ACTIVE LEARNING FOR PROBLEM SOLVING IN PROGRAMMING IN A COMPUTER STUDIES METHOD COURSE

Edward Zimudzi
Department of Mathematics and Science Education,
University of Botswana, Gaborone,
BOTSWANA.
edward.zimudzi@mopipi.ub.bw

ABSTRACT

This paper presents an implementation of active-learning-based teaching model for teaching a topic on programming techniques in an undergraduate computer science education course which prepares students to teach the Botswana General Certificate of Science Education (BGCSE) computer studies subject. This programming topic is very crucial for developing lifelong skills in problem-solving and critical thinking skills; skills that are of crucial importance in the career of computing graduates. The topic has always been very difficult to master for pre-service computer studies students who have very little programming experience. We suggest this active learning approach for the reason that the students actively participate during the discussion, and the course tutor can easily identify the alternative conceptions that the students have, and be able to provide the necessary help to the future computer studies teachers. Active learning is a constructivist teaching approach that actively engages students in the learning process. The students learn problem-solving by doing, through a step-by-step process, and always build on what they already know previously. It uses different methodological interaction techniques, thereby improving student understanding of the programming concepts and the general motivation to learn more. We also discuss the role of the teacher in active learning approach.

Keywords: Active learning, programming techniques, problem-solving, constructivist learning theory, computer science education

INTRODUCTION

Among the aims of the BGCSE senior secondary school computer studies syllabus are for the students to be able:

1. To develop lifelong learning skills to be able to apply their ICT knowledge to solve real life problems.
2. To develop critical and logical thinking, self-reliance and initiative, which will serve as the basis for further training and positive work habits in the use of computers (BGCSE Computer Studies Syllabus.

These broad aims can be acquired through topics in the systems development life cycle and programming techniques. To increase the confidence in their future teaching of these topics at senior school, the student teachers have to be thoroughly prepared. They need to have hands-on experience especially with common programming constructs. Real-life problem solving using programming language is not an easy task, especially for novices to programming (Robins *et al.,* 2003). They need to thoroughly understand the syntax and semantics of a language to fully understand, and then convert that, using their own acquired mental models to fully convert their own understanding into computer code. The students should be involved in creating their computer-based problems, and providing possible solutions to them. This way, they are unlikely to benefit much from the study of the programming topics. The tutor would just guide the selection and help in designing and coming up with solutions. Most of the work should be done by the student.

Student teachers should be able to cover the design, development, implementation, maintenance and review principles, which include techniques and tools relating to the solution to a computer-based problem. The general objectives include:

a. Designing algorithms which relate clearly to the requirements of the system
b. Explain algorithms and how they relate to the system
c. Explain how hardware needs arise from the output required from the system
d. Use algorithmic tools like top-down design, structure diagrams, flowcharts,, libraries and procedures.
e. Interpreting and testing algorithms using dry runs and trace tables
f. Using pseudo code structures for selection structures, loops, input and output, counting and sequences.

Reinforcement of these studies is through practical work. These programming techniques have to be learnt through the use of many practical examples in everyday life. Although the emphasis is on structured programming techniques using pseudo code and flowcharts, the consolidation can only be through practical problem solution, and this practical solution will need the use of a programming language like Java or Visual Basic.NET that supports use of visual programming tools, both of which are widely available. These programming languages will provide necessary basic programming experience that is necessary for the teaching of senior school students, and will give the senior school students an edge if they start computing disciplines in programming courses at university.

CONCEPTUAL FRAMEWORK

Active Learning

Active learning is a process whereby students engage in activities, such as reading, writing, discussion, or problem solving that promote analysis, synthesis and evaluation of class content. It is a planned series of actions or events to invite the participant to process, apply, interact and share experiences as part of the educational process (McConnel, 1996). The interactive components support the goal and educational objectives for the learning activity. Active learning promotes reflection, problem solving, and critical thinking, manipulation of materials, analysis, synthesis and evaluation of the information (McConnel, 1996; Gao & Hargis, 2010; Hazzan, 2011).

Among the many descriptions of active learning, Silberman (1996) asserts that students figure things out by themselves, come up with examples, try out skills, and do assignments that depend on the knowledge they already have or must acquire. According to constructivists, learning is an active acquisition of ideas and knowledge construction, rather than a passive process. In other words, learning requires the individual to be active and to be engaged in the construction of one’s own mental models.

