Increasing self-efficacy in learning to program: exploring the benefits of explicit instruction for problem solving

I GOVENDER,[1] DW GOVENDER,[2] M HAVENGA,[3] E MENTZ,[4] B BREED,[5] F DIGNUM[6] AND V DIGNUM[7]

Abstract

The difficulty of learning to program has long been identified amongst novices. This study explored the benefits of teaching a problem solving strategy by comparing students' perceptions and attitudes towards problem solving before and after the strategy was implemented in secondary schools. Based on self-efficacy theory, students' problem solving self-efficacy as well as teachers' self-efficacy were investigated, showing that both students' and teachers' self-efficacy may have benefited from the explicit instruction. This would imply that teaching problem solving explicitly should be encouraged to increase self-efficacy to program.

Keywords: Programming, problem solving, teaching and learning, self-efficacy

1. Introduction

Common problems with introductory programming courses at universities and secondary schools include inadequate learning outcomes and consequently low pass rates (Kinnunen & Simon, 2012). Several studies (for example, Kinnunen & Simon, 2012; Govender & Govender, 2012) have pointed out the successes and difficulties experienced by students in introductory programming courses. Problem solving has been shown to be one of the key concerns, as it carries a high cognitive load. While a number of instructional methods and strategies have been proposed (see Lau & Yuen, 2009), to date, learning to program still poses difficulties, which in turn contributes to the low pass rates. For instance, a study that analysed the first year attrition rate in computer programming courses revealed that the main

[1]. Corresponding author: Dr Irene Govender, Discipline of Information Systems and Technology, University of KwaZulu-Natal, Govenderi4@ukzn.ac.za

[2]. Dr Desmond W Govender, Discipline of Computer Science Education, University of KwaZulu-Natal, Govenderd50@ukzn.ac.za

[3]. Dr Marietjie Havenga is attached to the Faculty of Education Sciences, North-West University, Potchefstroom. Email: Marietjie.Havenga@nwu.ac.za

[4]. Prof Elsa Mentz is attached to the Faculty of Education Sciences, North-West University, Potchefstroom. Email: Elsa.Mentz@nwu.ac.za

[5]. Dr Betty Breed is attached to the Faculty of Education Sciences, North-West University, Potchefstroom. Email: Betty.Breed@nwu.ac.za

[6]. Prof Frank Dignum, Utrecht University, Utrecht, The Netherlands, dignum@cs.uu.nl

[7]. Prof Virginia Dignum, Delft University, Delft, The Netherland, M.V.Dignum@tudelft.nl

challenge faced by students is a lack of problem-solving skills (Ford & Venema, 2010). According to Ismail, Ngah, and Umar (2010) a number of aspects of programming are not being taught explicitly. Hasni & Lodhi's (2011) study affirms this belief that teaching problem solving requires more than giving practice exercises to be solved; rather, the actual processes of problem solving must be taught.

In the context of changing computer languages and technologies, professional development becomes critical for teacher growth (Govender et al., 2013). Mentz et al's (2012) investigation of information technology (IT) teachers in economically deprived schools in South Africa revealed that many teachers lack the technical skill and pedagogical content knowledge. To address these challenges, a team of researchers implemented a support programme for a small group of IT teachers. The overall goal of the programme was to improve the quality of learning and teaching programming across six secondary schools across two provinces in South Africa. This article reports on one aspect of the implementation of the support programme, which is the teaching of problem solving and the resulting benefits thereof. Specifically, the paper seeks to answer the following question:

How does the explicit instruction of problem-solving affect students' self-efficacy to program?

This paper begins with a brief discussion of some of the literature relevant to the study of problem solving and self-efficacy, as this enables us to locate our approach to the problem of learning computer programming. Transdisciplinary perspectives from psychology, teacher-education and computer programming are integrated to address the issue of problem solving as it relates to programming. Self-efficacy and its related concepts, which have their foundations in social psychology, were used to suggest a conceptual framework for this study. In the next section the conceptual framework explaining the link between teaching explicit problem solving strategy and improved self-efficacy in programming is provided. The transdisciplinary approach integrates the process of self-efficacy analysis with insights borrowed from teacher education and programming in order to solve the practical problem referred to above. A description of the methodology is provided, followed by the analysis and a discussion of the key findings. The article concludes by suggesting that explicit problem solving instruction should be encouraged as it improves learners' as well as teachers' self-efficacy in programming.

