The exploration of student-centred approaches for the improvement of learning programming in higher education

Nazir Hawi

(Department of Computer Science, Notre Dame University, Louaize, Zouk Mosbeh, Lebanon)

Abstract: The author has undergone a major shift in the way of teaching his undergraduate computer programming courses. In the classroom, the teacher's computer is connected to a splitter and a video projector that display the computer's screen to the entire class. Using this technology, the programming language itself is used live in class to help the students learn how to program. The students are learning in a context by far livelier than those of previous methods. Teaching computer programming is not achieved by lecturing and writing the program instructions on board or by displaying program instructions to the class on transparencies or slides of electronic presentations. With the implementation of student-centered approaches, the students migrated from the state of passive receivers to constructors of computer programming concepts. Students are coached to develop a sense of exploration, individuality and autonomous thinking. The enthusiasm for technology has been facilitating and supporting the learner-centered approach. Everyone in this approach is a learner, including the teacher.

Key words: improving classroom teaching; interactive learning environments; programming and programming languages; post-secondary education; teaching/learning strategies

1. Introduction

The aim of this research paper is to share with the readers the experience of the exploration of student-centered approaches in the teaching and learning of an undergraduate introductory computer programming course. This course is offered by the Computer Science Department at Notre Dame University, Louaize, Lebanon. The author has been teaching this computer programming course "Computer Programming I" for years. The last time the author taught it was in the Fall 2008 semester. It was offered to a class of 30 students. The computer programming concepts are learnt with examples developed with the Visual Basic 2005 development tool.

Before the implementation of student-centered approaches, lesson goals were taught by illustrating worked-out examples on a big screen. Using these illustrations, the students engaged in discussions facilitated by the teacher. This strategy was often created in the way of class motivation. However, there was no evidence that the students used the software outside the classroom to build their own software solutions, create unusual applications, submit all their assignments or excel in exams. The efforts made by the teacher to create lively sessions were not met with enough responses in the form of work by the students, outside the classroom. Practice helps students send learnt computer programming concepts to their long-term memory (Davis, 1993). A close examination of this strategy raised a puzzling question: If communication had been improved, why were some students not doing any work to learn?

Nazir Hawi, Ed.D., assistant professor, Department of Computer Science, Notre Dame University; research field: learning computer programming.

An explanation was encountered in the literature. Students do not learn unless they construct meaning for themselves. By comparing student-centered approaches with the author's practices at times, it became evident that the author's way of teaching the course was much closer to traditional approaches than he thought, despite the integration of technology. Hence, there was a need to construct a more engaging learning environment based on student-centered approaches encompassing the usage of technology.

The move from theory to practice is not easy (Murphy, 1997a), yet the change has to start somewhere. The students, the author's partners in the teaching and learning process, were approached for suggestions on better ways to learn the course. They suggested engaging them in problem-solving activities. In every session where this idea is implemented, the students leave their comfort zone. The students, who are used to observing, listening and memorizing, have to get involved in intense work. In these problem-solving sessions, the teacher does not dispense information. The teacher observes, listens and prepares outside the classroom. The teacher and the students have to change to meet the challenges of those sessions.

This article starts by introducing the literature on learning. Teacher-centered approaches are contrasted with student-centered approaches. Also, the term "constructivism" is explained in preparation for the building of the aspired learning environment. In addition, some of the core components of constructivist classrooms are discussed and investigated. Each of them is handled in a separate section. Some areas of change are learning skills, situated learning, content coverage, teacher's role, democracy and performance feedback. The technology's role in the new approach is illustrated in the last section. The overall aim of these sections is to share with the readers the experience accompanying the efforts that are invested towards a productive learning environment that evolves from, and is consistent with, constructivist principles. Every section tries to translate aspects of constructivism into concrete instructional policies and practices guided by the literature and the teacher's experience.

2. Literature review

2.1 Traditional approaches and computer programming

Traditional teacher-centered approaches emphasize the role of the teacher as a source of knowledge and the students as the recipients (Bean, 1996), but "the days of passive learning and transmission teaching are (or at least should be) gone" (Cooper, 2004). Programming languages cannot be learned with a traditional approach (Jonassen, Peck & Wilson, 1999), because they encompass more than stacking information. Programming languages require learning basic skills in keyboarding and word processing, and demand learning new syntax, logic, control structures, procedures, modules, object-oriented programming, debugging and problem-solving. Most of programming languages require higher-order cognitive abilities, such as analysis, synthesis and evaluation that are frustrating to many novice programmers (Tiene & Ingram, 2001).