Active learning is widely accepted nowadays as a quality form of education (Silberman, 1996; Robins, *et al.,* 2003; Prince, 2004; Gao & Hargis, 2010; Hazzan, 2011; Lewis, 2011). Studies have shown that students prefer strategies promoting active learning to traditional lectures. Researchers agree that learning involves constructing our own meanings. They suggest the design of well-structured, cohesive material, and then encourage learners to actively engage with the material. Active learning is involving students directly and actively in the learning process itself. This means that instead of simply receiving information verbally and visually, students are receiving, participating and doing something, i.e. talking, listening to one another, writing and reading programs, and reflecting individually or in small groups.

Deep Learning

Our students are still novices to programming. They have just done a semester long introductory java course in programming. Literature on the learning of object-oriented programming languages and computer science education (Ben Ari, 2001; Gao & Hargis, 2010; Kinnunen & Simon, 2012; Lewis, 2011; Robins *et al.,* 2003, Thomson & Kinshuk, 2011) shows that it is not an easy task. It is one of those courses at university where average students struggle to pass. The course that they have done mainly focuses on object-oriented java language features. There is not much time for real-life application development and problem-solving. Our students have surface-level knowledge that they have got through memorization of isolated and unlinked facts. This superficial retention of material for examinations does not promote understanding or long-term retention of knowledge and information.

Our course will focus mainly on equipping the students with problem-solving techniques that will be required by the student teachers to be able to teach senior school problem-solving using programming languages. Learners should learn by integrating new knowledge with existing knowledge that they have acquired elsewhere. The teachers should understand that mental models change slowly, and could be improved through active participation of the learners. Faced with a situation in which mental models will not work, they should search for meaning, and this meaning is not imposed by direct instruction. The student should by himself search for relationships among the materials and interprets knowledge in light of the previous knowledge and experiences.

Deep learning is an approach and an attitude to learning, where the learner uses higher-order cognitive skills such as the ability to analyse, synthesize, solve problems, and thinks meta-cognitively in order to construct long-term understanding. It involves the critical analysis of new ideas, linking them to already known concepts, and principles so that this understanding can be used for problem-solving in new unfamiliar contexts. Deep learning entails sustained, substantial and positive influence on the way students act, think and feel. Deep learners reflect on the personal significance of what they are learning. They are autonomous – they virtually teach themselves. But they are also collaborative learners, with high meta-cognitive and learning skills (Almeida J.).

Current theory suggests a focus not on the instructor teaching, but on student, and effective communication between teacher and student. Constructivist theorists believe that approaches to learning arise from the student’s perception of the instructor’s requirements. The role of the instructor in forming these perceptions is crucial to the student’s understanding of the content being taught. It is also crucial to understand that instructors do not directly produce deep learning in the students. It is mainly the student’s effort that is important for deep learning to take place.

The instructor needs to help the students see the purpose of the work they have to do, and to monitor their success. Active learning entails discovery; that knowledge acquisition is an on-going process, and plenty of uncertainties. Discovery happens in the brain of the learner, which is stimulated to search, store, and solve by challenging questions and opportunities to explore them in depth. Making mistakes and correcting them are integral parts of the learning process, and should not dissuade the students from learning more.

Our goal is to foster deep learning of principles and skills, and to create independent, reflective, life-long learners. We feel that achieving this would require active participation of the students. A major recommendation to emerge from the literature is that instruction should focus not only on the learning of new language features, but also on the combination and use of those features, especially the underlying issue of basic program design (Robins *et al.,* 2003). Students are not given sufficient instruction on how to combine program pieces together. Good pedagogy requires the instructor to keep initial facts, models and rules simple, and only expand and refine them as the student gains experience (Robins *et al.,* 2003, Gasparinatou & Grigoriadou, 2011).

Basic Computer Concepts before Our Method Course

By the time students enrol in our course that deals with the programming topic, they have already done an introductory course in object-oriented programming in java. The course that they have done teaches basic object-oriented programming concepts. This is their first course in programming, and statistics show that most of our students do not do well in the course. They have surface level knowledge of programming structures, and they have very little knowledge of application. Their first course in programming teaches very basic java programming that emphasises more on the language; and not on application.