2. Review of related literature

It is a commonly held view that programming is a complex and difficult task for secondary students; it requires skills such as problem-solving, abstraction, mathematical logic and testing, debugging and troubleshooting (Saeli, Perrent, Jochens & Zwaneveld, 2011). Problem-solving lies at the heart of programming and many recent studies focus on this high-level skill as the key issue in programming (Govender, 2010; Lau & Yuen, 2009). The process of developing an algorithm – a step-by-step list of instructions for solving an instance of a problem is referred to as problem-solving. Programming is an integral part of computer science and its related fields. It has been shown that the number of students participating in computer courses in secondary schools has been declining, internationally (Downes & Looker, 2011). Anecdotal evidence suggests the same locally. More specifically, there is some speculation that the difficulty experienced by students in programming may contribute to the low number of students choosing to study computer science (Maddrey, 2011) or information technology (IT), as it is referred to in secondary schools in South Africa. Programming is a

major component of IT and if we understand what leads to success in programming, it might be possible to attract more students to study IT. The importance of the need to engage students in explicit problem-solving strategies to improve self-efficacy in the field is reflected in the paper. This study is driven by the commitment to engage in the explicit instruction of problem-solving strategies, and by the low number of students choosing to study computer science or IT. Not much has been written about IT secondary students' problem-solving ability and IT teachers in the provinces under investigation.

In this study, we explore another aspect that could have an effect on students' performance: self-efficacy to program. Seturaman and Medley (2009) indicated that students' beliefs about their self-efficacy in programming could be used to determine how well the students are doing in their programming courses. Furthermore, it has long been known that self-efficacy is an important motivation for learning (Zimmerman, 2000). In a related study, Askar and Davenport (2009) identified factors such as gender, computer experience and family usage of computers as variables that are related to self-efficacy in general. Jegede (2009) established that prior programming experience of the novice learner did predict Java programming self-efficacy.

Although previous studies have addressed approaches to teaching programming (Hasni & Lodhi, 2011) and factors that affect students' self-efficacy in relation to programming success (Seturaman & Medley, 2009), few, if any, have addressed the relationship between problem-solving efficacy and success in programming. Bandura (1994: 71) defines self-efficacy as "people's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives," and states that these beliefs "determine how people feel, think, motivate themselves and behave".

Maddrey (2011) determined that problem solving is one of the main factors that make programming difficult. In this study, it was surmised that with appropriate introduction to problem-solving, a student would become adept at thinking through the processes for different programming problems. At the same time, as students work with the problem, they may increase their self-efficacy in programming. We sought to explore the effects of teaching a problem-solving strategy explicitly.

3. Conceptual framework

Some research in self-efficacy advocates that teachers would gain more insight by considering not only students' actual competence, but also their perceptions of their competence. According to Hackett and Betz (1989), students' motivation and future academic preferences may be based on these perceptions. As noted earlier, researchers have shown that self-efficacy beliefs strongly influence the choice of majors and career decisions of students. However, it was made known that in some cases a lack of competence or ability was not necessarily to be blamed for avoiding computer courses, but that unreasonably low perceptions of self-efficacy were the cause (Hackett & Betz, 1989). School and teaching practices that foster both competence and the necessary accompanying confidence should be identified, as well as practices that "convert instructional experiences into education in efficacy" (Bandura, 1997: 5-12).

Bandura (2006) further argues that studies of teacher efficacy and the effect that teachers' self-beliefs have on their practices and student outcomes, will help explain how teachers' beliefs influence students' beliefs and achievement. According to the European Commission (2010), when teachers have a strong sense of self-efficacy they become more resourceful in

their work, by persevering in and strengthening their efforts in order to meet their goals. Hence teachers' sense of self-efficacy can influence the learning and motivation of students.