In the traditional teacher-centered approaches, the role of the leaner as a receiver conflicts with teaching. Davis (1993) believed that, "Teaching is the interaction between a student and a teacher over a subject". His definition emphasized classroom interaction. Seen from the other angle, effective learning cannot happen without any interaction between the teacher and the learners (Cooper, 2004).

2.2 How constructivism interprets learning

It appears that constructivists' ideas are building a new major paradigm (Taylor, Marienau & Fiddler, 2000) to the extent of some authors who believe that constructivism will be the prevailing theory of learning (Mayer, 2003). How does constructivism conceive learning? Candy (1991) wrote:

> Constructivism in education is concerned with two things: how learners construe (or interpret) events and ideas, and how they construct (build or assemble) structures of meaning. The constant dialectical interplay between construing and constructing is at the heart of a constructivist approach to education.

Constructivist learning aims at engaging the students actively and deeply with content in order to understand it (Mayer, 2003). It places the students at the center of the learning environment (Murphy, 1997c), where knowledge can be built through activities, such as "participatory learning, open-ended questioning and discussion and discovery learning" (Askew & Lodge, 2000). As a result, different people may construct different interpretations about a meaning (Donaldson & Knupfer, 2002). The major principles of the constructivist epistemology as identified by Winn (2003) are that: (1) Students construct their own knowledge; (2) Knowledge construction is situated in an environment; (3) The learning environment contains other people; and (4) Technology's job is to scaffold knowledge.

The fourth principle affirms that technology continues to play a role according to the constructivist conception of learning. It should support and guide cognitive processes leading to the development of an executable solution for a given programming problem (Mayer, 2003). The new instructional strategies will be based on technology (Figueira-Sampaio, et al., 2009) and the principles of constructivism. Developing and using them remain a real challenge (Tiene & Ingram, 2001).

2.3 Fostering student's participation

Constructivists encourage students to participate in determining their learning experiences (Askew & Lodge, 2000). The author asked his students to write down how in their perception the course should be taught and learned. They jotted down their ideas without hesitation. The feeling of having some control over the future of the course paved the way for a collaborative climate (Erlauer, 2003). A main goal of advocates of student-centered learning is to help students take on responsibility by making important decisions (Kaufeldt, 1999). The following sections describe those ideas and the investigation of learning activities to meet them.

3. Learning skills

3.1 The need for learning skills

The websites of major programming development tools constantly advertise for new releases. Computer programming is growing and ever-changing, and learning the latest release boosts the students' moral and raising their interest in studying (Erlauer, 2003). They believe that the possibility of an immediate or future use of the programming language gives their degree an added value. On the one hand, this is justified by the high demand for Visual Basic in Lebanon. On the other hand, people have always been fascinated with new hardware, software and technology in general (Simon, 2002).

The latest development tool release may not be the last (Askew & Lodge, 2000). Preparing students to learn current programming languages is fair, but not enough. In fact, they must be prepared to learn on their own any future releases (Simon, 2002). Many graduates reported back to the author that their managers asked them to learn an additional programming language. They were in need of transferring their knowledge to other programming languages (Askew & Lodge, 2000). A solid repertoire of learning skills could have assisted them (Simon, 2002).

Of the students who enroll in the author's courses each semester, many do not have basic learning skills. Many educators around the world feel the same (Weimer, 2002). This issue is a real challenge to teachers worldwide. The best words about this challenge are put in the form of a question by Cooper (2004):

> What do educators and educational systems need to do in order to foster the development of active, autonomous and self-directing learners, who are capable of taking responsibility for their own learning and who will be able, throughout their lives, to constantly update their skills and knowledge in response to a fast changing and complex global social and economic environment?

The author's quest for an answer led to two main ideas. The first one is to help the students learn how to learn. The second is to use some aspects of constructivism.

3.2 Problem-solving sessions

For some time, the author thought that the need for problem-solving sessions was met with worked-out examples. Some people believe that this is a powerful method for acquiring problems-solving techniques (Simon, 2002). However, it did not make the content relevant to students, and consequently, they did not exert enough efforts to learn (Erlauer, 2003). Bean's (1996) advice is to design the problems in a way that arouses the interest of passive students. Constructivists believe in grounding all learning as much as possible in tasks, activities and problems that are meaningful to the students (Tiene & Ingram, 2001).

