International Journal on New Trends in Education and Their Implications
January 2014 Volume: 5 Issue: 1 Article: 10 ISSN 1309-6249

INTEGRATION OF VISUALIZATION TECHNIQUES AND ACTIVE LEARNING STRATEGY IN LEARNING COMPUTER PROGRAMMING: A PROPOSED FRAMEWORK

Siti Rosminah MD DERUS
Faculty of Art, Computing and Creative Industry
Universiti Pendidikan Sultan Idris,
35900, Tanjong Malim
Perak, MALAYSIA

Assoc.Prof. Dr. Ahmad Zamzuri MOHAMAD ALI
Faculty of Art, Computing and Creative Industry
Universiti Pendidikan Sultan Idris,
35900, Tanjong Malim
Perak, MALAYSIA

ABSTRACT

This paper reviews the issues and problems faced by students in learning programming, thus recommend a conceptual framework to overcome the problem. Computer programming courses are said to be complex and difficult, particularly to novice students. Among the causes of students' failure in developing programming skills is their inability to visually illustrate the flow of the program code during the program execution. To overcome this problem, a Program Visualization (PV) is recognized as one of the available learning support tools that can help novice students in enhancing their understanding of the programming execution. Nevertheless, using the PV alone without the active engagement with the tools will not produce the optimal learning outcome on students' programming performance. Previous studies indicated that, active learning strategies are among the most effective strategies in learning programming. Apart from learning strategies, there is a requirement of active involvement of students in the learning process, the ability to think logically which affect their ability to solve problems, thus lead them to develop a program. In addition, using PV as learning aids is expected to increase the students' self-efficacy in learning assignment activity and overcome the challenges of learning. Consequently, it is also important that these aspects are viewed in studies related to the effectiveness of any instructional materials such as PV to enhance programming performance, particularly in finding approaches that can improve novices' self-efficacy.

Key Words: Programming, program visualization, active learning, logical ability, self-efficacy.

INTRODUCTION

It was globally known that programming courses are generally regarded as complex and difficult, particularly to novice students (Bennedsen & Caspersen, 2007; Caitlin, Pausch, & Kelleher, 2003; A. J. Gomes & Mendes, 2010; Jenkins, 2002; Robins, Rountree, & Rountree, 2003). In fact, it is considered as one of seven grand challenges in computing education (McGetrick et al., 2005).

Many studies have been conducted pertaining to novices' difficulties to learn programming (A. Gomes & Mendes, 2007; Mohd Nasir, Nor Azilah, & Irfan Naufal, 2010a; Mow, 2008; Phit-Huan, Choo-Yee, & Siew-Woei, 2009; Reginamary, Hew, & Koo, 2009; Tuovinen, 2000). Based on these studies, among the problems frequently faced by them is a difficulty in understanding abstract programming characteristics as well as to relate the program development environment with real problems (Ala-Mutka, 2004). According to Winslow (1996), typically a novice approach programming "line by line" rather than using meaningful program structures. This factor is due to their limited knowledge to surface and superficial, resulting their inability to develop a perfect mental model for problem solution involving the programming development process

(Reginamary et al., 2009; Robins et al., 2003). Research finding showed that among the reason of student's difficulty in mastering programming skills was due to their inability in visualizing the flow of the programming process (Ahmad Rizal, Mohd Yusop, Abdul Rasid, & Mohamad Zaid, 2011; Milne & Rowe, 2002). Moreover, the problem worsens when static media were employed in teaching dynamic concepts of programming (A. Gomes & Mendes, 2007; Linden & Lederman, 2011). Therefore, an effective instructional strategy with the aid of appropriate learning support tool is crucial to ensure an optimal learning outcome (Ala-Mutka, 2004; Mow, 2008). Program Visualization (PV) is a learning support tool which can be visually assisting the students in understanding the behaviour of programs (Lahtinen, Ala-Mutka, & Järvinen, 2005; Mow, 2008; Yousoof, Sapiyan, & Khaja, 2005). According to Pears et al., (2007), humans have a better ability in processing visual information. It is therefore, PV could potentially aid students to understand the important elements related to program behavior during the execution of programs. Furthermore, visualization could also be used in assisting students to link new knowledge with old knowledge (Hyrskykari, 1993).