In an earlier study, Bandura (1986) emphasised that being involved with the specific task experience—in this case problem-solving in programming—is the most effective source of self-efficacy information, in that it has critical implications for education. This implies that educational efforts should therefore focus on improving students' self-beliefs in order to improve achievement, even if success is not immediately obtainable, but achievable in the long term.

Stated differently, self-efficacy beliefs influence human behaviour through cognitive, motivational, affective, and decisional processes. They affect whether individuals think in self-enhancing or self-weakening ways, and how well they motivate themselves and persevere in the face of difficulties (Bandura & Locke, 2003).

In order to provide a theoretical lens to study the degree of confidence and motivation that students have in performing the problem-solving activities, we formulated a framework. This framework was based on the concepts of self-efficacy for the specific task of problem solving in programming. Bandura (2006: 307) states that "the efficacy belief system is not a global trait but a differentiated set of self-beliefs linked to distinct realms of functioning." There is no "one measure fits all" perceived self-efficacy. We identified concepts within the domain of problem solving with regard to specific tasks, using efficacy theory on the interpretive concept "I can" (Bandura, 2006). By the process of concept-driven and axial coding – explained further in the analysis section – the following seven dimensions of the learning experience were identified: *Enjoyable and supportive, solving simple problems, solving complex problems, perseverance, self-regulation, confidence boosting, overcoming difficulties.* These dimensions corresponded with the efficacy scale for computer programming as presented by Ramalingam and Wiedenbeck (1998), which has been validated in a number of studies (for example, Jegede, 2009, Askar & Davenport, 2009).

4. Methods and Materials

4.1 Procedure

This qualitative study followed an interpretive, descriptive and exploratory approach. The study focuses on the participants' perceptions and experiences of using the problem solving strategy taught which is characteristic of the qualitative approach. Creswell (2003) indicates that exploratory studies are most advantageous when little has been written about the phenomenon or the population being studied.

The study is located within a larger project that aimed at supporting and developing IT teachers' practice particularly in rural schools. The project included professional development for IT teachers, specifically with regard to teaching strategies in programming in order to improve students' programming skills. One such strategy was for problem solving and was labelled 'explicit instruction for problem solving'. The strategy was first work-shopped with the teachers before they implemented it in their Grade 10 classes. A comprehensive handbook was developed that indicated the problem-solving activities to be followed as well as a number of relevant exercises with the corresponding activities required. Table 1 presents a snapshot of the problem-solving activities involved in the strategy.

Table 1: The problem-solving activities framework (Adapted from Problem solving Activities: SANPAD Project, M. Havenga, 2011)

Main problem-solving activities
1. **Write down the main ideas and requirements of the problem.** -Read the problem and underline the key concepts to understand and interpret the question clearly. -Determine what you do not understand.
2. **Represent the problem** by using a diagram, table, flow chart, description or any other method to indicate how you understand the problem.
3. **Plan the detailed steps** -Determine the purpose of each method. -Plan the input, processing and output. -Go back to Step 1 and check whether you are on the right track.
4. **Code your planning in a programming language.** -Determine which code/constructs you will use to input the data. -Which statements will you use to process or calculate the data? -Which statements will you use to display the output? -Compile the program and correct the programming errors.
5. **Reflect on how well you have solved the problem.** -Use test data and ensure that the extreme cases of test data are included. -How did you choose the test data and extreme values? -Explain if you could correct any programming errors. -Did you use resources to support your programming process? -Are you satisfied with your solution? Explain.

4.2 Participants

Students from six classes from six secondary schools from two provinces in South Africa participated in the project "Empowering IT Teachers in Economically Deprived Rural Schools". A total of 96 students and six teachers participated in the project.

4.3 Data Collection

Data was obtained using three instruments, namely, questionnaires, interviews and journals. During the data collection process, the concept of self-efficacy emerged as an important aspect in the *project* as is characteristic of grounded theory. For the purposes of this report, self-efficacy was then used as the conceptual framework for one aspect of the project. While it is common practice to assess students' self-efficacy by asking them to rate their confidence according to some scale, it is considered just as valid to engage students in self-regulatory strategies in writing specifically about the problem at hand in response to probing questions (Shell, Colvin & Bruning, 1995). There are various ways to measure self-efficacy (Pajares, 1997). In general, when assessing self-efficacy, a rating scale may be used with particular emphasis on the subject area needed (Pajares, 1997). In this study, students' thinking processes were elicited from the questionnaires in order to determine students' self-efficacy

with regard to problem-solving as it relates to programming. Similarly, the interviews and journal entries served the purpose in determining the teachers' self-efficacy.

