Investigating the Role of Program Visualization and Neo-Piagetian Approach in Teaching and Learning of Programming  

by: Norizan binti Mohamad

Background

Computer programming is an essential skill that must be mastered by Computer Science students (Azwina & Rukaini, 2005). Yet, learning and teaching programming subject has been known as one of the grand challenges in computing education (Caspersen & Kolling, 2009). Many teachers agree that teaching beginners to program is difficult and this always has been and remains that way since decades.

While teachers are anxious to teach all about programming, students find computer programming difficult and struggle to master the concepts (Bergin & Reilly, 2005; Farkas & Murthy, 2005; Robins, Rountree, & Rountree, 2003). Particularly for novices, they claimed that computer programming subject is complex, difficult, often boring and far from the real world (Pyshkin, 2011; Yacob, Saman, & Yusoff, 2012). As a result, often the subject has the highest dropout rates (Robins, et al., 2003).

Surprisingly, in the past, students who repeat the programming subject for the second time or third time are rarely able to obtain highest grade. Indeed, this situation has puzzled the teachers. This is worsen when some students still do not know how to program at the conclusion of their introductory years (McCracken et al., 2001).

Why is learning programming hard for the students? In (Robins, et al., 2003), learning programming is considered as a cognitive process that relates to knowledge, attention, memory, problem solving and decision making which are important for human behavior. With regard to the individual, students need to have both the will (motivation) and the skill (capability) to be successful learners (Helme & Clarke, 2001).

Therefore, in order to teach more efficiently, teachers need to understand these two questions (Robins, et al., 2003): (1) “Are there successful and unsuccessful strategies for both learners and teachers?”; and (2) “What resources and processes are involved in understanding programming?”.

As a result, those engaged in the teaching of programming must consider how the learners learn and how they progress from one phase to the other in order to develop the programming skills (Baldwin & Kuljis, 2000).

Some people believe that the best programmers have strong mathematical background (Pyshkin, 2011), however giving too much computational mathematics for problem solving tasks does not guarantee that students are equipped with the necessary programming skills. There are still algorithmic analyses or theory of algorithms and its applications to teach to the students (Pyshkin, 2011). Consequently, one primary concern is “how to improve the approaches of teaching and learning of programming?”

Teaching and Learning Approaches

An improvement in programming learning approaches has been proposed by Yacob et al. (2012) using Total Quality Management principles. In it, problem-based learning was implemented which involves a constructivist approach to learning. Alternatively, writing programs is frequently referred to as an exercise in problem solving. McCracken et al.(2001) defined problem solving as a five step process: (1) abstract the problem from its description, (2) generate subproblems, (3) transform subproblems into subsolutions, (4) recompose, and (5) evaluate and iterate. Abstraction, therefore has been treated as an important aspect of programming since experts programmers operate at a high-level of abstract reasoning (Teague, 2012).

In addition, Kramer (2007) asserted that the key difference between top-performing and under-performing computing students is “The ability to perform abstract
thinking and to exhibit abstraction skills”.

**Neo-Piagetian Theory**

Recent research has proposed neo-Piagetian theory as a useful way of describing the cognitive development of novice programmers (Gluga, Kay, Lister, Kleitman, & Kleitman, 2013). Neo-piagetian development theory deals with abstraction and reasoning ability which are particularly relevant in computer programming (Gluga, et al., 2013). The use of neo-Piagetian has been proven beneficial in programming (Corney, Teague, Ahadi, & Lister, 2012; Lister, 2011) since neo-piagetian provides a way to describe the abstraction abilities of novice programmer.

It has been known that one of the causes of students' failure in programming subjects is their inability to visually illustrate the flow of program code during the program execution (Siti Rosminah & Ahmad Zamzuri, 2014). Using Neo-Piagetian approach helps to explain the reasons why novice programmers make little use of diagrams and exhibit a non-systematic approach to writing programs (Lister, 2011).

**Conclusion**

Programming demands complex cognitive skills such as reasoning and planning. In particular, understanding the cognitive development stages of the students is vital towards the acquisition of domain specific skills i.e. programming. As programming is the backbone of the technological advancement, students should be adequately competent with their programming skills. Applying neo-Piagetian approach and program visualization can enhance learners understanding in learning programming. Thus, greater understanding of the related theory such as Neo-Piagetian and its benefit on the teaching and learning of programming should be exploited by educators.

**References**


We welcome any feedback, suggestions, questions or comments from the readers. Email them to: norizanm@tganu.uitm.edu.my