Today, in problem-solving sessions, enough time is given to analyze every problem and to break it down into stages: input, processing and output. Breaking down a complex problem into manageable pieces is an important skill for programmers (Gertz, 2009). The students determine which facts the program should input. The students decide how the program should process the entered data. Then, they decide on the different messages to be displayed and their corresponding formats. In all the steps, the focus is on the process rather than the solution, which is a concept emphasized by constructivism (Murphy, 1997b). As the problem comes to be understood, students design the interface and set its controls properties. Technology supports problem-solving because its representational tools strengthen cognitive abilities (Bruer, 2003). During the processing phase that entails writing code, smaller tasks immerge. Students put together their efforts and the various basics they had learned to merge different ideas and come up with a solution. Then, they use the compiler to read the code and display the output. The code is amended until a desirable output is reached.

Every problem-solving session is a meaningful and joyful experience. The success of this activity is rooted in its secure environment, where the students explore and experiment without worrying about their grades (Merry, 1998). Trial-and-error is part of the process of constructing meaning (Murphy, 1997b). The best way to become a programmer is by developing programs (Marlowe & Page, 1998). In this way, knowledge is constructed in the head of every student. This process is called cognitive constructivism. Mayer (2003) labeled it as "socially mediated cognitive constructivism", because it is the result of conversing and debating (Mayer, 2003). Furthermore, he differentiates it from "social constructivism", which is concerned with how public knowledge of a social community comes into being (Mayer, 2003).

3.3 Reading

To help the students develop this important skill, the author asks his students to do in-class reading, using the course's textbook. Each reading is followed by open-ended questions for encouraging the students to interpret and analyze the content. For constructivists, this process promotes learning (Marlowe & Page, 1998) by helping students construct their own knowledge (Donaldson & Knupfer, 2002). In-class reading should help students read other computer programming textbooks, journals, magazines, the Microsoft Developer Network (MSDN) and other related resources on the Internet.

The technology proves to be valuable each time the class resorts to the classroom computer to test the ideas being advanced, using the programming language Visual Basic. It helps them discriminate between the relevant

and the irrelevant, coming to clear understanding of the content. This activity is an example of socially mediated cognitive constructivism. However, when the students follow this model individually at their convenience, the activity becomes "individually mediated cognitive constructivism" (Mayer, 2003).

With this activity, students learn to appreciate reading as an important source of learning. They start suggesting syntax corrections or alternative ways for writing instructions by referring everyone to specific sections they read in the textbook. Day after day, more textbooks appear on the desks. One time, a student bought his textbook the week before the last. He entered the class holding up the book and saying that it was an excellent resource for the development of Visual Basic applications. Not only educators, but professional programmers advise novice programmers to own one or more computer programming textbooks (Gertz, 2009).

3.4 Situated learning

Situated learning is about locating learning in a real world context (Tiene & Ingram, 2001). Formal education removes learners from that context (Fowler & Mayes, 2000), but the environment tries to simulate a real one. The solution is constructed by the whole class engaging in what Mercer and Wegerif (1999) called "exploratory talk". With this type of talk, students make suggestions for public consideration. These suggestions are critiqued constructively through offering alternatives (Mercer & Wegerif, 1999). Through interaction, students build knowledge by merging what they know with what other participants know (Fowler & Mayes, 2000; Figueira-Sampaio, et al., 2009). The teacher's role is to control disagreements and repetitions.

Would it be better if problem-solving sessions are held in a computer lab where each student or each small group of students worked on one computer? Light and Littleton (1999) reported that research shows that teachers increasingly believe that collaborative interaction around the computer in small groups is a productive way of learning. The author would recommend a setting where the whole class can work on one problem for a certain time and then separate into small groups where each sub-group works independently to find a complete solution for the problem at hand.

Problem-solving sessions simulate real world problems. Regular feedback from graduate students recruited by computer programming firms shows that working in groups is essential to building business solutions. A possible explanation is that greater cognitive resources are made available in groups compared to the case of one person trying to solve the problem alone (Light & Littleton, 1999). In fact, this is one aspect of social learning (Taylor, Marienau & Fiddler, 2000; Figueira-Sampaio, et al., 2009).

