, we proceeded with an empirical study, the methodology and findings of which we now present.

Quantitative empirical survey

The research design

Experimental quantitative research was done to gather data during two successive years and to compare the findings (see Leedy & Ormrod, 2005). (Although the course has been running since 2000 (only final course examination

marks available), pair programming was introduced in 2005, and the principles of cooperative learning incorporated only in 2006.)

The learning material

We selected a module in a university course in IT taken as a major by student teachers, namely Introduction to Delphi Programming, which is a beginning programming course for students in their second year (based on an introductory computer course in the first year) taken by student teachers preparing themselves to teach IT in secondary schools in South Africa, i.e. during the final three years of schooling (Grades 10 through 12). The module was a six-week course presented in the second term (from July to September) of both years, in the form of five 50 minute periods per week. The same experienced lecturer/ facilitator presented the module to both groups.

The population

All the students majoring in IT in 2005 (n = 27) and 2006 (n = 23) in a Faculty of Education Sciences at a South African university participated in the experiment. (The population numbers do not include the five students that dropped out during 2005 and the one that dropped out during the following year. The odd population numbers also meant that in both years there always was a ‘‘group’’ consisting of only one member. In contrast to 2005, rotation in 2006 ensured that no participant remained single for any extended period.) in 2005, there were 15 male and 12 female students in the group, and in 2006, there were 10 male and 13 female students. The average age of the students in the two groups was 20 years. There was no significant practical difference between the average M counts of the two groups (effect size d = 0.2). (South African universities use the M count for admission purposes – a value based on students’ marks in their final school year.)

Ethical aspects

Both the 2005 and 2006 groups of students were informed that their lecturer/facilitator was attending to the problem of discovering the most suitable teaching-learning method for the course. In a certain sense, it was business as usual for the students: they were taking a university course and trying to pass it. On another level, however, they were conscious of being involved in an experiment: in both years, they completed pre- and post-questionnaires about the teaching-learning strategy followed in that particular year. In addition to this, the 2006 group understood that their journals, the interviews with themand the field notes of the facilitator would be used to gauge the success of the teaching-learning strategy followed in that year. This group was also informed that responding to the questionnaires, keeping the journals and participating in interviews with the facilitator were not compulsory and that they would not be penalised in any way if they decided not to participate. All the students decided to participate. Permission to conduct the experiment was also obtained from the

Ethics Committee of the Faculty of Education Sciences.

Experimental and control groups

The participants in the experiment were second year students. The 2005 group served as the control group since they were expected to work in accordance with the principles of pair programming only (without the benefit of the incorporation of the principles of cooperative learning). The 2006 group served as the experimental group since they were required to work in accordance with the principles of pair programming into which principles of cooperative learning had been incorporated.

Because the number of students taking the course were relatively small, we did not have the benefit of using comparable experimental and control groups from the same academic year. Certain other factors also threatened the validity of the experiment: relatively small populations (which could mean that some of the differences in achievement could be ‘‘accidental’’), more males in 2005, and more females in 2006; two different groups, the fact that the impact of each principle of cooperative learning could not be determined and the fact that only the 2006 experimental group were expected to keep journals and were interviewed (the reasons for which we explain below as part of the qualitative research design). Despite all these threats, we proceeded with the experiment on the basis of two considerations. Firstly, we used the same facilitator, the same module within the same degree (B.Ed) as well as the total populations of each year. We supposed the two groups to be broadly similar; comparison of their M counts ($d = 0.2$ – low effect size) showed that they were at the same academic level. We further assumed the change in teaching-learning strategy to be the major difference. Secondly, we expected the experiment to still yield results that would be important and interesting to scholars in the fields of Information Technology, pair programming, cooperative and collaborative learning, and education in general – to the extent that they might feel to replicate the study in their own settings.

Different teaching strategies

Table 1 summarizes the teaching-learning strategies employed by the facilitator in helping the students master the contents of the module *Introduction to Delphi Programming* in the two years during which the experiment ran.

Data collection

The marks achieved in individual practical examinations in *Delphi programming* of the two groups (for 2005 and 2006), written at the end of the module each year, were collected and compared. Both examination papers covered the skills required by Delphi programming, and though not identical were of equal standard. In both papers, students were expected to write a complete Delphi program by applying advanced problem-solving skills combined with the

application of a graphical interface. The same learning outcomes had been set for both years, and the assessments were done by the same examiner and moderator. The moderator was purposely requested to ensure that the standard of the two examinations would be exactly the same.

Statistical techniques

Cohen's (1988) effect sizes (d = 0.2 – small effect; d = 0.5 – medium effect; d = 0.8 – large effect) were used to determine possible practical significance between the academic achievements of the two groups after completion of the course. The same was done with the M counts of the members of the two groups.