4.3.1 Questionnaires

Two sets of questionnaires, in the form of a comprehensive worksheet (eliciting respondents' thinking processes) were used in the study. The first questionnaire was administered before the problem solving strategy was taught. It included a problem to be solved – students were asked to show their calculations, and explain their thinking process and any difficulties they may have experienced as well as how they overcame them. After the teachers had taught the problem solving strategy over a month-long period, a second questionnaire was then administered, which included another problem of comparable level to what was done in the first questionnaire. Participants answered questions regarding the given tasks with respect to whether they thought they had solved them, their experience in using the problem-solving activities (strategy), whether the problem-solving activities supported them in their programming, their difficulties encountered and how they overcame these difficulties. The questionnaires were administered to the students with the assistance of their teachers during their normal class time.

4.3.2 Interviews and journal entries

In order to obtain an overall understanding of the process and experiences of both students and teachers, and the teachers' views of the students' experiences, semi-structured interviews were conducted with teachers. These semi-structured interviews were meant to determine their views of problem-solving before and after they implemented the problem-solving strategy. They also kept a journal of entries – indicating their challenges, problems and achievements – as each lesson was taught for the duration of the intervention. Using the students' responses from the questionnaire, we were able to triangulate the data for validity and reliability.

4.4 Analysis

The data in the questionnaires were arranged and coded according to a list of a *priori* themes — a process known as concept-driven coding (Gibbs, 2010) – based on the theory of self-efficacy to determine their perceptions of solving the given problems before and after the strategy was taught. The two coding procedures that are characteristic of grounded theory — open and axial coding — were used to further analyse the text data; moving backwards and forwards between both approaches to coding (Gibbs, 2010). Both concept-driven, and open and axial coding can be used concurrently to arrive at a set of appropriate categories. In the open coding process, themes were identified and named. Axial coding was then used to find relationships between the different themes to form more abstract categories. A brief description of data captured before and after the strategy was taught is presented. Both sets of results were compared and analysis conducted. The data was collected and analysed by a team of researchers in the project thus ensuring researcher triangulation and validity of the study.

5. Ethical measures

In this study, ethical measures included assurance of anonymity and obtaining the informed consent of teachers and students, as well as permission from the Department of Education and the university to conduct the research. The aims of the study and methodologies were communicated to all of the participants in the project. It was made clear to all participants that they could withdraw from the process at any stage and that their decision to participate

or not would in no way be viewed negatively by the project facilitators and researchers. However, while students' consent was obtained for their participation and use of their work for analysis, their participation was directly related to the teachers' consent as well. This meant that if teachers withdrew, then the respective students would not be able to participate. We believe these measures allowed us to meet the standards that can be considered appropriate when conducting research in an ethically acceptable manner.

6. Findings and Discussion

This section reports on the main findings and suggests possible implications of the use of the teaching strategy for affecting student self-efficacy towards programming. Specifically it reports on students' experiences with the strategy, teachers' responses regarding their experiences with the teaching strategy, and the problems encountered. Supporting quotations from students and teachers with regard to the framework for self-efficacy are presented. In addition, a brief description of the statistics regarding the responses in the questionnaires before and after the strategy was taught is given.

6.1 Students' experiences of solving a problem before the introduction of the strategy

From 95 pre-questionnaires received, only four students obtained the correct answer to the problem given as part of the questionnaire (see Appendix A for the problem). However, 26 students believed that they had obtained the correct answer, 34 were not certain that they had solved the problem and 35 students believed that they had not solved the problem. The majority (73%) of the students were not able to overcome the difficulties that they experienced. It is clear from the data obtained in the questionnaire that most students showed no planning in their solutions to the problem. The written responses to the question "Explain how you solved the problem", showed very little planning or no planning at all and a misguided approach to their thinking. Because of the large dataset, it would not be feasible to quote all the explanations; however, some typical explanations of their workings are given below:

> I multiplied the amount with copies and then I added it amounts that were the answer then added them all together then come out with final answer.
>
> I calculated all the money and multiplied it all and then at the end I come out with a total.