3.5 Benefits

Learning with technology fosters collaboration among students. They learn how to discuss, argue and come to a consensus (Jonassen, Peck & Wilson, 1999). Some companies are hiring people who can cooperate well with others, since one person cannot do all the work. Learning collaboration before joining the rows of professional programmers in future jobs is a great experience (Erlauer, 2003). The time that used to be spent by the teacher facing the students collectively is replaced by students' interactions with others. This strategy is believed to generate creative thinking (Bean, 1996).

3.6 Bringing change inside the classroom

This social interaction changes the attitudes of many students. For instance, Ghattas, who used to be bored, became attentive most of the time. He had a loud voice that put an end to anyone else trying to convey an idea. The author had to intervene on several occasions to give turns to others. Shaker, another student, who used to distract the class with side talk, started to ask for clarifications. The more the students were motivated, the more their voices filled the classroom.

3.7 Bringing change outside the classroom

The social dialogues among students in class generate better communication outside the class. The author noticed that some students start to link themselves up with different friends outside the class. Even after the end of the course, Maya and Elie, 2 students who did very well on the course, came to the author's office to tell him that they had agreed to work together in their senior study course during the coming semester. In the senior study course, the students work on an assigned project supervised by a faculty member.

3.8 Content decision

Introducing problem-solving sessions requires much time. Should the teacher leave out some topics? Advocates of constructivism warn against eliminating content (Marlowe & Page, 1998). Besides, cutting down the amount of material is not possible because it is needed for the next course in the sequence (Erlauer, 2003). The alternative is to find or create a set of problems in a way that students can learn the various concepts underpinning each topic while developing their own solutions. Although the implementation of this idea needs tremendous efforts, but it is rewarding since it is learner-centered.

3.9 Continuously changing content

Computer programming features are continuously replaced by improved ones. Paolo, a student, considered that learning the latest programming language release was confusing. He studied Visual Basic in Grade 10. The idea of relearning the language was irritating to Paolo. He found it difficult to relate previous knowledge with the new one (Bruer, 2003; Erlauer, 2003). Maybe this applies to some majors more than others. In the computer field, every couple of years there is a change in vision, syntax, libraries, compilers, classes, methods and controls.

3.10 Prior knowledge helps

The class discussed Paolo's argument. Some of the students considered prior knowledge of an old version of a programming language an advantage. In the book *Successful Children, Successful Learning*, Merry (1998) emphasized this view. He explains it as:

> A view based on the idea that children actively construct understandings by taking in information and relating it to what they already know: comparing, evaluating, predicting, making inferences, and so on.

They thought it as an advantage because they were able to link the new version with the old one. Charbel, a student, noted that the control structures were the same, but the syntax changed. Fuad, another student, pointed out that programming became easier, especially with the introduction of new concepts like visual programming and event handling. These observations based on comparison and evaluation demonstrated how the students assembled their own interpretations.

Paolo considered these changes irrelevant to him. Constructivists believe that a learner's previous knowledge is very important in the process of constructing new knowledge (Murphy, 1997c). It is possible that the previous information was not organized in Palo's memory to be retrieved effectively in new situations (Huba & Freed, 2000). When the schema on a topic is weak, it is difficult for a student to grasp the new version of the programming language (Tiene & Ingram, 2001). Paolo and the author met several times to solve the problem. He ended up dropping the course as part of a bigger plan to change his major to pure business. Charbel and Fuad enrolled in the second course in the sequence.

3.11 The teacher's role

What is the teacher's role in the new setting? The teacher should stop dominating the class and functioning as an exclusive content expert (Weimer, 2002). The teacher steps back, automatically, giving the chance for the

students to become the source of information. The teacher is a cognitive mentor who intervenes to direct discussions especially when the students agree on a wrong understanding of syntax, logic or concept (Mayer, 2003). The teacher passes clues to keep the learning process on fire. These concrete changes in the teacher's attitudes can only be a result of a cognitive shift which requires tremendous efforts.

One of the indications of a cognitive shift is the language used by the teacher. Now, the author is better at noticing the word "teach" forming in his brain. Immediately, the author rephrases the sentence to use "learn" instead. For instance, the sentence "On Monday, I will teach you how to create an object" is replaced by "On Monday, you will learn how to create an object". It takes a lot of work to trap and change the words "teach", "teaching" and "teacher" (Marlowe & Page, 1998).