PROGRAM VISUALIZATION (PV)

Program visualization (PV) can be defined as a visual representation of program or algorithm execution in the form of graphical components. The main goal of PV is to assist students in understanding the dynamic behaviour of the program by displaying aspects like values of variables, evaluation of statements, and changes in the program state in general. In addition, PV could also explain visually the hidden processes during the program run-time (Bednarik, Moreno, Myller, & Sutinen, 2005). PV consists of two categories: dynamic PV and static PV (Rajala, Laaksi, Kaila, & Salakoski, 2008). Dynamic PV visualizes the flow of programming execution for every line of code for each value to the program variables used. An example of a dynamic PV tool is Jeliot3(Moreno, Myller, Sutinen, & Ben-Ari, 2004). While Static PV visualizes program structures and relations between program objects. An example of a popular static PV tool is BlueJ (Kölling, Quig, Patterson, & Rosenberg, 2003).

Programming involves numerous of implicit knowledge which lecturers are having constraint to explain explicitly with greater clarity and orderly, particularly the abstraction concept in programming (Fetaji, Loskovska, Fetaji, & Ebibi, 2007). Exist tension among novice students in understanding through static media such as text book, projected presentation, white board or verbal explanation (Fouh, Akbar, & Shaffer, 2012). Therefore, PV provides an environment which dynamically represent programming concepts for better understanding (Gomes & Mendes, 2007; Gračanin, Matković, & Eltoweissy, 2005). However, in developing PV, the role and limitation of students' cognitive ability need to be addressed throughout the development process. Developers need to understand how program visualization can be used to foster learning and they should not base the design on their own preferences that may or may not fit well with learners. For that reason, understanding how PV affects students' learning requires deep understanding of how information is processed in human memory structure. Design that do not consider students' cognitive ability will only lead to failure of the device to assist effective learning (Tudoreanu, 2003).

Though PV have strong arguments towards assisting learning, the excessive studies about the effectiveness of these teaching and learning aids show inconsistence results (Hundhausen, Douglas, & Stasko, 2002; Linden & Lederman, 2011; Sheard, Simon, Hamilton, & Lonnberg, 2009). One of the most recognized studies in the field of visualization in computer science education is the meta-study by Hundhausen et al. (2002). They conducted a study which analysed 24 educational experiments using visualizations. Only 46% (11 out of 24) of studies yielded significant results favouring the treatment group. Hundhausen et al. (2002) reported that in many of those evaluated experiments the focus was on the number of visualized components instead of how those visualizations benefited students' learning. Hundhausen et al. (2002) concluded that the passive usage (viewing) of visualization does not guarantee better learning performance and it is really important to engage and activate the learner with visualization during the learning activity. The importance of interaction between students and material is reported in the educational literature, in the field of multimedia learning which the addition element of interactivity to the multimedia learning enhanced the students performance (see e.g Evans & Gibbons, 2007; Mayer, Dow, & Mayer, 2003).

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

ACTIVE LEARNING STRATEGIES INVOLVING ENGAGEMENT OF STUDENTS

Active learning is generally defined as any method of teaching that involves students in the learning process (Prince, 2004). It requires students to do meaningful learning activities and think about what they are doing. In the theory of constructivism, knowledge is constructed by students as a result of the learning experience through which they had (Driscoll, 2005; Loyens & Gijbels, 2008). It means during the learning process, students will activate prior knowledge stored in long-term memory and try to connect with new knowledge learned (Loyens & Gijbels, 2008).

Programming skills cannot be obtained directly if they have not actively engaged in learning activities (Fetaji et al., 2007; Reginamary et al., 2009). Abstract concepts in programming can't be concrete unless students are given hands-on experience that will make the concept of being clear and also learned through their own experience (Parham, 2003). This is proven through studies carried out by Gonzalez (2006) that implement active learning strategy and cooperative learning in the introduction programming course (CS1), showed positive results in students' programming performance when 70% of the students successful while learning in advanced programming courses (CS2). Therefore, as PV is beneficial for students to comprehend the abstract concept of programming, it is essential to administer an active learning strategy in order to guide student to actively engage with the tool. This can be implemented by utilizing Engagement Taxonomy (ET) proposed by (Naps et al. (2002), the modes by which students could become active participants in exploring with a PV. ET describes different forms of engagement that can be promoted by a visualization tool and it provides testable hypotheses about how the engagement level of visualization affects the learning outcomes. The central idea of the taxonomy is a higher-level engagement between learner and the visualization results for better learning outcomes. The ET consists of six levels of engagement and they are described in Table 1. Nevertheless question arises for novice students since they do not have strong existing knowledge related to topics studied~~.~~ Without a strong pre-existing knowledge, active involvement may be limited. In this regard, to study the impact of the engagement level and programming learning performance would be of a great value and interest.

