Metacognitive and Problem-Solving Skills to Promote Self-Directed Learning in Computer Programming: Teachers’ Experiences

Marietjie Havenga
North-West University
Potchefstroom Campus
South Africa
Telephone: (+27) 18 299 4281
Email: marietjie.havenga@nwu.ac.za

Betty Breed
North-West University
Potchefstroom Campus
South Africa
Email: betty.breed@nwu.ac.za

Elsa Mentz
North-West University
Potchefstroom Campus
South Africa
Email: elsa.mentz@nwu.ac.za

Desmond Govender
University of KwaZulu-Natal
Durban
South Africa

Irene Govender
University of KwaZulu-Natal
Durban
South Africa

Frank Dignum
Utrecht University
Utrecht
The Netherlands

Virginia Dignum
Delft University
Delft
The Netherlands
Abstract

This article reports on Information Technology (IT) (Computer Science) teachers’ experiences regarding the promotion of self-directed learning (SDL) in computer programming through the application of metacognitive and problem-solving skills. Programming is complex and students experience difficulty in solving programming problems as these require various high-level skills. One of the aspects that direct the solution of such problems may be substantiated by students’ self-directedness where they take responsibility for their own learning. We developed metacognitive and problem-solving guidelines with the aim to support teachers in promoting SDL as part of an intervention programme. Six teachers were selected as part of a case study in economically deprived schools that offered IT to Grade 10 students. Data collection comprised semi-structured interviews with teachers before and after applying the intervention. Teachers’ classroom experiences after the intervention included students’ detailed planning, a higher level of understanding, management of students’ thinking processes and their active involvement in programming. Our findings suggest that the teaching of metacognitive and problem-solving skills could provide opportunities and benefit students to manage their own learning. We proposed some patterns of meaning that could give insight into the development of a self-directed learner in computer programming.

Keywords: Computer programming, information technology, metacognition, problem solving, self-directed learning, teachers

Introduction and Problem Statement

According to the United Nations Educational, Scientific and Cultural Organisation (UNESCO, 2004), quality is the heart of education with the specific focus of Education for All. Quality also determines “... how well students learn ...” (UNESCO, 2004:17, 18). Meaningful learning is more comprehensive than being knowledgeable or obtaining content knowledge. It does not only refer to the end product but also to the dynamic process of obtaining and managing knowledge. One of the aspects regarding how well students learn may also be substantiated by their self-directedness. The main objective of self-directed learning (SDL) is to take responsibility for one’s own learning, which refers to the self-motivated quest for knowledge and management of learning processes. Individuals who take responsibility for their own learning, learn more and better than those who do not (Knowles, 1975). Self-directed students manage their own thinking processes, they explore opportunities for deeper learning and address problems as challenges (Guglielmino, 1977:73). Self-directed learning is an integral part of developing quality education. Some activities that enhance self-directed learning include, among others, learning how to solve problems and applying metacognitive skills. Metacognition enables the student to manage
his or her ‘knowing’ in order to be a responsible, independent and dedicated learner (Schraw & Dennison, 1994:460) - characteristics that contribute towards a more self-directed learner. These qualities are especially important when solving programming problems that require metacognitive control and reflection to guide the effective solution of problems.

The purpose of this research was to explore Information Technology teachers’ experiences regarding the use of metacognitive and problem-solving skills in computer programming to promote the development of self-directed learning. The main research question was: How can teachers’ apply metacognitive and problem-solving skills in computer programming to promote self-directed learning?

To address this question, the article is structured as follows: an overview is firstly given of the conceptual-theoretical framework on which the empirical investigation is based. This is followed by the empirical investigation and discussion of the findings and finally the conclusion is given.

**Conceptual-Theoretical Framework**

**Computer programming**

One of the main aims of the school subject Information Technology (IT) is to use appropriate activities to plan solutions, devise algorithms and solve programming problems through logical and creative thinking by using appropriate techniques and tools (Department of Basic Education, 2011:6, 8). The development of programming solutions requires a combination of precision and artistic prowess to produce the desired program or software product. IT students are required to apply a specific programming approach such as object-oriented programming, comprehend and analyse programming problems, design solutions and integrate structures and constructs as specified by the programming language in use.