It takes a great deal of work to establish a constructivist learning environment. Not all teachers who are used to standing and delivering most of the content material are ready for the challenge. Helping the students learn with technology necessitates that the teachers should learn to use the technology themselves. This is problematic for some teachers (Jonassen, Peck & Wilson, 1999).

When the author joined Notre Dame University, 2 of the author's colleagues wanted to know why it was so important for him to teach with a video projector. After listening carefully to the author's point of view, they told him that in no way they could use the technology to teach programming. What worried them most was the content coverage. Traditional education methods allow teachers to cover as much material as time allows them (Marlowe & Page, 1998). One of the teachers said that he could not imagine himself teaching without using the chalk. He admitted that he liked the noise emitted by every stroke on the blackboard. "It keeps the students awake", he added. The other teacher believed that the students were responsible for understanding and applying the ideas that he transmitted in class. Some of the author's advisees who took a course with him confessed to the author that he was boring to the extent of their dropping the course. They preferred to wait for another teacher to offer the course. Constructivists see his approach as a repressor of creativity, autonomy, independent thinking, competence, confidence and self-esteem (Marlowe & Page, 1998). Thus, not only students are resistant to change, but teachers are, too.

Some students cannot easily grasp the fact that the teacher is not anymore the permanent focus point. There are moments when they think that the teacher is weak and the class is beyond control. They feel panic and ask for immediate intervention. They are used to being dominated by teachers and cannot see their classmates in charge.

Sometimes, intruders open the door and look into the class. To their surprise, they find an unmonitored class functioning. It takes them some time to find the teacher (Marlowe & Page, 1998). The astonishment on their faces is rewarding.

3.12 Democracy

The transition to democracy started by listening to the students. Lea, Stephenson and Troy (2003) stated that, "If education is to be truly student-centered, students should be consulted about the process of learning and teaching". For instance, Rania and Mike rose to the occasion and expressed their need for problem-solving sessions. Their input changed the course structure since the teacher introduced what they asked for. Rania started asking questions and voiced her views. Mike stopped daydreaming and participated in building solutions.

In student-centered approaches, the teacher shares power with students in amounts that can be handled (Weimer, 2002). For instance, the students cannot decide to get rid of the exams or the assignments. The teacher delegates power to the students as long as the learning process is leading to productive academic ends (Marlowe & Page, 1998). For example, the students cannot invite a speaker on issues irrelevant to the course content.

3.13 More democracy

When the author decided to use student-centered approaches, the author invited his class to take part in reshaping the grading policy. The stated weights in the syllabus were 20% on the first exam, 20% on the second exam, and 35% on the final exam. After discussing the merits of the different suggested combinations, the class ended with a vote. Twenty students out of 22 voted for 25% on each exam. Then, they asked for another change. Joseph suggested a weight of 50% for the exam grade on coding, and spreading the remaining 50% on short-answer, multiple-choice and True/False questions. Another student suggested a ratio of 40% to 60%. The majority voted for the second suggestion. The author promised to put these suggestions in effect. The author could see many contented faces. The change affected the course's grading system and the grading scale of its exams. These concrete changes helped the students feel that they had control over the learning process. When teachers abandon some of their authority, they assist in developing constructive learners (Jonassen, Peck & Wilson, 1999). This big incentive makes the students work toward the success of their learning choices (Kaufeldt, 1999). By choice, they become active in learning computer programming. Ever since, the author discusses the grading system with his students the first day of the semester.

3.14 Performance feedback

As the class observes the various reactions to suggestions in its attempt to solve a problem, engaged students obtain a great deal of feedback (Askew & Lodge, 2000). When students listen to different interpretations of a reading, perceptions get reinforced or rejected. The principle of multiplicity of interpretations, representations and truths is at the heart of constructivism (Murphy, 1997b). It is essential to give students continuous feedback about what they are doing (Huba & Freed, 2000). The new learning environment offers continuous and immediate feedback (Kaufeldt, 1999). The feedback is a fruit of social interaction between any student in the class, his/her classmates and the class teacher.