Table 1: Engagement Taxonomy Level (Source: Naps et al.,(2002))

No.	Engagement Level	
1	No Viewing	There is no use of visualization tools
2	Viewing	Considered the core of student engagement with the visualization tool. Students only view the behavior of program activities from screen display.
3	Responding	Learner interacts with visualization by responding to visualization's related questions.
4	Changing	Visualization or state of visualization can be altered.
5	Constructing	Learner can create own visualizations.
6	Presenting	Learner presents visualizations for discussion and feedback.

OTHERS FACTOR RELATING TO STUDENT'S PROGRAMMING PERFORMANCE

Aspect from the correct design and the right active learning strategy still does not guarantee students' programming performance. Factors such as student's perception and self-efficacy that they are able to complete tasks, control strategies and learning environment should also be taken into account (Ghazali, Nik Mohd, Parilah, Wan Haslina, & Ahmed Thalal, 2011). Self-efficacy is a belief in an individual's capability to organize and perform necessary actions required to attain designated types of performance (Bandura, 1977). Previous reseach conducted showed a significant relationship between student self-efficacy learning performance especially in the area of programming (Ramalingam, LaBelle, & Wiedenbeck, 2004). In relation to the effectiveness of any instructional materials such as utilizing PV to enhance learning programming, these are seen as important aspects of approaches, which could improve novice self-efficacy.

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

The relationship between problem solving and programming development should be the main focus of any introductory course in programming (Eckerdal & Berglund, 2005; Schneider, 1978). The core skill of problem solving is depending on competencies of students in making logical assumption (Ahmad Rizal et al., 2011). There is a correlation between students' logical skills with programming performance (Mohd Nasir, Nor Azilah, & Irfan Naufal, 2010b; Parham, 2003). Therefore, logical and analytical skills are function as an important elements in determining the ability of students to solve problems. Logical and analytical skills are crucial in order to understand the problem, to analyze the situation and review results of each step taken and, creativity is necessary for them to form solution to the problem.

Hence, in this study, a prototype PV with various active learning strategies is developed and designed specifically for novice to test their effectiveness of performance and self-efficacy. This study will also look at the impact of learning strategies aided by PV on students from different logical capabilities in detail. The acctive learning strategy is based on the engagement taxonomy levels as recommended by Naps et al., (2002). According to Naps (2005) which stated that the level of ET truly involves students actively participate is the highest of four levels. Therefore, this study will only focus on four active learning strategies based from the four highest levels of ET, which is: i) responding), ii) changing), iii) constructing and iv) presenting.

Theoretical framework

The theoretical framework of this study is grounded on Atkinson-Shiffrin Memory Model (Atkinson & Shiffrin, 1971) and Cognitive Load Theory (Sweller, 1988). Learning is a process which information is received, processed, encoded and retrievable from memory structure (Lin & Dwyer, 2004). Atkinson & Shiffrin, (1971) classifies memory storage into three types, namely sensory memory, working memory and long-term memory (Figure 1). Each of these memories has its own role in receiving and processing information. Typically, information stored in sensory memory is merely about ½ to 1 second, working memory will store information about 15 to 30 seconds and long-term memory will store information permanently (Ismail, 2011). New information can purely be stored in long-term memory after they first appear and process in working memory. However, as disclosed, working memory capacity is limited and the duration of information storage is very short. Therefore, not all information penetrated the memory structures will work successfully, and registered permanently as in the long-term memory, in the form of perfect schema (Chandler, 1995).