Although students may enrol for IT as an elective in schools, they experience this subject as difficult, especially the programming part. Since programming is complex (Sajaniemi, Kuittinen & Tikansalo, 2007:1; Govender & Govender, 2012:238), it requires additional thinking skills beyond knowledge and codification. One of the problems is that students mainly focus on the product of programming namely on computer programs, rather than on the process of programming. This is evident from our experiences in class, as novices usually start to program without thinking deeply about the problem and planning their solution. Their programming attempts result in repetitive actions of debugging, changing of code and addressing of (some) error messages, if possible.
Teachers play an important role in directing and facilitating teaching and learning activities in class (Govender & Govender, 2012:250). Computer programming is not an exception. Teachers therefore need to scaffold students to take responsibility for their own learning as they plan and manage their thinking processes and direct their own programming activities.

**Self-directed learning and metacognition**

Self-directed learning refers to a process where individuals manage their own learning, identify learning needs, formulate learning goals, implement appropriate learning strategies, apply the required resources and evaluate their learning outcomes (Knowles, 1975:18). The application of SDL skills offers a way to enable students to ‘help themselves’ and to adapt to requirements and changes (Francom, 2010:32). One of Francom’s general principles for fostering SDL skills is to provide for both subject-matter knowledge and self-directed learning skills together.

Metacognition is intrinsically part of SDL and refers to the conscious planning, control and evaluation of one’s own cognitive processes, such as our own thoughts that engage in learning processes (Sternberg & Sternberg, 2012:234). Veenman, Van Hout-Wolters & Afflerbach (2006) distinguish between metacognitive knowledge and metacognitive skills where the former refers to a person’s declarative knowledge and the latter comprises a person’s procedural knowledge for example, how to solve problems. Metacognitive knowledge involves knowledge of a person, knowledge of a task and knowledge of different strategies (Flavell, 1979). Metacognitive skills such as monitoring comprise an individual’s ability to assess the state of his or her cognitive activity whereas metacognitive control refers to an individual’s ability to regulate cognitive activity (Miller & Geraci, 2011:303). Metacognitive control refers to activities that help students to manage their learning and/or problem solving (Schraw, 2001:4) and relates to the planning, monitoring and evaluation of the learning process (Hartman, 2001:35). Examples of metacognitive skills are motivation, goal orientation, knowledge of one’s own strengths and weaknesses as well as judgement and beliefs about personal learning (Ertmer & Newby, 1996:6).

Brown claims that metacognition should be taught to enable students to be strategic, to monitor their thinking processes and to be effective (Brown, 1992:149). According to Veenman et al. (2006:8, 9), teachers’ instruction of metacognitive skills and the provision of feedback appear to enhance the metacognition and learning of students. They mention three main principles regarding successful metacognitive instruction, namely embedding metacognitive instruction in the subject matter, introducing
students to the usefulness of metacognitive skills and the continuous teaching to ensure that students will apply metacognitive skills (Veenman et al., 2006:10).

The interplay between metacognition, problem solving and computer programming

According to Sternberg and Sternberg (2012:487), general problem-solving steps include: problem identification, problem definition, strategy formulation, organisation of information, allocation of resources, monitoring and evaluation. The solving of programming problems requires similar steps as well as the codification thereof. It comprises the implementation of abstract ideas, plans and/or designs by means of expressions, statements and programming constructs (syntax) in such a way as to resemble the correct logic and meaning of those expressions, statements and constructs (semantics) (Sebesta, 2013:134) and to solve the programming problem effectively.

The role of metacognition in the solving of programming problems is imperative. Bergin, Reilly and Traynor (2005:81, 82) found that students who perform well in programming use more metacognitive-management strategies than lower-performing students. In fact, the more complex a programming problem is, the greater the need for metacognitive control, purposeful reflection and positive feedback (Havenga, 2011:96). A programmer needs to apply in-depth reading skills and meta-comprehension to judge how clearly and effectively he or she understands the programming problem. Furthermore, programmers need to direct their problem-solving processes, apply a programming approach, correct programming errors, think deeply about their programming solutions and test program output. These problem-solving steps require metacognitive control such as planning (planning the solution), monitoring (monitor the design and development of the program) and evaluation (test and reflect on the programming solution). Students should therefore manage their programming processes, motivate their decisions, articulate their actions and investigate alternative solutions to improve the quality of their programs. The teacher has thus the responsibility to support students in developing metacognitive skills and applying these during problem solving and program development.