George brought remarkable additional work to the office. He was rewarded by being allowed to exhibit his project to the whole class on the big screen. To honour him, his classmates clapped him. This immediate highly rewarding peer feedback triggered the emergence of good projects by him and by others. This event forged a new routine. At the start of every session, there were students eager to display their latest projects, which received deep attention. As a result, the students posed a wealth of questions to the experienced student to figure out how things were done. It was an extremely motivating learning experience that amazed everyone. The students remarked that this was the best course in the university. Serge, a student, wondered if there was another class at the university where students were willing to put in much time and effort. It was a rewarding feedback for the teacher.

3.15 The role of the technology

There is no doubt that technology can assist in implementing constructivist principles in this setting (Murphy, 1997a). In fact, technology in this setting has 2 roles. First, in the classroom, it is used as a supportive communication tool. It can support learning as long as it is not used as a substitute for the teacher in delivering information (Jonassen, Peck & Wilson, 1999). In a computer programming class, the programming language is used live in the classroom, while projecting it on the big screen in a problem-solving setting. This is a constructivist activity, because the students are using the technology by engaging themselves in real computer programming. Second, inside and outside the classroom, the class is using a top notch development tool to learn computer programming concepts. This development tool has many rich features that motivate students to engage in learning the programming language. For instance, in the development tool's code editor, the interpreter program detects all syntax errors in each code line and underlines them with a wavy blue line. This feature is similar to the

spelling check of word processors. Maya, a student, found it very helpful to have a program that points out at the errors. Otherwise, she has to figure out the syntax errors by herself which is time consuming for novice programmers. In addition, the compiler determines whether the students' explorations are executable. Sami, a student, found the code editor's intellisence statement completion feature very helpful because it provides the programmer with a list of choices. With this feature, the programmer does not have to remember details or use references to remember them because they are right there ready to be selected. Tarek, a student, remarked that there was no need to bother oneself with an infinite number of trips to the teacher's office since the compiler would not run a program unless it was error free. In fact, there is a special window that displays the type of error, its location in the program, and possibly a list of suggested solutions. Further, the output window displays the results for each execution. If the output is not desirable, the students can change the code to meet the requirements. In fact, they are in control of the whole activity. This is a very powerful form of learning (Tiene & Ingram, 2001).

3.16 Pairing students

Research shows that pairing students is a powerful strategy (Askew & Lodge, 2000). Students encountering difficulties in programming are paired with more skilled students for tutoring purposes. Peer tutoring and peer collaboration promote learning (Underwood, G. & Underwood, J., 1999). The combined effect of the group gives rise to insights that might not come about if the students are working individually (Dunlap & Grabinger, 2000). This pairing places the subject matter at the heart of the communication among classmates (Davis, 1993). With pairing, students feel valued and cared for. Also, pairing nurtures emotional safety which is a dimension that constructivists strive for. It is an excellent source for peer feedback (Kaufeldt, 1999). In addition, students can use the blackboard academic suite to post questions to be answered by their classmates or the instructor. This can be just one step towards seeking help from the Visual Basic community around the world by joining user groups (Gertz, 2009).

3.17 Blackboard academic suite

Student-centered approaches can be implemented outside the classroom. In the author's computer programming course, students are invited twice to participate in forums related to the course's goals through the discussion board feature. The author encourages them to post their views as well as comment on others'. Also, they are advised to support their views with references. Most students engage in forums by adding threads (Jowallah, 2008). Actually, students ask for more on-line course discussions. The latter offer the chance for students who do not participate in in-class discussions outside the classroom.

3.18 Recommendations

Fuad, a student, suggested having exams in the computer lab that reflect the way students are learning (Erlauer, 2003). The best way for the students to demonstrate their understanding of programming is programming (Marlowe & Page, 1998). It is also important to have exams that simulate problem-solving in the real world (Dunlap & Grabinger, 2000). In a constructivist approach, teachers should assess the meaning that students have acquired (Jonassen, Peck & Wilson, 1999). So, the challenge is to develop exams that can reveal if a student is capable of applying what was learned to solve new problems (Mayer, 2003). Unfortunately, for the time being, the resources do not help in achieving this assessment practice. Assessing the students' learning is still done in the classroom and on paper.

The second recommendation is to suggest an alternative to in-class reading. Ask each student, either alone or with a partner, to choose a topic, investigate it and demonstrate it to the class. To make sure that these activities do not end up disseminating information while the students remain the receivers (Simon, 2002; Marlowe & Page,

1998), more than one team can be scheduled to work on the same topic. Outside the classroom, the teams will read, synthesis and analyze the content of a particular section. In the classroom, they are ready to discuss and debate the content to demonstrate what they have understood. Everyone else is expected to be engaged in this interactive learning experience. Constructivists agree that when students learn on their own, their understanding becomes deeper (Marlowe & Page, 1998).