Findings

A large practical significant difference (d = 0.8) was found to exist between the academic achievements of the two groups in the end of module examinations. The averages indicate that the 2006 group performed considerably better on the same module than the previous year's group (see Table 2). No student failed in 2006 whereas four failed in the previous year.

Qualitative data

In addition to the two procedures so far described (the merits and demerits of pair programming into which principles of cooperative learning had been incorporated for the experimental group, and the gathering of quantitative data), we also collected qualitative data. The CSEd community has so far shown, mostly quantitatively, the benefits of pair programming in first programming courses. It is only recently that a qualitative lens has begun to focus on why these benefits occur (see Simon & Hanks, 2007), and we wished to gain deeper insight here.

Population

The same populations as for the quantitative study were used.

Instruments

Since both the control and the experimental group completed a *questionnaire* consisting of the following two open-ended items, this data could be used for comparative purposes:

> *Do you enjoy working in groups or pairs?*
>
> *What teaching-learning approach would you enjoy most? (Describe the pros and cons of your preferred approach.)*

Table 1. Teaching strategies 2005 vs. 2006.

Year 2005	Year 2006	Cooperative learning principles incorporated in 2006
Explanation of pair programming	Intensive training session on pair programming as well as cooperative learning	Knowledge of pair programming and cooperative learning improves goal setting and fosters positive interdependence
Complete assignments outside the scheduled contact session if unable to complete in class; assignments varied in length, each taking between 45 to 90 minutes; the fact that the students were not expected to be in class / laboratory promoted lack of focus	Both partners physically present; pair programming only performed during scheduled contact sessions; assignments to be completed at the end of a session for purposes of assessment; this contributed to stronger focus on the task at hand; assignments varied in length from 45 to 90 minutes	Face to face interaction; students expected to physically work together under controlled circumstances. Goal setting fosters positive interdependence. Students understand that they need to complete an assignment in the given timeframe and therefore realized that they were to swim or sink together
Students select their own partners and stay in the same pair for the duration of the course	Pairs each week randomly selected by the lecturer; no student worked longer than one week with a particular partner (cf. Williams, 2002, regarding the principle of switching partners every two to three weeks).	Development of social skills. Learn to work with different personality types and at varying skill levels
Need to change roles frequently, but decided on their own when to change. No role-changing supervised by the lecturer	Participants to complete two assignments while members of a particular pair; reversed roles after completion of the first assignment. Lecturer supervises role-changing (see Katira, Williams & Osborne, 2005 on pair compatibility)	Goal setting which fosters positive interdependence
Only the outcome of the assignment assessed; participants receive the same pair mark (only final module mark used for empirical study – see below)	Different assessment strategies: self- assessment, peer assessment, lecturer assessment, assessment of the demonstration, individual assessment;	Positive interdependence Individual accountability
	Individual assessment after each week on the outcomes of the specific week. Bonus points if both receive above 90% on individual assessment	Individual accountability Positive interdependence (goal setting)
	Average of individual marks taken into account for the pair mark as an incentive for the partners to contribute equally to the final achievement	Positive interdependence – one cannot succeed unless both do

(*continued*)

Table 1. (*Continued*).

Year 2005	Year 2006	Cooperative learning principles incorporated in 2006
The same assignment for all pairs	Each pair a different assignment of the same standard; demonstrated to the whole class group. Either members of a pair could be chosen to demonstrate	Goal setting which fosters positive interdependence; Social skills and individual accountability to formulate ideas and demonstrate to the whole class group;
No structured opportunity to reflect on how well the pair functions	Opportunity for reflection after each week on how well the pair functioned, which actions were helpful and which were a waste of time (reported in a journal)	Group processing
Frequent absences because of the fact that assignments could be completed at home	Both partners required to be in class; absent partners received no marks for work done; improvement in class attendance	Positive interdependence

Table 2. Average, standard deviation and effect size on examination marks for 2005 and 2006.

Year	N	Average %	Standard deviation	Number of students who did not pass	Effect size (d-value)
2005	27	48.7	16.1	4	0.8
2006	23	62.8	17.8	0	

To shed more light on why the 2006/experimental group fared better after the incorporation of principles of cooperative learning in pair programming, not for purposes of comparison, the members of this group were requested to each keep a *journal* in which they had to respond to five open-ended questions:

> *Did any of your partners at any time enjoy a free ride at your expense in this project? Briefly describe your experience in this respect.*
>
> *How would you describe your own contribution to the success of the project?*
>
> *Was the mark given for the project fair in your opinion? Substantiate your answer.*
>
> *Which aspects did you like most when working in a pair?*
>
> *Which aspects did you like least when working in a pair?*

They were also free to write down their reflections about what transpired during the module.