Schools these days provide computer awareness (literacy) classes to their students starting in upper primary levels. By the time that they reach senior secondary schools students have gained some experience on game software, simple programming in scratch, and the use of day-to-day software used on cellular phone and computers. They have also had some experience with basic word-processing software like Microsoft Word, Microsoft Excel, Paint, etc. They have had experience as users of computer programs.

Mental Models

It is important to address the kinds of mental models which underlie programming when we teach these programming concepts. Models are crucial to building understanding of control, data structures and data representation, program design and problem domain. Active learning literature shows that students who are encouraged to actively engage and explore programming related information, by for example paraphrasing or restating a problem it in their own words, performed better at problem solving and creative transfer (Robbins *et al.,* Ludi, 2005; Thompson & Kinshuk, 2011).

The encouragement derived from creating a working program can be very powerful. In this context students can work and learn on their own and at their own pace. Working on easily accessible tasks, especially programs with graphical and animations, can be stimulating and motivating for students. Practical tasks, paired or collaborative work and peer learning has also been shown to be beneficial. There should also be gradual withdrawal of initial support from the instructor as the students gain more and more experience on the previous topics.

Loops, conditionals, arrays and recursion have all been identified as language features that are especially problematic, and could benefit from particular attention. Several authors have suggested, however, that the most important deficits relate to the underlying issues of problem solving, design, and expressing a solution/design as an actual program. The frequent practical programming exercises are almost certainly central in addressing this issue (Robbins *et al.,* 2003).

Basic Programming Concepts Learnt During Our Course

Our topic on programming concepts has got four general objectives which are:

I. Demonstrate knowledge of programming techniques.
II. Apply algorithmic tools in solving problems.
III. Show understanding of different programming languages.
IV. Show understanding of program translators.

Students are generally supposed to be able differentiate between low-level languages and high-level languages, differentiate between program translators, to describe structured programming techniques, use tools like flowcharts, pseudo code, and trace tables.

Most of the work during the semester will be on structured programming techniques. This is the difficult part for the students, and the focus of this paper, the part that needs practical application.

Our students are basically novices to programming. Despite that they have taken an introductory programming course in the Computer Science Department, they are still struggling with the basic structured programming concepts like sequences, selection structures and loops. Before mastering these concepts, there is no way to move on. These are the basics, and there is no way for effective application of the concepts before the students master these techniques. Most of everyday applications will require the use of these structured programming concepts to implement.

Like any normal class, we always have ineffective novices into effective ones. According to Robins et. al. (2003) ineffective novices "are those that do not learn, or do so after inordinate effort and personal attention", and that "the most significant differences between effective and ineffective novices relate to strategies rather than knowledge", which is available in textbooks and other sources that introduce the knowledge in a structured way. The strategies for accessing and applying this textbook knowledge for program generation, which is crucial to the learning outcome, generally receive little attention. Active learning strategies can be employed for most of the novices to have relevant mental models to programming. Active learning strategies will address that which deals with motivation, confidence, knowledge and development of the necessary mental models to programming concepts. We need to engage the students, and make them want to be effective programmers through their own participation and gradual introduction of programming concepts.

Students would have to solve programming problems on their own, answer programming questions, discuss individual solutions, and brainstorm ideas during class. We would have to start with simple programming constructs, and move on to the more difficult ones gradually. Bring the java language late when most of the students have mastered understanding the logic of the program; by the use of pseudo code and flowcharts. Our experience shows that the introduction of a specific language early will distract attention of the students. It is very easy to de-motivate the students because a simple error; like leaving out a semi-colon from a program, can actually cause a student to discontinue trying to find a solution to the problem at hand.

We usually start with sequences, then selection structures then we finally go for loops at the end. Then we can bring in examples from a variety of application areas and allow students to suggest and provide possible solutions, either individually, or in small groups. Solutions are discussed and shared among the class. This way, we make sure that there is involvement of all the students in this topic. We encourage solutions in pseudo code and flowcharts. Only when the program logic is understood do we start to use the java programming language. The java language that we use has been done by the students in a computer science course but the students still have various challenges in it. The use of the java language is expected to consolidate their knowledge of the language and also general programming concepts.