4. Conclusion

Currently, the teaching and learning environment is a hybrid between the teacher-centered and student-centered approaches. Some methods are retained, such as explanation, discussion and the evaluation process. So far, the introduction of new methods has been incremental, because the restructuring of the instructional practice had to be evaluated and improved.

Altogether, steering the class in the direction of student-centered learning is productive. To begin with, democracy leads to gaining the students' support to achieve more than what they think they can. Many students end up creating wonderful projects on their own. This can be attributed to in-depth understanding of the content and to the freedom in making a choice. The students expressed freely their own perspectives on important issues. They readily told the teacher what the course was lacking, for instance, problem-solving session. Since then, democracy had been reinvigorating the course.

Rania and Mike were right. Problem-solving sessions helped them construct their own meaning in a discovery-learning setting. The effectiveness of this strategy culminated in students submitting additional work. By developing more applications than required by the teacher, students demonstrate a level of mastery of the content in a specific area of computer programming. The higher the level of mastery is, the more astonishing the students' projects are. Astonishing projects indicate that the students engaged themselves fully in their learning.

Employing student-centered approaches has no end. It makes the course subject to revision and reframing. It brought with its problems that needed to be addressed. For instance, problem-solving sessions and in-class reading consume a lot of class time. Also, some students remain not interested. A study conducted by Hockings (2009) showed that, "Student-centered learning was ineffective for around 30% of undergraduates in a large and diverse group studying business operations management".

The change in the way of teaching in the classroom has propagated and has taken several forms. There have been changes in the course design, the classroom management, the students' ways of learning, the computer services center, the computer selling and buying community, and the learners' homes. The following relationships have been affected: teacher-student, student-student, teacher-computer services, chairperson-computer services, student-parent and many others.

Since the implementation of student-centered approaches, it has been a learning period for everyone, the teacher and the students. At the end of each semester, the author looks forward to the next semester in order to continue the development of the new instructional approaches and to implement them not only in the entry level course, but in all his other computer programming courses.

References:

Askew, S. & Lodge, C. (2000). Gifts, ping-pong and loops-linking feedback and learning. In: Askew, S. (Ed.). *Feedback for learning*. London: Routledge/Falmer, 1-18.

Bean, J. (1996). *Engaging ideas: The professor's guide to integrating writing, critical thinking, and active learning in the classroom.* San Francisco: Jossey-Bass.

Bruer, J. T. (2003). Learning and technology: A view from cognitive science. In: O'Neil, H. F. & Perez, R. S. (Eds.). *Technology applications in education*. Mahwah, New Jersey: Lawrence Erlbaum Associates, 159-172.

Candy, P. C. (1991). Self-direction for lifelong learning: A comprehensive guide to theory and practice. In: Taylor, K., Marienau, C. & Fiddler, M. (Eds.). *Developing adult learners: Strategies for teachers and trainers.* San Fransisco: Jossey-Bass, 19.

Conole, G., Jacobs, G. & Squires, D. (2000). *The changing face of learning technology.* Cardiff: University of Wales Press.

Cooper, P. (Ed.). (2004). Learning and teaching: Challenges for the future. *The Newsletter of the Leicester Doctorate of Education, January*, (11), 1-12.

Davis, J. R. (1993). *Better teaching, more learning: Strategies for success in postsecondary settings.* Arizona: The Oryx Press.

Donaldson, J. A. & Knupfer, N. N. (2002). Education, learning, and technology. In: Rogers, P. L. (Ed.). *Designing instruction for technology-enhanced learning*. London: Idea Group Publishing, 19-54.

Dunlap, J. C. & Grabinger, R. S. (2000). Rich environment for active learning: A definition. In: Conole, G., Jacobs, G. & Squires, D. (Eds.). *The changing face of learning technology*. Cardiff: University of Wales Press, 8-38.

Erlauer, L. (2003). *The brain-compatible classroom: Using what we know about learning to improve teaching*. Alexandra, Virginia: Association for Supervision and Curriculum Development.

Figueira-Sampaio, A. S., et al. (2009). A constructivist computational tool to assist in learning primary school mathematical equations. *Computers & Education, 53*(2), 484-492.