From their explanations, we identified the students' perceptions of problem solving in programming. The main themes identified in the students' responses to the question, "If you experienced any difficulties, did you overcome them? If so, how?" were frustration and not being able to overcome their difficulties. Some quoted examples of student responses to the question are given in Table 2 below.

Table 2: Examples of student responses prior to learning the problem solving steps explicitly

Yes I did experience some difficulties but the worse thing is that I did not overcome them because I did know how to calculate
Difficulties that I have faced is that I have not manage[d] where to start and finish.
I didn't overcome the difficulties because I don't understand
Yes, I got some difficulties but not that difficult and I overcame them by reading to understand, that is the important rule for reading. After I understand the problem and then I made the calculations
No, because the difficulties that were there are worrying me especially when comes to calculations of that were there.
I get frustrated and try to work through the problem, but it depends on what kind of problem it is.
It was hard for me to start but if you are a IT learner you have to open minded so that you can able to overcome any programming problems. I tried by all means to solve it by myself.
I didn't overcome the difficulties that I have faced, I'm not even sure of my answer
I did not overcome them it was hard to solve and I was solving it alone no formula no understanding
I tried in any way to solve it by counting in any other way I heard to but the problem was too hard to solve and it was complicated.

Given that these students had chosen to study IT, one would have expected a certain level of interest from them and to some extent the motivation to learn IT. These two aspects are part of self-efficacy. It would appear that their sense of self-efficacy in relation to problem solving was relatively low. The self-efficacy perceptions identified in section 3, (namely, *enjoyable and supportive, solving simple problems, solving complex problems, perseverance, self-regulation, confidence boosting, and overcoming difficulties*) were not discernible in the dataset obtained before the problem –solving strategy was implemented.

6.2 Students' experiences of solving a problem after the implementation of the strategy

After the students were explicitly taught problem solving using a specific framework, they used the strategy to solve all problems they were given as new constructs in programming were taught. This framework was then used throughout the teaching and learning process. After a period of approximately five weeks, the second questionnaire on problem solving was administered. The response from students regarding the problem-solving strategy was overwhelmingly positive, and their confidence level in problem solving had increased significantly.

Using concept-driven and open coding, seven categories or concepts were identified in the responses given in the post-questionnaire, three of which relate to their views of the problem-solving strategy (activities) used. The next four categories were based on self-efficacy theory with emphasis on the interpretive meaning "I can" in their explanations. Table 3 lists these concepts together with extracts from the data that contributed to their identification.

Table 3: The identified concepts with examples of evidence from the data

Concepts/category	Excerpts from questionnaires in support of the concepts/categories identified
Supportive and enjoyable using the problem solving activities [framework] to solve problems	They were challenging and enjoyable. They make you think and make you use the skills you have acquired in programming […..] were very interesting because some of them it was for the first time we met with them and they became very educational and not only at a programming level. It helps me because it has an easy way to explain about steps you can follow to solve a programming problem. Without the problem-solving activities it will be like trying to cut a plank with your hand.
Solving simple problems	Problem solving activities helped in making me aware of how to step by step approach programming problems. So I enjoyed them very much because they made programming understandable with their algorithm solutions and also made programming seem easy and interesting. Problem solving activities were a fun topic to deal with and I hope to meet with them again because they bring out the inner thinking abilities of a programmer. It help me to understand and it give me some instruction to know how can I start a simple program or a class and it give me some point to plan to develop a program.
Handle complex programming tasks	I find the problems easier to solve because they [teach] learn you how to solve it step by step, how to separate the problem into 2 or more parts so that it becomes easier to solve. Yes it helps you clearly identify your input, process and output especially with the long programs
Perseverance	The experience you get is that you can do [this] problems and you will attempt other problems knowing that you can or you are able to solve the problems on your own without any help from the teacher or other students. Expectation of difficulties on the problem lead me to hard concentration and communicating with a partner how we are going to break it and then we came up with solutions. The problem that I experienced is that I forgot how the range in calculated. I overcame my difficulty by going to a search engine called google that's where I found all the methods of data handling and I chose the formula for Range.
Self-regulation	It is the key to knowing or understanding Information Technology. Problem-solving activities define almost everything we need to know and with the help of the professor you adapt easily to problem solving activities. Reason: It has boost my understanding of IT or the computer itself. I now know the uses and functioning of the computer. Yes I think it supports me because it gives me a chance to solve a programming problem by following steps and asking me questions that will help me in order to correctly solve the problem that I was given.