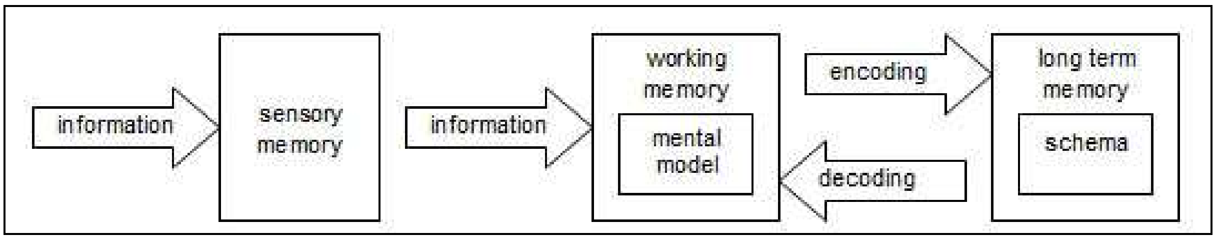

Figure 1: Information Flow In Memory System (Source: Atkinson & Shiffrin, 1971)

Cognitive Load theory (Sweller, 1988; 2002) describes the learning process from the perspective of information processing system that keep all information obtained from working memory in the long-term memory in the form of perfect schema. Scheme is a memory structure that allows students to deal with a large number of information blocks as if they are a single block. New information which enters the working needs to be integrated with pre-existing schemas in long term memory for sufficient mental model development in working memory. For this to take place, relevant schemas in long-term memory must be activated and decoded into working memory which is also defined as the process of schema aquisition. The inability of perfect mental models construction can be a barrier in the learning process, primarily to novice students who are not capable of recognizing the relevant pre-existing scheme in the learning process. Due to limited capacity of working memory, when student are overloaded with information and the complexity of teaching materials that are not well managed, it will be resulted in a cognitive load that potentially cause the process of loaded coding scheme perfectly affected (Jong, 2009; Sweller, 1988).

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

Cognitive load is divided into three different categories of: i) intrinsic cognitive load, refers to the level of complexity or difficulty of ~~the~~ information to be learned; ii) extraneous cognitive load, refers to the techniques and strategies used in presenting information ; and iii) germane cognitive load, refer to impulse formation scheme through learning content management and integration of educational content to student knowledge (Hasler, Kersten, & Sweller, 2007; Sweller, Merrienboer, & Paas, 1998; Sweller, 2010). Consequently, the development of instructional materials must be taken into consideration of principles which can reduce intrinsic cognitive load and extraneous cognitive load, and simultaneously will increase germane cognitive load (Caspersen & Bennedsen, 2007). Optimizing the learning outcomes is through by balancing the three cognitive loads, for instance to reduce intrinsic cognitive load and extraneous cognitive load which will indirectly increase germane cognitive load (Bennedsen & Caspersen, 2007; Sweller, 2010)..

Conceptual Framework

Based on theories, principles and literature overview, the study proposes a framework of utilizing a visualization technique for the purpose of learning the program as depicted in Figure 2. A prototyped PV developed specifically for novice students in understanding the basic concepts of programming. The visual information presented by PV dynamically is expected to help students to understand the key elements relating to program behavior during the program execution. Nevertheless, the PV developed should be balanced with the role and limitation of students' working memory capacity. This is because the visual information conveyed by PV should be able to help students develop accurate mental models in working memory for processing, thus creating the perfect scheme to be stored in long-term memory. Considering the limited capacity of working memory, if the visual information presented is not in line with the scheme which has been in existence in the long-term memory, this will cause cognitive load happens to students as they have to strive to understand the new information to be conveyed without associating it with existing knowledge. Therefore, the development of PV based on instructional design principles is crucial in this study to ensure the effectiveness of PV as a programming learning support tool.