The next section explores IT teachers’ experiences regarding the purposeful application of metacognitive and problem solving skills that may provide opportunities for students to direct their own learning and programming activities.
Empirical Investigation

Methodology

We applied a qualitative research approach and employed a longitudinal case study design to determine what happens in response to a change (Gill, 2011:121, 130, 427), such as teachers experiences when implementing the intervention programme with the aim to promote self-directed learning in programming students.

Selection of participants

Information Technology teachers in economically deprived schools in two provinces in South Africa were identified to be part of this research. Since there are not many schools in these rural areas that offer IT as a subject, only six teachers who taught Grade 10 students were selected as participants in this study. This selection included three teachers at schools in North-West and three teachers in KwaZulu-Natal respectively. Except for one teacher (with an MSc degree in Computer Science), all participating teachers had either enrolled for a PGCE (Post-Graduate Certificate of Education) or had a BEd qualification with specialisation in IT. All of these teachers had at least three years of IT teaching experience.

Research ethics

Participation was entirely voluntary and all teachers completed informed consent forms. Permission to conduct this research was obtained from the North-West Department of Education, KwaZulu-Natal Department of Education, the ethics committee of the university where the study was conducted, as well as from school principals.

Data collection activities

Data collection involved semi-structured interviews with teachers to determine their initial teaching activities, an intervention by applying specific metacognitive and problem-solving guidelines and semi-structured interviews following the intervention.

Initial interviews

These interviews focused on teachers’ class activities to determine whether they taught any metacognitive and problem-solving skills prior to the implementation of the intervention programme, for example the
management of programming tasks. The interviews were recorded to ensure that everything mentioned during the discussion was preserved.

Intervention

During the intervention, the teachers were trained to follow specific problem-solving guidelines as well as metacognitive skills to direct the effective solution of programming problems. Note that knowledge of problem solving and metacognition are integrated in all steps. The following guidelines were taught as part of the intervention programme:

1. Read the problem, underline the main ideas, comprehend and write down the main requirements of the problem. Revise and articulate what the programming problem is all about.
2. Represent the solution and monitor your comprehension thereof.
3. Plan the detailed steps (purpose, input, processing and output) and mention the purpose and processes of each section or method. Reflect on your planning of the solution and motivate your decisions.
4. Code your planning in a programming language, evaluate your program, correct programming errors and articulate your actions.
5. Test your program, reflect on your programming code as well as programming semantics, explain how well you have solved the problem, and discuss whether this is the best solution.

Interviews after the intervention

The subsequent interviews focused mainly on teachers’ experiences with regard to the following question: How would you describe your experience regarding the use of metacognitive and problem-solving skills as part of your teaching approach?

Qualitative data analysis

After preparing the interview transcripts for data analysis, the textual information was manually coded according to a priori codes to indicate pre-defined main ideas such as aspects of metacognition and problem solving. These codes were identified based on the use of metacognitive skills (such as planning, management of thinking processes and reflection), and problem-solving skills (e.g. problem comprehension, analysis of the problem, solving the problem and program testing). The codes were
categorised and refined to represent the following themes, namely Teaching of metacognitive processes and Facilitation of problem-solving activities.

**Researcher’s role**

Although we visited the mentioned schools, the primary researcher compiled the problem-solving guidelines, explained the application thereof to teachers and analysed the semi-structured interviews.

**Findings and Discussion**

**Discussion of the findings**

This section discusses the findings and attempts to answer the main research question, namely: How can teachers’ apply metacognitive and problem-solving skills in computer programming to promote self-directed learning? In addition, we propose some patterns of meaning regarding the integration of metacognitive and problem-solving skills to emphasise their supportive role in programming.