Confidence boosting	The problem solving activities was more enjoyable I really felt proud after completing it. I realize the difficulties in examples 5 and 6 because it was little bit challenging for me, but I tried my level best to overcome, it is that which brings frustration in my problem solving activities but at least I manage to [learn] design. The problem-solving activities were challenging but they were also enjoyable and I managed to see how problem-solving activities could boost you to solve the problem and your skills in programming. Yes because they help me to write the questions in my own words so that I can understand them better
Overcoming difficulties	I experienced some difficulties during solving the problem and I overcame them by researching information, discussing the problem with other learners who are doing IT and then explained to each other and now we solved the problem. …. I experienced problems in determining which value is the largest between average and range midpoint so I referred back to the textbook and I chose (from several options) the if-then else statements to approach the problem.

The activities that formed part of the problem-solving strategy came forward as particularly significant in increasing their *confidence*. More importantly, the *challenge* of following the strategy in trying to solve the problems gave the students immense satisfaction (*problem-solving activities were challenging but they were also enjoyable and I managed to see how problem-solving activities could boost you to solve the problem and your skills in programming*…). The enjoyment of completing the task contributed to their satisfaction as well. The activities of the strategy and the questionnaire forced students to think about what they were doing and to reflect on their actions. This *self-regulatory* process enhanced their motivation. Hence *planning and perseverance* became apparent because of their enjoyment and boosted confidence. In following the activities of the problem-solving strategy, students were able to handle *simple problems* as well as *complex problems*. Even though it was not always feasible to use all the activities for very simple problems (as indicated by a learner: "*I don't think it supports me that much because sometimes when you program the program doesn't run. I find this frustrating and nerve-racking*"), they were able to use it for the complex problems. *Overcoming their difficulties* emerged as an important factor, in terms of firstly not giving up easily, then researching information by reading further on aspects that were not clear (e.g. understanding the meaning of the range midpoint), and finally discussing the problem with their peers.

Of the 96 responses received in the second questionnaire, only 10 students obtained the correct answer to the given problem (see Appendix: post-problem). However, 44 students believed that they had obtained the correct answer or solved it partially, 36 were not certain that they had solved the problem and 16 students believed that they had not solved the problem. The majority (89%) of the students were able to overcome the difficulties that they experienced and were willing to use the strategy. Only 10 students were uncertain about the success and confidence attained in using the strategy. Even though they were not sure, there is still an element of potential use for the improvement of problem solving in programming.

Some quotations from students who were unsure of how these activities helped them are reproduced below.

> I don't think these problem-solving activities help, I just prefer solving the problem directly by typing the program. But they do help sometimes when trying to create a program that performs multiple functions but it is time consuming.
>
> No because sometimes the problem will be too demanding from you and it would sometimes require you to think very hard. Sometimes problem-solving activities are confusing and require your full concentration and cooperation.

While a small percentage (10%) of students obtained the correct answer in the post-questionnaire, the improved self-belief in problem solving was achieved. This improved self-efficacy is evident from the responses ascertained in the pre- and post-questionnaires. Although only a few more (6%) students managed to solve the problems after, their perceived self-efficacy in programming has increased considerably – a step towards improving ability. The small number of correct solutions may be attributed to the problem given in that many students did not understand the meaning of range midpoint. Improved performance outcome may not be immediately evident, but according to literature (e.g. Zimmerman, 2000), improved self-efficacy is one of the many factors that can help improve efforts and in turn improve performance outcome in the longer term.