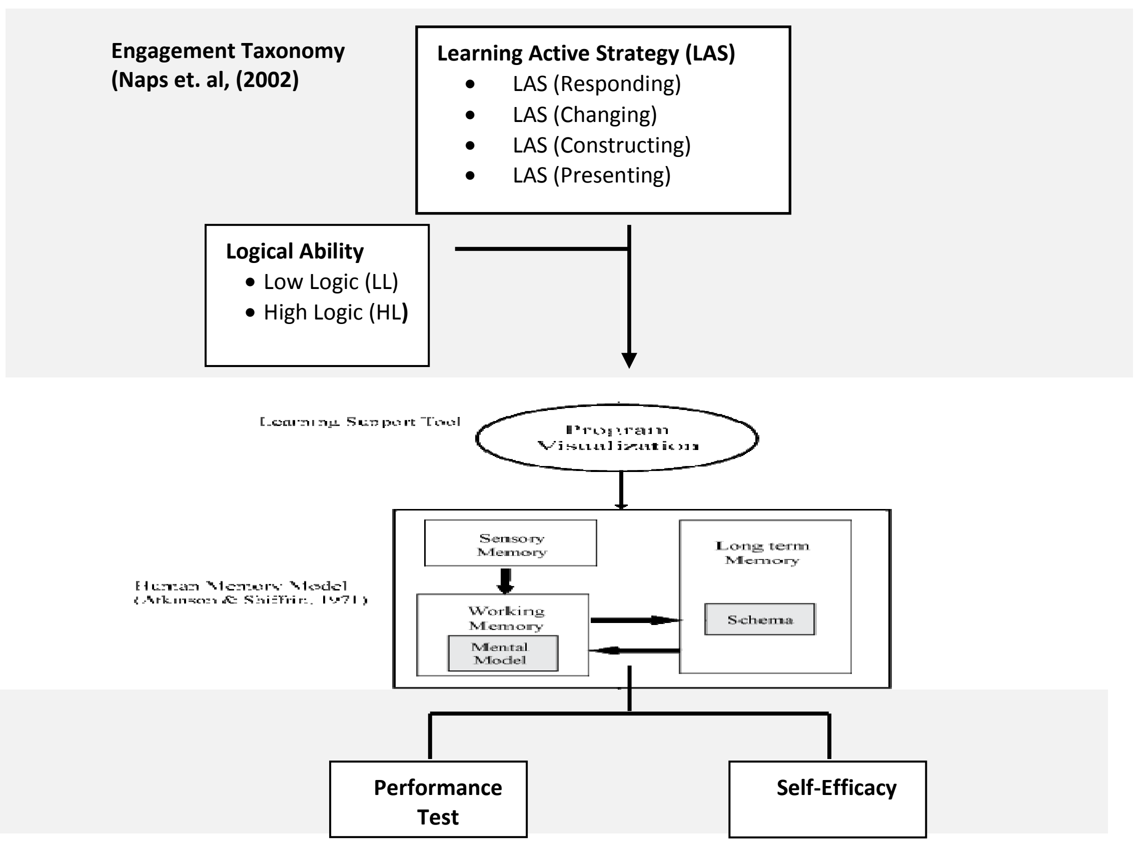

Figure 2: Conceptual Framework

Strategy and approaches the visual representation of information is also essential to ensure effective use of information in long-term memory. Inappropriate use of strategies will lead to increase a cognitive load during learning activities, particularly if the visual information from the PV only focuses the dynamic presentation of program execution. Students should be actively engaged with PV in order to active processing occurs in working memory. Consequently, this framework proposed the implementation of active learning strategies based upon the engagement taxonomy levels as recommended by Naps et al., (2002) with the central idea of ET that the higher the level of student engagement with PV, the better the learning outcome. However, does this assumption could effectively affect novice? Students will actively be involved if they have been existing knowledge related to the topics learned. Without pre-existing knowledge, active engagement is likely to be limited. Thus, it is important to study the impact of engagement level on the students' programming performance, especially among novices.

The development of education world nowadays does not only focus on learning performance. Consideration is given towards students' self-efficacy improvement of foreseeing them to carry out learning tasks and overcome the challenges of learning. Students with high self-eeficacy tend to learn and strive to achieve success as compared to students with low self-efficacy, although they have the same capabilities (Bandura, 1986). Thus, this study is important to observe whether through the active learning strategies approach aided by PV are able to increase students' self-efficacy, especially novice.

This framework involves the relationship of logical ability with the proposed active learning strategy. This is because as believed, student's logical ability is recognized as one of the important elements needed to solve programming problems during the learning activities. Thus, it is important to investigate whether the student's logical ability really affect the learning performance through the active learning strategies approach aided by PV.