Fowler, C. J. H. & Mayes, J. T. (2000). Learning relationships from theory to design. In: Conole, G., Jacobs, G. & Squires, D. (Eds.). *The changing face of learning technology*. Cardiff: University of Wales Press, 39-50.

Gertz, M. (2009). *How do we learn to be good programmers?* Retrieved October 5, 2009, from http://blogs.msdncom/vbteam/archive/2009/05/05/how-do-we-learn-to-be-good-programmers-matt-gertz.aspx.

Goodman, P. S. (2002). *Technology enhanced learning: Opportunities for change*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Hockings, C. (2009). Reaching the students that student-centered learning cannot reach. *British Educational Research Journal, 35*(1), 83-98.

Huba, M. E. & Freed, J. E. (2000). *Learner-centered assessment on college campuses: Shifting the focus from teaching to learning.* London: Allyn and Bacon.

Jonassen, D. H., Peck, K. L. & Wilson, B. G. (1999). *Learning with technology.* Upper Saddle River, New Jersey: Merrill.

Jowallah, R. (2008). Using technology supported learning to develop active learning in higher education: A case study. *US-China Education Review, 5*(12), 42-46.

Kaufeldt, M. (1999). *Begin with the brain: Orchestrating the learner-centered classroom*. Tucson AZ: Zephyr Press.

Lea, S. J., Stephenson, D. & Troy, J. (2003). Higher education students' attitudes to student-centered learning: Beyond "educational bulimia"? *Studies in Higher Education, 28*(3), 321-334.

Light, P. & Littleton, K. (1999). *Learning with computers: Analysing productive interaction*. London: Routledge.

Marlowe, B. & Page, M. (1998). *Creating and sustaining the constructivist classroom.* Thousand Oaks, California: Corwin Press.

Mayer, R. E. (2003). Theories of learning and their application to technology. In: O'Neil, H. F. & Perez, R. S. (Eds.). *Technology applications in education.* Mahwah, New Jersey: Lawrence Erlbaum Associates, 127-57.

Mercer, N. & Wegerif, R. (1999). Is "exploratory talk" productive talk? In: Light, P. & Littleton, K. (Eds.). *Learning with computers: Analysing productive interaction.* London: Routledge, 79-101.

Merry, R. (1998). *Successful children successful learning.* Buckingham: Open University Press.

Murphy, E. (1997a). *Constructivism: From philosophy to practice*. Retrieved October 16, 2008, from http://www.stemnet.nf.ca/~elmurphy/ emurphy/cle.html.

Murphy, E. (1997b). *Constructivist learning theory*. Retrieved October 16, 2008, from http://www.stemnet.nf.ca/~elmurphy /emurphy/cle2b.html.

Murphy, E. (1997c). *Characteristics of constructivist learning and teaching*. Retrieved October 16, 2008, from http://www.stemnet.nf.ca/~elmurphy/emurphy/cle3.html.

O'Neil, H. F. & Perez, R. S. (2003). *Technology applications in education: A learning view.* Mahwah, New Jersey: Lawrence Erlbaum Associates.

Rogers, P. L. (2002). *Designing instruction for technology-enhanced learning.* London: Idea Group Publishing.

Simon, H. A. (2002). Cooperation between educational technology and learning theory to advance higher education. In: Goodman, P. S. (Ed.). *Technology enhanced learning: Opportunities for change*. Mahwah, New Jersey: Lawrence Erlbaum Associates, 61-74.

Taylor, K., Marienau, C. & Fiddler, M. (2000). *Developing adult learners: Strategies for teachers and trainers.* San Francisco: Jossey-Bass.

Tiene, D. & Ingram, A. (2001). *Exploring current issues in educational technology*. Boston: McGraw-Hill.

Underwood, G. & Underwood, J. (1999). Task effects on co-operative and collaborative learning with computer. In: Light, P. & Littleton, K. (Eds.). *Learning with computers: Analysing productive interaction*. London: Routledge, 10-23.

Weimer, M. (2002). *Learner-centered teaching: Five key changes to practice (1st ed.)*. San Francisco: Jossey-Bass.

Winn, W. (2003). Beyond constructivism: A return to science-based research and practice in educational technology. *Educational Technology, 43*(6), 5-13.

(Edited by Nicole and Lily)