6.3 Teachers' experiences of using the strategy to teach problem solving in programming

The interview transcripts suggest that teachers were also pleased with the framework used to teach problem-solving. Some excerpts from teachers' interviews and journals follow:

> …I see the difference in using the PS strategy…[T1]
>
> 'Yes, I think IT teachers would benefit from the support [we received]. It goes back to problem solving skills'. [T2]
>
> 'It was two-way, I was learning with my learners while I was implementing this [PS strategy]'. [T3]
>
> 'I think this is the strategy that I will use when I introduce problem solving'. [T4]
>
> 'I find it very interesting.'[T5]
>
> 'The method I was using before was difficult, now it is much easier and they are happy'. [T6]

Teachers found the framework to be helpful. It forced the students to plan before typing on the computer. The fourth quotation above indicates willingness on the part of the teacher to use this approach not only during the course of the project but to continue doing so. The excerpts above further suggest that teachers are motivated to use this strategy, implying that teachers' self-efficacy in teaching problem solving also increased. As indicated earlier, if teachers' self-efficacy increases in teaching programming then it will have an influence on students' behaviour. The transcripts of teacher interviews and the journal entries helped to triangulate the results.

7. Conclusion

This study explored the influence of teaching problem solving explicitly to Grade 10s in a programming course within the subject IT on students' self-efficacy to program. The

teaching of problem solving explicitly was shown to have benefits for self-efficacy in programming as perceived by the study participants. Bandura (1986) affirms that people's beliefs about their capabilities shape the ways in which they will work. Stated differently, these self-perceptions of capability help determine what individuals do with the knowledge and skills they have gained. In short, a high sense of self-efficacy in problem solving tends to motivate individuals to persevere and increase their efforts in achieving success in programming. Certainly these beliefs or self-perceptions do not necessarily determine whether success will be achieved or not, but they do have an influence. More specifically, the teachers who have undergone the training and who have implemented the strategy also showed a remarkable interest in the use of the strategy. The effect of this intervention programme was two-fold. First, these teachers have gained more insight into their problem-solving strategy and thereby increased their self-efficacy to teach. As noted earlier, teachers' self-efficacy will influence the way they teach and persevere in the face of difficulties. Secondly, students showed improved self-efficacy towards problem solving after the strategy was used for a period.

Problem solving in programming is a major part of the curriculum of the subject IT. It would serve the subject well to foster students' self-efficacy in programming by targeting their problem-solving skills. Teaching problem solving explicitly should be encouraged. It may benefit students' self-efficacy, potentially improving programming ability given more time.

8. Limitations

IT in schools is an elective course. Students taking the selected IT classes likely did so because of an interest in the subject matter or parents or peers influence. The problem-solving strategy was employed for a short duration given the time constraints of the project, which might affect the full impact of improved self-efficacy in problem solving. Given a longer timeframe and the use of experimental methods, it would be possible to assess the impacts of the strategy on achievements of the students in programming.

References

Askar, P. & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming Among Engineering Students. *Turkish Online Journal of Education Technology* 8(1): 7.

Bandura, A. (1986). *Social foundations of thought and action: A social-cognitive view.* NJ: Englewoods Cliffs: Prentice-Hall.

Bandura. A. (1994). Self-efficacy. In V. Ramachaudran (ed) *Encyclopedia of human behavior* 4: 71-81. New York: Academic Press.

Bandura, A. (1997). *Self-efficacy: The exercise of control.* New York: Freeman.

Bandura, A. (2006). *Self-Efficacy Beliefs of Adolescents.* Information Age Publishing.

Bandura, A. & Locke, E. (2003). Negative Self-Efficacy and Goal Effects Revisited. *Journal of Applied Psychology* 88(1): 87-99.

Creswell, J. (2003). *Research Design: qualitative, quantitative and mixed methods approaches.* 2nd ed. Beverley Hills, CA: Sage Publications.

Downes, T. & Looker, D. (2011). Factors that influence students' plans to take computing and information technology subjects in senior secondary school. *Computer Science Education* 21(2): 175-199.

European Commission,(2010). *Teachers' Professional Development — Europe in international comparison: An analysis of teachers' professional development based on the OECD's Teaching and Learning International Survey (TALIS).* Luxembourg: Office for Official Publications of the European Union.