CONCLUSION

Programming courses are closely related to skills comprises the set of sequence or order through the programming language which then is translated and executed by a computer. However, the subject is said to be difficult and complex. This is because novice students require a proper understanding to capture an abstract concept of the learning and difficult to imagine. Previous finding shows that amongst factor students' difficulty to master programming is because they could not visualize the programming process. Therefore, the designed and developed PV is function as the potential learning support tools to strengthen students' understanding of the programming learning process. PV using the approaches and techniques of dynamic performances in illustrating the changes to the program code and thus helping students grasp the changes that occurred to the program states. Nevertheless, using PV merely without the active engagement of students will not be able to give an effective impact on students' programming performance. Therefore, an active learning strategy is functions an approach that must be embedded in the learning process of programming. Through active learning strategies aided by PV, it is hoped that it will help to develop students' confidence in mastering the basic concepts of programming. There is still a thin layer lining in the clowd and we may expect to see the light at the end of the tunnel.

IJONTE's Note: This article was presented at World Conference on Educational and Instructional Studies – WCEIS 07- 09 November, 2013, Antalya-Turkey and was selected for publication for Volume 5 Number 1 of IJONTE 2014 by IJONTE Scientific Committee.

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

International Journal on New Trends in Education and Their Implications
January 2014 Volume: 5 Issue: 1 Article: 10 ISSN 1309-6249

BIODATA AND CONTACT ADDRESSES OF AUTHORS

Siti Rosminah MD DERUS is currently a PHD student in the Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Malaysia. She has a Bachelor degree in Electrical Engineering (Computer Technology) from Universiti Teknologi Malaysia, and a Master in Education from Universiti Teknologi Malaysia. Her research interest is instructional technology and programming visualization.

Siti Rosminah MD DERUS
Faculty of Art, Computing and Creative Industry
Universiti Pendidikan Sultan Idris,
35900, Tanjong Malim, Perak, MALAYSIA
E. Mail: ctrosminah@gmail.com

Ahmad Zamzuri MOHAMAD ALI is Associate Professor of Multimedia in the Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, Malaysia. He has a Bachelor degree in Electrical Engineering from Universiti Teknologi Malaysia, a Master in Education from Universiti Teknologi Malaysia and PhD in Multimedia Design from Universiti Sains Malaysia. His research interest is multimedia design, instructional technology, mobile learning, and integrated knowledge.

Assoc.Prof. Dr. Ahmad Zamzuri MOHAMAD ALI
Faculty of Art, Computing and Creative Industry
Universiti Pendidikan Sultan Idris,
35900, Tanjong Malim, Perak, MALAYSIA
E. Mail: zamzuri@fskik.upsi.edu.my

REFERENCES

Ahmad Rizal, M., Mohd Yusop, A. H., Abdul Rasid, R., & Mohamad Zaid, M. (2011). The Effect of Using Learning Model Based on Problem Solving Method on Students with Different Cognitive Style and Logic Ability. Educational Research, 2(9), 1498–1505. Retrieved June 22, 2013, from http://interesjournals.org/ER/pdf/2011/September/Madar et al.pdf

Ala-Mutka, K. (2004). Problems in learning and teaching programming-A literature study for developing visualizations in the Codewitz-Minerva Project. *Codewitz Needs Analysis*, 1–13. Retrieved January 22, 2012, from www.cs.tut.fi/~edge/literature_study.pd

Atkinson, R. C., & Shiffrin, R. M. (1971). *The control processes of short-term memory* (pp. 1–23). Stanford, California.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191–215.

Bandura, A. (1986). Social foundations of thought and action. Englewood Cliffs, NJ.

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

Bednarik, R., Moreno, a., Myller, N., & Sutinen, E. (2005). Smart program visualization technologies: planning a next step. *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 717–721.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in Introductory Programming. *ACM SIGCSE Bulletin*, *39*(2), 32–36.

Caitlin, K., Pausch, R., & Kelleher, C. (2003). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, *37*(2), 83–137. Retrieved january 15, 2002 from http://dl.acm.org/citation.cfm?id=1089734

Chandler, P. (1995). Is conventional computer instruction ineffective for learning? In *Australia Computers in Education Conference*. Perth. Retrieved Mei 3, 2012, from http://www.penta2.ufrgs.br/edu/telelab/6/chandler.htm

Driscoll, M. P. (2005). Psychology of learning for instruction. Boston, MA: Allyn & Bacon Publishers.