The facilitator–researcher also conducted interviews with a few of the 2006 participants (n = 6, of whom three scored more than 80% in the practical module examination and three below 60%). The purpose of these interviews was to understand how students who actually experienced pair programming combined with the principles of co-operative learning felt about this particular approach, and also to shed light on the improved achievement of the 2006/experimental group (not for purposes of comparison). Each interview took about 30 minutes and was audio-taped (see Simon & Hanks, 2007 on the value of semi-structured interviews).

Data processing

We used a phenomenological approach for understanding the participants' views on pair programming as expressed in the questionnaire, the journals and the interviews. According to Leedy and Ormrod (2005), a phenomenological study is a study that attempts to understand people's perceptions, perspectives and understanding of a particular situation, in this case experience of pair programming infused with the principles of co-operative learning. Merriam (2002) concurs that *the purpose of such a study is to describe the meaning of a lived experience from the perspective of those experiencing the phenomenon.* The most distinguishing feature of phenomenological research is its focus on describing the ''essence'' of a phenomenon from the individual's perspective. Although phenomenological researchers tend to use long, open-ended interviews for data gathering, other techniques (such as document analysis) may also be used in a phenomenological study (Merriam, 2002). Content analysis, as outlined by Leedy and Ormrod (2005), was applied to the transcribed interviews and the students' journals.

A list of significant and non-overlapping statements was compiled from the transcriptions; all irrelevant and redundant information was excluded. Several main themes emerged from clusters of statements. The derivation of themes was cross-checked by the research assistant who attended the 2006 pair programming sessions.

The same process was repeated with respect to the contents of the journals and the responses to the questionnaire. These analyses also yielded certain themes with respect to the students' experiences of the 2006 course.

Findings

Analysis of the participants' responses with respect to the two items in the questionnaire (both at the beginning and the end of the module) revealed that the participants in both groups enjoyed working in pairs. Whereas more than one of

the participants in the 2005/control group still insisted at the conclusion of the module that their partners had enjoyed a free ride, none of the 2006 participants reported any similar experience. Members of both groups also indicated at the beginning (but not at the end) that they expected the work distribution to be unequal and that people would not do their best in group situations. After completing the module, some members of the 2005/control group indicated in their responses to items in the questionnaire that they had indeed encountered problems with pair programming because their personal success depended on others. They complained that some partners did not contribute to the project to the expected degree. No such complaints were received from the 2006 group after the completion of the module. Only one participant in the 2006 group indicated after completing the course that she would have determined her own pace if she had been allowed to work on her own.

Since our analysis of the before- and after-questionnaires failed to reveal exactly why the 2006 experimental group had fared better than the 2005 control group, we analysed the responses of the former group in their journals, and cross-checked the findings with the facilitator's daily field notes. We found the following to be key aspects of their success:

* Fairness of the assessments: With the exception of one, all the participants perceived all the assessment procedures to be fair. Only one respondent only once indicated that he or she had felt it to be unfair.

* The effects of incorporating principles of cooperative learning: Table 3 contains 7 positive remarks pertaining to cooperation (in comparison with only 3 negatives in Table 4).

* The acquisition of new knowledge and skills: Tables 3 and 5 show that the 2006 group succeeded in mastering several new forms of knowledge and (higher order) skills. The journals contained no negatives in this respect.

* Social interaction was perceived as both a positive (Table 3) and a negative (Table 4), which makes it impossible to draw any firm conclusions here.

The aforementioned interviews with the 6 participants from the 2006 experimental group yielded the following:

* In general students liked pair programming (no negatives were recorded in this regard). The following were typical responses:

> *You can always learn something from your partner and even if you need to guide yourpartner, the fact that you need to explain a difficult programming action helps you inunderstanding the concept better as well.*
> *We teach each other.*

Table 3. Favourable aspects of pair programming with principles of cooperative learning incorporated.

Theme	Quotations from participants' journals
Cooperative skills	To help each other Share new knowledge and information Cooperation to achieve the goal I understand problems easier and quicker when shared with my partner Decisions made by both To argue about a problem helps to solve it I learned to cooperate with somebody
Social interaction	I learned to communicate with my partner Met new friends
Knowledge and skills	I learned different ways of programming and problem solving

Table 4. Unfavourable aspects.

Theme	Quotations from participants' journals
Cooperative skills	My partner wanted to work too quickly We had different opinions about how to solve the problem I wanted to choose my own partner
Social interaction	My partner accused me of not listening My partner did not communicate effectively My partner could not make decisions

In my opinion, pair programming is the way in which programming needs to be learned.
I enjoyed working with somebody else.