The programming topic should be taught in a way that promotes students' learning experience in a supportive teaching environment. We suggest that this topic on programming concepts should be based entirely on constructivist teaching methods and implement active learning. This topic provides the techniques required for the study of designing solutions to computer-based problems, designing of input/output, use of algorithmic tools and modular designs to solve computer-based problems; and that will help in the formulation and analysis of computer-based problems later in life.

The discussion that follows uses the Active-Learning-Based Teaching Model suggested by Hazzan *et al.,* (2011). This model is suggested for implementation in the teaching of a computer science method course.

Basic Active Learning Structure

According to Felder & Brent (2009), there are three basic steps involved in the basic active learning structure:

a. Tell the students to organise themselves into groups of 2-4 and randomly appoint a recorder in each group if writing will be required.
b. Pose a challenging question or problem and allow enough time for most groups to either finish or make reasonable progress toward finishing. The problem could actually be broken into small, several steps and treat each step as a separate activity.
c. Call on several individuals or groups to share their responses. Then discuss the responses.

The active learning literature offers many variations of the above approach. Others of the commonly used approaches include:

I. Think-pair-share: whereby a problem is posed and students work on it individually for a short time; then have them form pairs and reconcile and improve their solutions.

II. Thinking aloud pair problem solving: have the students get into pairs and designate one pair member as the *explainer* and the other pair as the *questioner*. Give the explainers to explain the problem statement line by line to their partners, and tell the questioners to ask questions when explanations are not clear. Proceed in this manner until the exercise is complete. Explainers and questioners may change roles during the whole problem solution. In programming exercises, this will significantly help in the understanding of the problem to be solved, which is a needed skill in problem analysis.

Role Played By Pupils in an Active Learning Environment

An active learning environment is a constructivist. Society today needs young people who are flexible, creative, and proactive who can solve problems, make decisions, think critically, communicate ideas

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 3, No. 2, September 2012

effectively and work efficiently within teams and groups. The 'knowing of knowledge' is no longer enough to succeed in the increasingly complex, fluid, and rapidly evolving world in which we live. Students need to have opportunities to develop personal capabilities and effective thinking skills as part of their well-rounded education. By using active-learning methodologies, students change from being passive recipients of knowledge to active and participatory learners.

Passive Learning	**Active Learning**
Being passive recipients of knowledge	Active and participatory learners
Focus on answering questions	Asking questions
Being 'spoon fed'	Taking responsibility for their own learning – reflective learners
Competing with one another	Collaborating in their learning
Wanting to have their own say	Actively listening to opinion of others
Learners of individual subjects	Connecting their learning

Rationale for the Implementation of Active Learning in Programming

It is suggested in Hazzan *et al.,* (2011) that active learning may also promote the professional development and perception of the prospective computer science teachers, as the following justifications propose.

Firstly, according to constructivism, new knowledge is constructed gradually, based on the learner's existing mental structures and on the feedback that the learner receives from the learning environments. In this process, mental structures are developed in steps, each elaborating on the preceding ones. One way to support such gradual mental constructions is by providing learners with a suitable learning environment in which they can be active (Ragonis & Hazzan, 2010). The working assumption is that the feedback, provided by learning environment in which learners learn a complex concept in an active way, may support mental constructions of the learned concepts. In our case, in order to support the construction of the computer science teachers' professional perception of programming concepts, students enrolled in our course must have a learning environment that supports this mental construction (Hazzan & Ragonis, 2011).

In order to support the construction of the professional perception in the method course, it is important that during the course, the student teachers experience acting different roles. Sometimes, the student teacher plays the role of a senior school pupil and is asked to perform senior school level assignments. At other times, they wear the hat of the computer science teacher. "As it turns out, active learning enables the switching between such situations in a very natural manner" (Hazzan & Ragonis, 2011).

The student teacher can improve the construction of their professional perception also by incorporating reflective processes into the construction process (Ragonis & Hazzan, 2010), by becoming reflective practitioners. Hazzan & Ragonis (2011) asserts that reflective practitioners are professionals who continuously improve their professional skills based on their on-going reflection with respect to their professional performance. Active leaning will encourage reflective on the part of the student teachers since it provides learners with an opportunity to reflect on the activities they perform during class as part of the class activity.

The student teachers will experience implementation of various teaching methods that can be incorporated during active learning**.** Based on the constructivist approach, the student teacher's experience of different teaching methods promotes their understanding of the methods' advantages and disadvantages.