**Theme 1: Teaching of metacognitive processes**

Teachers mentioned the use of some metacognitive skills before the intervention. Participant 1 [P1] provided an opportunity for students to plan their solution before starting to code: *The first period, I have to mark the paper [assignments] before they [students] can go and code. They fix errors on a piece of paper before they can apply it on the computer.* In contrast, Participant 4 relied on students to ‘discover’ details regarding new topics and assumed that these students could direct their own thinking processes. He also mentioned, … *I always encourage students to be creative.* Participant 6 referred to the use of additional strategies such as problem-based learning and collaborative learning, which are both strategies that may enhance self-directed learning.

After implementing the guidelines from the intervention programme, teachers outlined examples of metacognitive skills that they taught in class. Participants 1, 2 and 5 referred to the importance of students planning their solutions before writing programs. According to P1, *the more you plan, the more you understand the question … they can meet the demands of the question easily. I think some managed too.* Students are required to ask themselves, … *so in every question now: what am I supposed to do?* P2 claimed, *here the planning is very important. It emphasises the particular point to plan before they write the program.* The first two teachers referred to students’ active involvement in their programming task. P3 scaffolded students’ mental activities when he taught them to refocus on the problem in hand, while Participant 4 emphasised the importance of strategies as he mentioned … *It gave them a good
understanding in terms of how to tackle a problem. Participant 6 supported students by grouping them, and requiring of students to discuss their problems. Our findings following the intervention are in line with Veenman et al. (2006) and Francom (2010:35, 36) since they mentioned that the provision of subject-matter knowledge and self-directed learning skills (e.g. metacognition) together support students to manage their thinking processes.

Theme 2: Facilitation of problem-solving activities

Teachers taught specific problem-solving activities before the intervention. Participants 1 and 2 focused on question analysis and requested students to determine their input, processing and program output during the planning of their programs: … they draw an IPO [input, processing, output] table and then they write the steps [P2]. In addition, P1 and P2 emphasised the use of algorithms as part of the detailed planning of their programs. It becomes easier if you are going to start with [an] algorithm, I deal with practical examples, real life situations and then we try to analyse it [P2]. Although P3 and P5 gave students homework they did not mention specific problem-solving activities used before the intervention. Participant 6 discussed some assistance to students regarding how to analyse programming problems.

After the intervention, the participating teachers emphasised detailed requirements during each problem-solving step. P3 referred to the following: Once you just break it down into simple steps, they [students] start to catch on and like it because it is something that they can relate to. Furthermore, use of the Guidelines supported students: It forces them to think critically about issues, not just memorising the program [P3]. Participant 4 made an interesting observation, namely, They have to know everything about the first step to move to the next step until you reach the solution. This teacher also discussed some strategies such as how to approach a programming problem. P5 emphasised that students need to take time when analysing a problem, It does not matter if they spend a long time analysing the problem, because they cannot develop a solution if they do not understand. Our findings are in line with Bergin et al. (2005:81, 82) as they claimed that teachers’ instruction of management processes and activities enabled programming students to succeed and enhance their performance.

Emerged patterns of meaning

We deduced some patterns of meaning from our findings that may describe the interplay between problem-solving and metacognitive skills and their role in the development of a self-directed learner in computer programming. In this regard, we employed the term “cohesion” to indicate the integration and harmony between these two activities. There needs to be cohesion – or a degree of interaction – between
the use of metacognition and problem solving when teaching computer programming. The following types of cohesion are distinguished, namely low cohesion, medium cohesion and high cohesion. Note that our purpose was not to quantify the strength of interaction but rather to focus on the interplay between problem solving and metacognition.