* The majority of students opined that they would not have achieved the same mark if they had worked on their own. The following was a typical response:

I would have achieved much less.

*Students thought that occasionally changing the roles was important to ensure that both partners get the opportunity to operate the keyboard and to guide the highlevel thinking processes.

The (different) roles force communication between the two partners.
It was good practice to keep the roles the same for the whole duration of each assignment because that keeps your mind on track without interfering with the thought processes.

Table 5. Skills learned.

Theme	Quotations from participants' journals
Programming skills	I learned new programming skills I learned shortcuts in programming I learned to keep on trying to detect the errors until the program ran I learned how to deal with dialog boxes, list boxes and recursion
Higher order thinking skills	I learned to reflect on work done previously I learned to be more creative I learned to evaluate my own work I learned to plan before starting programming I learned to think logically
Social and communication skills	I learned to cooperate with others I learned to communicate I learned to listen to others I learned to be patient I learned to keep a cool head I learned social skills while working in a pair I learned the value of mutual trust

* In general, participants had no problem with working with a partner with less knowledge of programming skills. To work with somebody that is an expert programmer:

> *could at first be intimidating.*
> *could be a bonus when the person has good communication skills and helps you understand the programming concepts.*
> *never was a problem, I adjusted to the situation.*
> *taught me a lot; I asked a lot of questions.*
> *forces you to work harder and to be better prepared.*

* Students felt it was important to learn to work with different partners. They also felt a strong individual responsibility while working in a pair, which is an indication of the role that positive interdependence and individual accountability could play in pair programming coupled with the principles of cooperative learning.

> *If you do not do your part of the work, you are not going to reach the goal and achieve good scores in the individual test.*
> *You will let your partner down.*

* Students knew that they would not achieve the goal if both partners did not participate in full.

> *I feel it is my duty to explain to my partner when I realise that he/she does not understand a specific concept.*

I feel that I helped my partner in understanding the programming concepts.
I will never carry on if my partner does not know what I am doing.

*Participants felt that both partners had a unique responsibility in completing the assignment.

If everyone did not do his bit, we would not have achieved success in the test.
Both of us had a strong feeling of responsibility, and each contributed to the joint effort.
It was important for both to understand what was happening.
(All the interviewees offered similar remarks.)

*In their opinion, the mark allocation was fair. The fact that they were expected to allocate a score for their partners' participation in the project was a good practice that ensured that both partners took equal responsibility.

If you don't contribute to the effort, you have to expect a lower test mark.
If your partner fails in contributing as much as you did, you penalise him or her by giving him/her a lower mark.
The fact that partners do not attain the same marks is fair practice if you have done more than your partner.

* Under the heading *"general remarks"*, participants again stressed the advantages of working in pairs.

It gives you more confidence to ask questions.
You learn to ask questions because you realize that nobody has all the answers.
Because of the fact that we need to work in pairs with almost all the members in the class, you get to know each other.

*A careful analysis of the transcribed protocols of the interviews revealed no negative aspects of the teaching-learning strategy that had been employed for the experimental group in 2006.

The results of the interviews confirmed that the incorporation of principles of cooperative learning could indeed be regarded as key to their success.

Concluding remarks

Our research formed a diverse set of evidence that led us to conclude that pair programming into which certain principles of cooperative learning had been incorporated could serve as an effective teaching-learning strategy for mastering

computer programming. We found this claim to be supported not only by our literature review on pair programming as a form of collaborative learning, especially when principles of cooperative learning have been incorporated, but also by our quantitative and qualitative research. The experimental group not only outperformed the control group in their practical module examinations and there were fewer dropouts, but also, generally speaking, both groups thought pair programming to be conducive to the effective learning of programming skills. In contrast to some members of the 2005 control group, the 2006 experimental group's questionnaires, journals, interview results and the facilitator's field notes showed that all of them had contributed to the success of the pairs to which they belonged. The qualitative data revealed that they had experienced the learning during 2006 quite positively, and had acquired several skills associated with cooperative learning. This, in our opinion, emphasizes the need to incorporate principles of cooperative learning into pair programming. Another problem associated with pair programming, namely the perceived unfair allocation of marks, also seems to have disappeared with this approach.

Despite several factors that threatened the validity of our experiment, pair programming can be seen as an effective strategy for teaching and learning programming skills, especially if its effectiveness could be improved by the incorporation of principles associated with cooperative learning.

Acknowledgement

This research is based on work financially supported by the National Research Foundation (NRF) in South Africa. Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF does not accept any liability in regard thereto. We hereby also acknowledge the invaluable input of three anonymous reviewers.