Ford, M. & Venema, S. (2010). Assessing the Success of an Introductory Programming Course. *Journal of Information Technology Education* 9: 133-147.

Gibbs, G. (2010). *Analyzing Qualitative Data.* Los Angeles: Sage.

Govender, I. (2010). From Procedural to Object-Oriented Programming (OOP) — Performance in OOP: An empirical study. *South African Computer Journal* 46: 12.

Govender, I. & Govender, D.W. (2012). A constructivist approach to teaching a programming course: Students' responses to the use of an LMS. *African Journal of Research in Mathematics, Science and Technology Education* 16(2): 12.

Govender, D.W., Govender, I., Breed, B., Havenga, M., Mentz, E., Dignum, F. & Dignum, V. (2013). Supporting Information Technology Teachers through Programming Professional Development: A South African Case Study. *Journal of Communication* 4(2): 153-160.

Hackett, G.&Betz, N.E. (1989). An exploration of the mathematics self-efficacy/mathematics performance correspondence. *Journal for Research in Mathematics Education* 20: 261-273.

Hasni, T.& Lodhi, F. (2011). Teaching Problem Solving more Effectively. *SIGCSE Bulletin Inroads* 2(3): 58-62.

Ismail, M. Ngah, N. & Umar, I. (2010). Instructional strategy in the teaching of computer programming: A need assessment analysis. *Turkish Online Journal of Educational Technology* 9(20): 125-131.

Jegede, P.O. (2009). Predictors of Java Programming Self-Efficacy among Engineering Students in a Nigerian University. *International Journal of Computer Science and Information Security* 4(1 & 2): 7.

Kinnunen, P. & Simon, B. (2012). My program is ok — am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education* 22(1): 1-28.

Lau, W.W. & Yuen, A.H. (2009). Toward a Framework of Programming Pedagogy. *IGI Gobal* 3772-3777.

Maddrey, E. (2011). The Effect of Problem-Solving Instruction on the Programming Self-efficacy and Achievement. *Proquest Dissertations and Theses*. Nova Southeastern University, Florida, USA.

Mentz, E., Bailey, R., Havenga, M., Breed, B., Govender, D.W., Govender, I., Dignum, F. & Dignum, V. (2012). The diverse educational needs and challenges of Information Technology teachers in two black rural schools. *Perspectives in Education* 30(1):71-79.

Pajares, F. (1997). Current Directions in Self-Efficacy Research. In M Maehr & P Printrich (eds)*Advances in Motivations and Achievements* (10: 1- 49). Greewich: CT: JAI Press.

Ramalingam, V. & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analysis of Novice Programmer Self-Efficacy. *Journal of Educational Computing Research*. 19(4): 367-386.

Saeli, M., Perrent, J., Jochens, W.M. & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education* 10(1): 73-88.

Seturaman, S. & Medley, M.D. (2009). Age and Self-efficacy in Programming. *Journal of Computer Sciences in Colleges* 25(2): 122-128.

Shell, D.F., Colvin, C. & Bruning, R.H. (1995). Self-efficacy, attributions, and outcome expectancy mechanisms in reading and writing achievement: Grade-level and achievement-level differences. *Journal of Educational Psychology* 87: 386-398.

Zimmerman, B.J. (2000). Self-Efficacy: An Essential Motive to Learn. *Contemporary Educational Psychology* 25: 82-91.

Appendix A
Pre Problem

You are requested to calculate the total cost of 321 photocopies. 13 pages consist of one picture per page. Seven of the pictures must be printed in colour. For 100 or less copies, a page costs 15 cents, and for more than 100 but fewer than 200 copies, the first 100 copies cost 15 cents per page and thereafter 12 cents per page. If you want to duplicate 200 or more copies, the first 100 copies cost 15 cents per page, the next 100 copies 12 cents per page and the remainder 8 cents per page. The cost of photocopying pictures is R1.50 per page and coloured pictures cost R2.50 for each page.

Post Problem

Design an algorithm to compare the average of a list of numbers with its range midpoint. The range midpoint is calculated by determining the largest and smallest number in the list and averaging them. Both the average and the range midpoint must be displayed with a message indicating which is larger.