Issues in Information Systems
Volume 15, Issue I, pp. 399-408, 2014

PROGRAMING NOT REQUIRED? DID THE IS-MODEL CURRICULUM GET IT RIGHT?

Jeanne M Baugh, Robert Morris University, Pittsburgh, PA, *baugh@rmu.edu*
Paul J. Kovacs, Robert Morris University, Pittsburgh, PA, *kovacs@rmu.edu*

ABSTRACT

The requirement for a programming language within the Information Systems major may be going by the way side, according to the IS2010 Model curriculum. But taking a programming language does more than just teach the student to write a syntactically correct program. It teaches the student how the computer makes decisions, stores data and basically how the machine that is so pervasive in our daily lives operates. This paper will show that learning a programming language increases the student's knowledge in many areas related to computers.
Students enrolled in beginning programming courses (C++, Java, Visual Basic and COBOL) were surveyed to ascertain their perceptions of the course. The result of the survey shows that more is attained from learning to program other than just writing correct code. Students learn how the computers work and what a program is all about, thus attaining an appreciation for the computing power at their fingertips. Students learn to think logically in order to solve a problem with computer code. Because computers are infused prominently in all of our daily lives, not only Computer Information Systems majors, but all students should take a programming course.

Keywords Computer Programming Course, Novice Programmers, Computer and Information Systems Curricula

INTRODUCTION

The Information Systems model curricula [7, 8, 10] stated that schools should produce graduates competent and confident in developing and deploying Information Systems. But what skill set is really necessary? The skill set required for employment in the "real world" is constantly evolving [9]. Coding efforts are often large projects worked on by many individuals and comprising many, many lines of code. Employers may want their employees to program as well as communicate. Programming skills along with communication skills go hand in hand for successful employment. In fact, employers are increasing demanding this of their entry level employees [12]. Many IS programs require courses where various communications skills are required, such as Systems Analysis and Project Management. Hoverer, what about other skills, such as computer programming?

Traditionally the Information Systems program has had the requirement of programming. The bulk of IS business schools believe in the necessity for programming and database skills [3]. Approximately 99% of these schools offer at least one programming course and all offer database. Likewise, the 47 accredited ABET [2] programs offer multiple programming courses as well as database. In the Information Systems Model Curricula, [13] programming was omitted from the list of requirements for an IS degree. Many business schools and ABET accredited programs do not agree with this decision. Additionally, many of us IS educators find it difficult to understand the thought process of IS 2010 curriculum designers in not requiring programming.

Perhaps the thinking of the IS 2010 curriculum designers is that requiring programming will deter students from declaring an Information Systems major? Perhaps they feel that programming is no longer needed for the IS major? The text from the model curriculum states:

> "One of the more noticeable changes to the IS model curriculum is the removal of application development (IS 2002.5 Programming, Data, File, and Object Structures) from the prescribed core. It is important to understand that although application development is not included in the core, it has not been removed from the IS program, and the task force acknowledges that a strong case can be made for inclusion of programming, computational thinking, data structures, and related material in an IS program. Application development can still be offered in most IS programs. By offering application development as an elective the IS 2010 model curriculum increases its reach into nonbusiness IS programs while also creating flexibility for curricula that choose to include an application development course. The programs that want

to go even further and include a sequence of programming courses can choose from approaches introduced either in the Computer Science or in the Information Technology curriculum volumes"

Apigian, C.H. and Gambill [3] studied catalog copy from 240 schools of business and found that 99.17% taught at least one course in programming. A study of ABET accredited programs showed that these programs required at least several courses in Programming. At the authors' institution, IS students are required to take two semesters of a programming language [1] also found that there is an expectation of industry that students must know programming. Few would argue that information systems students should study both systems analysis and data management. Both areas are not only tied to programming, but both are dependent on programming.

Programming is a necessary skill. According the bureau of labor statics website (http://www.bls.gov), with a computing degree, a student could expect to earn about $70,000 to start. Computer Science is the highest paying college degree and computer jobs are growing at two times the national average.(http://code.org) However, as can be seen in figure 1, students are not going into the field in big enough numbers to match the projected need.