References

Atli, G. (2006). Critical personality traits in successful pair programming (Masters dissertation, Bowling Green State University). *OhioLINK Electronic Theses and Dissertations Centre*, bgsu1150232487.

Bevan, J., Werner, L., & McDowell, C. (2002). Guidelines for the use of pair programming in a freshman programming class. *Proceedings of the 15th Conference of Software Engineering Education and Training*, 100–107.

Chong, J., & Hurlbutt, T. (2007). The social dynamics of pair programming. *Proceedings of the 29th International Conference of Software Engineering*, 354–363.

Cliburn, D.C. (2003). Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges*, 19(1), 20–29.

Cohen, J. (1988). *Statistical power analysis for behavioural sciences* (2nd ed.). New York: Erlbaum.

Gross Davis, B. (1999). Cooperative learning: Students working in small groups. *Speaking and Teaching*, 10(2), 1–4.

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1*. Association for Computing Machinery (ACM) Special Interest Group Computer Science Education Bulletin*, 36(3), 176–180.

Howard, E.V. (2006–2007). Attitudes on using pair programming. *Journal of Educational Technology Systems*, 35(1), 89–103.

Johnson, D.W., & Johnson, R.T. (2006*). Joining Together. Group theory and group skills* (9th ed.). New York: Allyn and Bacon.

Katira, N., Williams, L., & Osborne, J. (May 2005). Towards increasing the compatibility of student pair programmers. *International Conference on Software Engineering*, 625–626.

Leedy, P.D., & Ormrod, J.E. (2005). *Practical research: Planning and design* (8th ed.). New Jersey: Pearson.

Le Jeune, N. (2003). Critical components for successful collaborative learning in CS1. *Journal of Computing Sciences in Colleges*, 19(1), 275–285.

McDowell, C., Hanks, B., & Werner, L. (2003). Experimenting with pair programming in the classroom. *Association for Computing Machinery (ACM) Special Interest Group Computer Science Education Bulletin*, 35(3), 60–64.

McWhaw, K., Schnackenberg, H., Sclater, J., & Abrami, P.C. (2003). From co-operation to collaboration: Helping students become collaborative learners. In R.M. Gilles (Ed.), *Co-operative learning: The social and intellectual outcomes of learning in groups.* London: Routledge Falmer.

Mendes, E., Al-Fakhri, L., & Luxton-Reilly, A. (2006). A replicated experiment of pairprogramming in a 2nd year software development and design Computer Science course*. Association for Computing Machinery (ACM) Special Interest Group Computer Science Education Bulletin*, 28(3), 108–112.

Merriam, S. (2002). *Qualitative research in practice. Examples for discussion and analysis.* San Francisco: Jossey-Bass.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *Association for*

Computing Machinery (ACM) Special Interest Group Computer Science Education Bulletin, 35(1), 359–362.

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.

Preston, D. (2006). Using collaborative learning research to enhance pair programming pedagogy. *Association for Computing Machinery Special Interest Group for Information Technology Education (ACM SIGITE)*, 3(1), 16–21.

Simon, B., & Hanks, B. (2007). First year students' impressions of pair programming in CS1. *International Computing Education Research 2007*, pp. 73–85. September 15–16. Atlanta, Georgia, USA.

VanDeGrift, T. (2004). Coupling pair programming and writing: Learning about students' perceptions and processes. *Special Interest Group Computer Science Education*, 2004, 2–6.

Veenman, S., Van Benthum, N., Bootsma, D., Van Dieren, J., & Van Der Kemp, N. (2002). Cooperative learning and teacher education. *Teaching and Teacher Education*, 18, 87–103.

Waite, W.M., Jackson, M.H., Diwan, A., & Leonardi, P.M. (2004). Student culture vs. group work in Computer Science. *Association for Computing Machinery (ACM) Special Interest Group Computer Science Education Bulletin*, 36(1), 12–16.

Werner, L.L., Hanks, B., & McDowell, C. (2004). Pair-programming helps female Computer Science students. *Journal of Educational Resources in Computing*, 4(1), 1–8.

Williams, L.A., & Kessler, R.R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the Association for Computing Machinery*, 43(5), 109–114.

Williams, L., McDowell, C., Nagappan, N., Fernald, J., & Werner, L. (2003). Building pair programming knowledge through a family of experiments. *International Symposium on Empirical Software Engineering* (pp. 143–152). Italy: University of Rome.

Williams, L., & Upchurch, R.L. (2001). In support of student pair programming. *Special Interest Group Computer Science Education*, 2001, 327–331.

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