Active learning will also increase the student teacher's awareness of the existence of a cognitive diversity in general among the students. He will be able to use appropriate pedagogical tools and therefore be able to offer substantial help for the situations they will meet in their future work with senior school students.

The student teacher will also develop high order thinking skills of analysis, critical thinking, synthesis, and evaluating tasks, skills that are relevant today in the

Active-learning-based teaching model

The Active Learning Based Teaching Model (Hazzan *et al.,* 2011) consists of four stages – trigger, activity, discussion, and summary and is illustrated in Figure 1.

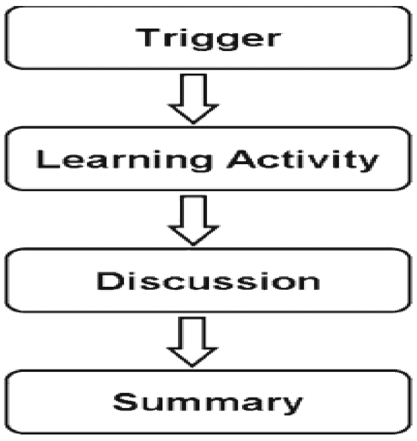

Figure 1. Active-Learning Teaching Model from Harzan *et al.,* 2011

First stage: Trigger

For this purpose, the student teachers are presented by a challenging active-learning-based trigger, an open-end activity of a kind with which they are not familiar. Specifically, a trigger should enhance and foster meaningful learning and should have the potential to raise a wide array of questions, dilemmas, attitudes, and perceptions. It is proposed that a trigger should be realistically complex and relevant for the learners. Depending on the trigger's main objective, the activity can be worked on individually, in pairs or in small groups (Hazzan *et al.,* 2011).

One of the main objectives of introducing a new topic using a trigger is to train the student teachers how to face and deal with open-ended and unfamiliar situations. Such situations, which are so predominant in computer science education require student teachers to consider multiple reaction options. In order to achieve this objective, it must be possible to approach a trigger in more than one way. Furthermore, a well-designed trigger exposes the students to a rich and varied mix of computer science and pedagogical aspects. Throughout the model stages, this vast collection of ideas is discussed, elaborated, refined and re-organized (Hazzan *et al.,* 2011).

Second stage: Activity

The students work on the trigger presented to them. The duration of this stage is determined by the complexity of the trigger used and on its educational objectives.

Third stage: Discussion

After the required period of time, during which the students work on the trigger either individually, in pairs, or in small groups, the entire class is gathered.

At this stage, products, topics and thoughts that originated during the activity stage are presented to the entire class and are discussed. The students refine their understanding of concepts, attitudes, and ideas, as part of the construction process of their professional perception.

The instructor highlights important ideas presented by the students and emphasizes principles derived from these ideas. In order to convey the notion that no unique solution exists for most teaching situations in general, and for the specific activity presented by the trigger in particular, the instructor does not judge students' positions and opinions. At the same time, however, classmates are encouraged to react and express their opinions and their constructive criticism with respect to the different ideas or materials presented.

Fourth stage: Summary.

This stage puts the topic into the context of the course and emphasizes the concepts that were discussed. It is managed differently than the three previous stages. First, it is significantly shorter. Second, while in the first three stages the students are the main actors, in the Summary stage, the course instructor takes front stage. The instructor wraps up, summarizes and highlights central concepts, teaching ideas, conceptual frameworks, and other related topics that were raised and discussed during the previous three stages. It is also important to note that the students can be asked to take the front stage here and act as a constructor with the guidance of the teacher (Hazzan *et al.,* 2011).

The summary can be expressed in different forms, such as a framework formulation, listing connections between the said topic and other topics, concept map, and so on.

The Role of the Instructor in the Active-Learning-Based Teaching Model

The instructor has to create a supportive intellectual environment that encourages students to be fully active during the whole class period.

In the first Trigger stage, the instructor presents the trigger to the student teachers.

In the second Activity, stage the instructor circulates among the different groups and listens to individual opinions, is sensitive to what they say, and encourages them to deepen their thinking. When needed, the instructor guides the students in their discussion. Though the guidance should encourage alternative thinking approaches, the instructor is advised not to dictate any position.