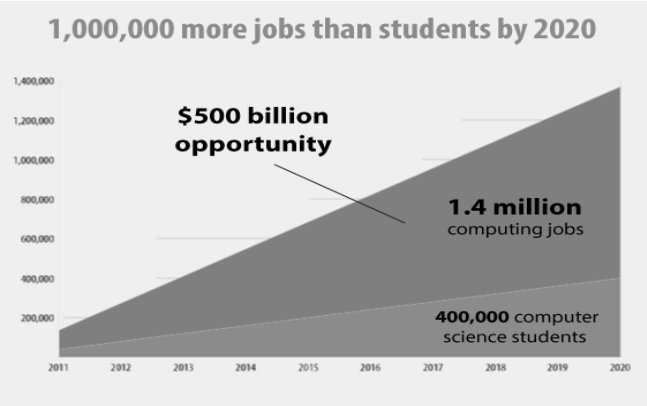

Figure 1. Job projections according to code.org

There are many advocates for requiring programming not only for the Information Systems major, but for all students: (http://www.code.org/quotes)

- Susan Wojcicki Senior Vice President, Google - "Learning to code makes kids feel empowered, creative, and confident".
- Bill Gates, Chairman, Microsoft -"Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that I think is helpful in all domains".
- Dr. Mehmet Oz, Cardiothoracic Surgeon, Author, and TV Personality-"To best prepare for life in the 21st century, today's students should ideally learn basic computer programming".
- President Bill Clinton-"At a time when people are saying "I want a good job - I got out of college and I couldnt find one," every single year in America there is a standing demand for 120,000 people who are training in computer science".
- Mike Bloomberg Mayor, New York City-"We salute the coders, designers, and programmers already hard at work at their desks, and we encourage every student who can't decide whether to take that computer science class to give it a try. New York City's economic future depends on it, and while we're already giving thousands of our students the opportunity to learn how to code, much more can and should be done".
- Tony Hsieh, CEO, Zappos- "I think everyone should get a little exposure to computer science because it really forces you to think in a slightly different way, and it's a skill that you can apply in life in general, whether you end up in computer science or not".

PROGRAMMING SURVEY

The authors of this paper as well as many others feel that the IS- 2010 model curriculum designers' thought process might be flawed. But what do the students think? What do students feel they get out of a programming course? Students who were enrolled in a programming course at the authors' institution were surveyed to access their views on the course. The survey instrument consisted of 24 open and closed-ended questions and was administered during the spring semester of the 2013/2014 academic year. These students were either taking beginning C++, Java, Visual Basic, or COBOL. A small number of the students were online, but most of them were in a traditional on ground course. The breakdown of the student population is summarized in Table 1. Table 2 defines the students' majors, and Table 3 defines how difficult they thought the course was.

Table 1. Population

Course	**Responses**
C++	62.22%
Java	13.33%
Visual Basic	14.81%
COBOL	9.63%

Table 2. Majors

Major	**Responses**
CIS	41.48%
Engineering	48.89%
Business	1.48%
Other	8.15%

Table 3. Difficulty

Difficulty	**Responses**
Very Easy	4.44%
Somewhat Easy	9.63%
Easy	25.19%
Somewhat Difficult	46.67%
Difficult	14.07%

Of the students surveyed, 61% of the students reported that the course was their first time programming and 72% stated that it was a requirement for their major. As can been seen in Table 2, only 41.48% of the students were CIS majors. Other areas made up the remaining survey participants. Therefore, this is a good sample of students where the majority of which will probably not become programmers. In fact, 47% of all students surveyed did report that they do not expect to ever program in the future. Table 3 summarizes the course difficulty level from the view of the students. The IS-model curriculum suggests that students may be shying away from an IS degree because they perceive the programming requirement to be a difficult one. But 39.26% of the students surveyed felt the course was easy. This survey result seems to be in conflict with many of the comments the students made to open-ended questions later on the survey. This result may need to be studied further.

While most students surveyed took programming because it was required, in the end, how did they really feel about the course experience? Fifty-six percent of the students stated that they were extremely interested in taking the class and only 6% reported they had little or no interest.

What do they feel they learned from the course? Many did report that they had some prior knowledge, but almost all reported that their knowledge did increase. Table 4 summarizes some of the knowledge areas students reported possessing before and after taking the course.