Eckerdal, A., & Berglund, A. (2005). What does it take to learn ' Programming Thinking '? In ACM (Ed.), ICER '05 Proceedings of the first international workshop on Computing education research (pp. 135–142).

Evans, C., & Gibbons, N. J. (2007). The interactivity effect in multimedia learning. *Computers & Education*, *49*(4), 1147–1160.

Fetaji, M., Loskovska, S., Fetaji, B., & Ebibi, M. (2007). Combining virtual learning environment and integrated development environment to enhance e-learning. In *Proceedings of the ITI 2007 29th Int. Conf. on Information Technology Interfaces* (pp. 319–324).

Fouh, E., Akbar, M., & Shaffer, C. a. (2012). The role of visualization in Computer Science education. *Computers in the Schools*, *29*(1-2), 95–117.

Ghazali, Y., Nik Mohd, R., Parilah, M. S., Wan Haslina, W., & Ahmed Thalal, H. (2011). Kepercayaan Jangkaan Keupayaan Kendiri Dalam Kalangan Pelajar Kursus Bahasa Arab (*Self-Efficacy Among Students Of Arabic Language Course*). *GEMA Online Journal of Language Studies*, *11*(1), 81–96. Retrieved July, 2, 2013, from http://ejournal.ukm.my/gema/article/view/69/63.

Gomes, A., & Mendes, A. J. (2007). Learning to program-difficulties and solutions. In *International Conference on Engineering*. Cimbra, Portugal. Retrieved Mei 3, 2012, from http://ineer.org/Events/ICEE2007/papers/411.pdf.

Gonzalez, G. (2006). A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In *SIGCSE '06 Proceedings of the 37th SIGCSE technical symposium on Computer science education* (pp. 133–137).

Gračanin, D., Matković, K., & Eltoweissy, M. (2005). Software visualization. *Innovations in Systems and Software Engineering*, *1*(2), 221–230.

Hasler, B. S., Kersten, B., & Sweller, J. (2007). Learner Control , Cognitive Load and Instructional Animation. *Applied Cognitive Psychology*, *729*, 713–729.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, *13*, 259–290.

Hyrskykari, A. (1993). Development of program visualization systems. In *2nd Czech British Symposium of Visual of Man-Machine Systems* (pp. 1–21). Prague,Czech.

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, pp. 53–58). Retrieved Mac, 6, 2012, from http://www.ics.heacademy.ac.uk/Events/conf2002/tjenkins.pdf.

Kölling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003). The BlueJ system and its pedagogy. *Journal of Computer Science Education*, *13*(4), 1–12. Retrieved November 12, 2012 from http://www.bluej.org/papers/2003-12-CSEd-bluej.pdf.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, *37*(3), 14.

Lin, C.L., & Dwyer, F. (2004). Effect of varied animated enhancement strategies in facilitating achievement of different educational objectives. International Journal of Instructional Media, 31(2), 185-199.

Linden, T., & Lederman, R. (2011). Creating visualizations from multimedia building blocks : A simple approach to teaching programming concepts. In *Information Systems Educators Conference -ISECON 2011* (pp. 1–10). Wilmington North Carolina. Retrieved September 3, 2012 from http://proc.isecon.org/2011/pdf/1619.pdf.

Loyens, S. M. M., & Gijbels, D. (2008). Understanding the effects of constructivist learning environments: introducing a multi-directional approach. *Instructional Science*, *36*(5-6), 351–357.

Mayer, S., Dow, G. T., & Mayer, R. E. (2003). Multimedia Learning in an Interactive Self-Explaining Environment: What Works in the Design of Agent-Based Microworlds? *Journal of Educational Psychology*.

McGetrick, A., Borle, R., Ibbett, R., Llyod, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing: Education-A Summary. *The Computer Journal*, *48*(1), 42–48.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching trogramming — Views of students and tutors. (D. Watson, Ed.)*Education and Information Technologies*, *7*(1), 55–66.

Mohd Nasir, I., Nor Azilah, N., & Irfan Naufal, U. (2010a). Instructional strategy in the teaching of computer programming : A need assessment analyses. *TOJET*, *9*(2), 125–131. Retrieved Mei 18, 2012 from http://tojet.net/articles/9214.pdf.