In the third Discussion stage, the instructor must act as a good listener and be sensitive to crucial points suggested by the students. Specifically, the instructor should encourage the students to explain why and how they developed their suggestions, suggest exploring different options, foster reflection processes, all without passing judgment on the students' opinions. The instructor highlights the important points of each opinion and presents possible connections between different ideas.

In the fourth Summary stage, the instructor sums up the ideas presented during the previous stages. This summary highlights the main points that were discussed. The instructor adds new ideas and clarifications that were not suggested by the students themselves.

CONCLUSIONS

This paper explains the implementation of an active-learning model proposed by Hazzan *et al.,* (2003) in the teaching of a topic on programming languages for a pre-service computer studies method course. The paper explains how the model is used with novice students, the role that the students should play, and the role of the instructor in order to make the students participate in the process. The more active learners are, the more meaningful is their understanding of the programming topic since learning takes place inside the learner. It is important to encourage the students to work together, talk computing language, and provide opportunities to challenge their mental models.

REFERENCES

Ben, A.M. (2001). Constructivism in computer science education. *J. of Comput. in Math. and Sci. Teach*, 20(1): 45–73

BGCSE Computer Studies Syllabus. Curriculum Development Division. Min. of Education and Skills Development, Botswana.

Gao and Hargis, (2010). Promoting Technology-assisted Active Learning in Computer Science Education. *The Journal of Effective Learning (JET),* Pg 81-93. Available

Gasparinatou, A. and Grigoriadou, M. (2011). Supporting students' learning in the domain of computer science, *Computer Science Education*, 21:1, 1-28. Available online: 30 Mar 2011. <http://dx. doi. org/ 10.1080/08993408.2010.509909> [April 12, 2012]

Hazzan, O. and Lapidot, T. (2004). Construction of a professional perception in the "Methods of Teaching Computer Science" course. inroads – SIGCSE Bull. 36(2): 57–61.

Hazzan, O. *et al.,* (2011). *Guide to Teaching Computer Science*: An Activity-Based Approach, London: Springer-Verlag Limited. http://www.springer.com/content/m0m145621u75l638/

Hermida, J. Deep Learning, <www.julianhermida.com/algoma/law1scotldeeplearning.htm> [April 23, 2012]

Kinnunen, P. and Simon, B. (2012). My program is ok – am I? Computing freshmen's experiences of doing programming assignments, *Computer Science Education*, 22(1), 1-28. <http://dx.doi.org/10.1080/08993408.2012.655091> [March 14, 2012]

Lewis, C.M. (2011). Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education*, 21(2), 105-134. <http://dx.doi.org/10.1080/08993408.2011.579805> [June 17, 2011]

Ludi, S. (2005). Active-learning activities that introduce students to software engineering fundamentals. ITiCSE'05, 128–132. Portugal: Monte de Caparica.

Ma L., *et al.,* (2011). Investigating and improving the models of programming concepts held by novice programmers, *Computer Science Education,* 21(1), 57-80. <http://dx.doi.org/10.1080/ 08993 408. 2011.554722> [April 16, 2012]

McConnell, J.J. (2005). Active and cooperative learning: Tips and tricks (Part I). *Inroads – SIGCSE Bull,* 37(2): 27–30.

Prince, M.J. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education*, 93(3). www.nscu.edu/felder-public/Papers/ Prince_ AL.pdf

Ragonis, N. and Hazzan, O. (2010). A Reflective Practitioner's Perspective on Computer Science Teacher Preparation. ISSEP2010, Zurich, Switzerland: 89–105. http://www.issep 2010. org/ proceedings_of_short_communications.pdf [April 3, 2012]

Richard, M. Fielder and Rebecca Brent (2009). Active Learning: An Introduction. <www4.ncsu.edu/ unity /lockers/f/fielder/public/PapersALpaper> [March 13, 2012]

Robins, *et al.,* (2003). Learning and Teaching Programming: A Review and Discussion, *Computer Science Education*, 13:2,137-172. <http://dx.doi.org/10.1076/csed.13.2 .137/14200> [April 16, 2012]

Silberman, M. (1996). Active Learning: 101 Strategies to Teach Any Subject. *Pearson Higher Education*

Thompson, E. and Kinshuk, (2011). The nature of an object-oriented program: How do practitioners understand the nature of what they are creating? *Computer Science Education*, 21(3), 269-287. http://dx.doi.org/10.1080/08993408.2011.607010> [April 16, 2012]