First scenario: Low cohesion

Although the participating teachers used some problem-solving skills they did not focus on the meaningful integration of metacognition during programming. Participant 2’s teaching approach – from the initial interviews – may be an indication of this scenario since he required students to use an IPO table and mentioned that they need to plan and put more effort on the coding part, without referring to detailed guidance and management strategies. P2 only mentioned that they draw an IPO table and then they write the steps. Responses by P3, P4 and P5 from the initial interviews indicated similar patterns. P3 gave students homework and explained each line without directing students’ thinking processes to analyse each line themselves. Participant 4 asked students to discover their own knowledge, to be creative (see Theme 1) and coming up with their own ideas after discussing a new topic, while P5 gave students class work without teaching explicit monitoring and controlling of their thinking activities. In this scenario, metacognitive knowledge, control and reflection are so-called background activities without the impact of such activities on problem solving and programming. This scenario is represented in Figure 1, where a curved line indicates the use of some problem-solving (PS) steps (referred to in the intervention), as marked with numbers 1 to 5 and the dotted linear line represents background metacognitive skills (MC).

Second scenario: Medium cohesion

This scenario comprises the use of problem-solving activities with some examples of metacognitive skills, as mentioned by P1, P4 and P6. The first teacher required students to go back and check their input, processing and program output. Furthermore, P1 emphasised after the intervention the use of metacognitive management skills and reflective self-questioning as well as students’ active involvement in their task. As planning is also part of metacognitive management, P1 mentioned the more you plan, the more you understand the question. P4 supported student cooperation during programming as well as the
requirement that they had to know everything about the first step in order to move to the next step until reaching the solution. Analysis of P6’s interviews indicated that he asked students to read and understand the problem and write an algorithm. In addition, he also applied the use of collaborative and problem-based learning and focused on real-life problems and situations. Since some of the students lacked strategies on how to solve a problem, Participant 6 discussed how to tackle a problem as part of the intervention programme. Although the participating teachers referred to some examples of management and reflection, this type of cohesion does not necessarily link metacognitive skills to specific problem-solving steps but rather focuses on a general way of supporting students (see Figure 2). The second scenario is represented in Figure 2, where MC is not necessarily “in phase” with PS.

![Figure 2: Medium cohesion between PS and MC](image)

Third scenario: High cohesion

By using this scenario, specific metacognitive skills are taught “in phase” with the problem-solving and programming steps, as outlined in the intervention programme. This implies that where high problem-solving and programming knowledge and skills are required, a high level of metacognitive monitoring and control may scaffold these processes. Although the participating teachers applied some teaching activities as part of the intervention, our findings indicate that they mainly applied problem-solving and metacognitive skills as two distinctive rather than integrated activities. Some reasons may include the following:

- the explicit teaching of metacognitive skills requires detailed lesson planning to integrate these during class activities; and
- teachers require more time to implement integrated problem-solving and metacognitive skills.

In this regard, Francom (2010:35, 36) refers to the principle of progression from teacher to learner direction where students progress towards higher self-direction over a period of time. Teachers are required to further guide students in this challenging field of study and enable them to monitor and control their thinking processes. Figure 3 represents high cohesion between problem solving and metacognition in the sense that these activities are ‘in phase’ with each other to optimise their effective synergistic use and provide for opportunities to develop and support programming students to be self-directed.
It can be deduced that, although the use of metacognitive skills and strategies is an essential part of learning various subjects, their optimal application may be subject-specific in terms of the unique characteristics of each subject, as indicated in the three scenarios of cohesion. This means that in computer programming, metacognition forms an integral part of each step of understanding the problem, representing the solution, planning the detailed programming steps, coding the planning in a programming language and testing the program.

**Conclusion**

This investigation was aimed at exploring how IT teachers applied both metacognitive and problem-solving skills to enhance self-directed learning. Some activities emerged from teachers’ classroom experiences after the intervention, namely a higher level of understanding and management of students’ thinking processes during programming. Our findings suggest that the teaching of integrated metacognitive and problem-solving skills may provide opportunities and hold potential to enhance students’ programming performance and management of their own thinking processes. Furthermore, some patterns of meaning were outlined regarding the degree of integration and synergism between metacognition and problem solving that may contribute to the development of a self-directed learner in computer programming. We hope to motivate teachers in using integrated metacognitive and problem-solving patterns – as discussed in the third scenario – to enhance self-directed learning.

**Acknowledgement**

This research is based on work financially supported by SANPAD – the South African Netherlands Project for Alternative Development. Any opinion, findings and conclusions or recommendations expressed in this article are those of the authors, and therefore SANPAD does not accept any liability in this regard.
References