Mohd Nasir, I., Nor Azilah, N., & Irfan Naufal, U. (2010b). The effects of mind mapping with cooperative learning on programming performance, problem solving skill and metacognitive knowledge among Computer Science students. Journal of Educational Computing Research, 42(1), 35–61.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. *Proceedings of the working conference on Advanced visual interfaces - AVI '04*, 373.

Mow, I. T. C. (2008). Issues and difficulties in teaching novice computer programming. In *Innovative Techniques Technology, E-learning, E-assessment and Education* (pp. 199–204).

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., … Velazquez-Iturbide, J. A. (2002). Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02 Working group reports from ITiCSE on Innovation and technology in computer science education* (pp. 131–152). ACM.

Parham, J. R. (2003). An assessment and evaluation of Computer Science education. *Journal of Computing Sciences in Colleges*, *19*(12), 115–127. Retrieved Jun 15, 2012 from http://www.cs.clemson.edu/~jparham/parham_2003_publication.pdf

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., … Paterson, J. (2007). A survey of literature on the teaching of Introductory Programming. In *ITiCSE-WGR '07 Working group reports on ITiCSE on Innovation and technology in computer science education* (pp. 204–223). ACM.

Phit-Huan, T., Choo-Yee, T., & Siew-Woei, L. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. *2009 International Conference on Computer Technology and Development*, *2*, 42–46.

Prince, M. (2004). Does active learning work? A review of the research. *Journal of Engineering Education*, *93*(3), 223–231.

Rajala, T., Laaksi, M.-J., Kaila, E., & Salakoski, T. (2008). Effectiveness of Program Visualization: A case study with the ViLLE tool. *Journal of Information Technology Education: Innovatins in Practice*, *7*, 15–32. Retrieved Mac 10, 2012 from http://jite.org/documents/Vol7/JITEv7IIP015-032Rajala394.pdf.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '04* (p. 171). New York, New York, USA: ACM Press.

Reginamary, M., Hew, S. H., & Koo, A. C. (2009). Multimedia Learning Object to build cognitive understanding in learning Introductory Programming. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia -MoMM '09* (pp. 396–400). New York: ACM New York, NY, USA ©2009.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. Retrieved February 17, 2012 from http://www.tandfonline.com/doi/abs/10.1076/csed.13.2.137.14200.

Schneider, G. M. (1978). The Introductory Programming course in Computer Science : Ten principles. In SIGCSE '78 Papers of the SIGCSE/CSA technical symposium on Computer science education (pp. 107–114). ACM.

Sheard, J., Simon, S., Hamilton, M., & Lonnberg, J. (2009). Analysis of research into the teaching and learning of programming. In *5th International Workshop on Computing Education Research Workshop* (pp. 93–104).

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285.

Sweller, J. (2002). Visualization and instructional design. In *Proceedings of the International Workshop on Dynamic Visualization and Learning* (pp. 1501–1510). Germany. Retrieved January 18, 2012 from http://www.iwm-kmrc.de/workshops/visualization/sweller.pdf.

Sweller, J. (2010). Element interactivity and intrinsic , extraneous , and germane cognitive load. *Educational Psychology Review*, *22*(2), 123–138.

Sweller, J., Merrienboer, J. J. G. Van, & Paas, F. G. W. C. (1998). Cognitive Architecture and Instructional Design, *10*(3), 251–296.

Tudoreanu, M. E. (2003). Designing effective program visualization tools for reducing user's cognitive effort. In *SoftVis '03 Proceedings of the 2003 ACM symposium on Software visualization* (pp. 105–213). New York, New York, USA: ACM Press.

Tuovinen, J. E. (2000). Optimising student cognitive load in computer education. In *Proceedings of the Australasian conference on Computing education -ACSE '00* (pp. 235–241).

Winslow, L. E. (1996). Programming pedagogy- A psychological overview. *ACM SIGCSE Bulletin*, *28*(3), 17–22.

Copyright © International Journal on New Trends in Education and Their Implications / www.ijonte.org

Yousoof, M., Sapiyan, M., & Khaja, K. (2005). Reducing cognitive load in learning Computer Programming. *World Aademy of Sciene, Engineering and Technology*, 469–472. Retrieved February10, 2012 from www.waset.org/journals/waset/v12/v12-93.pdf