Table 4. Knowledge Areas

Question	Having some knowledge prior to taking course	Knowledge increased after taking course
Understand number and word storage in the computer?	65.00%	94.81%
Understand how a computer works?	62.96%	90%
Understand how a computer makes logical decisions?	57.04%	89.63%

Specific topics occur in all programming languages, among them are conditional (if) statements, loops, sub-programs and general logical programming concepts. Students were asked basically to report how much they think they learned in these basic areas. The Likert scale ran from 1 to 5, with 1 being "no understanding" to 5 being "great deal of understanding". The findings are summarized in Figure 2. Not many reported "1- no understanding" in any of the categories. This is a testament to those of us who work very hard in the classroom to teach these difficult topics.

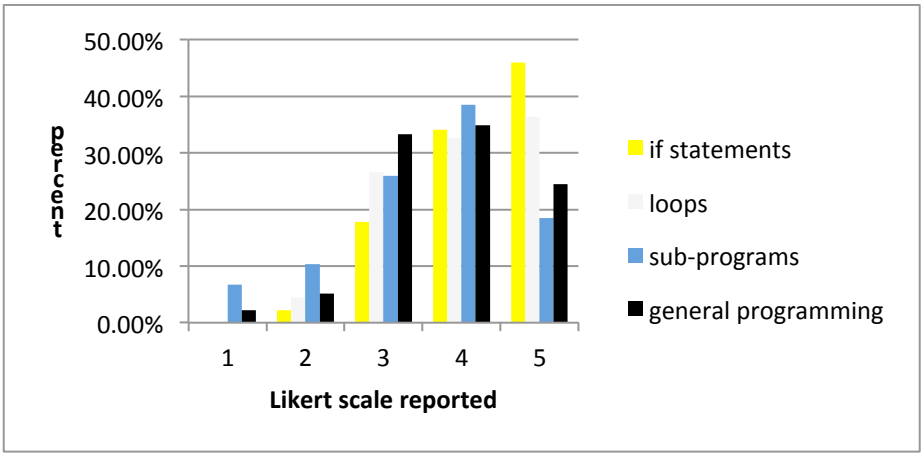

Figure 2. Knowledge of Various Concepts at Completion of Course. (Likert scale of 1 to 5)

"Programming is exciting, stimulating, fun and develops new ways of thinking" (http://www.code.org). When asked, 65% of the students reported that they felt both relieved along with a great sense of accomplishment when they got their code to run. Those teaching programming have experienced seeing students echo both of these sentiments.

Perhaps one of the most exciting results of the survey is the fact that 82.22% of the students reported that they would recommend a programming course to their friends who are not computer majors. Clearly they felt learning to program is an important skill. Figure 3 breaks down this percentage by major (CIS and Other) It makes sense that a CIS major would recommend the course, but that other majors are recommending it as well is a testament from the students, to the importance of the knowledge gained from the course. Figure 4 shows a breakdown of the reported amount of time spent on the course. This result is somewhat unexpected in that many of the comments we, as

instructors, receive form students are that they (the students) have been spending an inordinate amount of time in getting their programs to run.

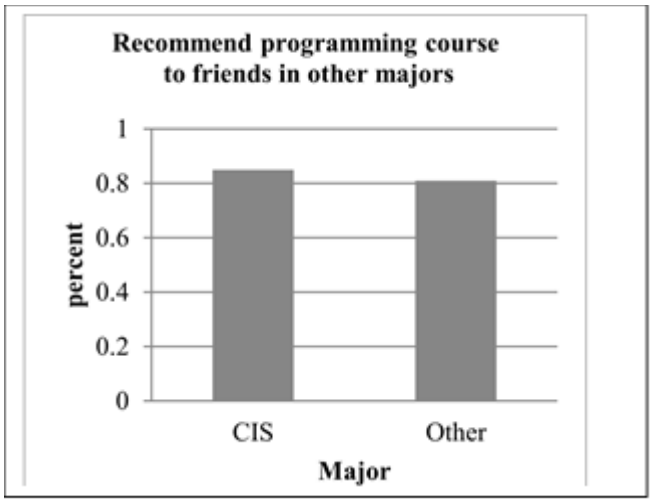

Figure3. Percentage By Major

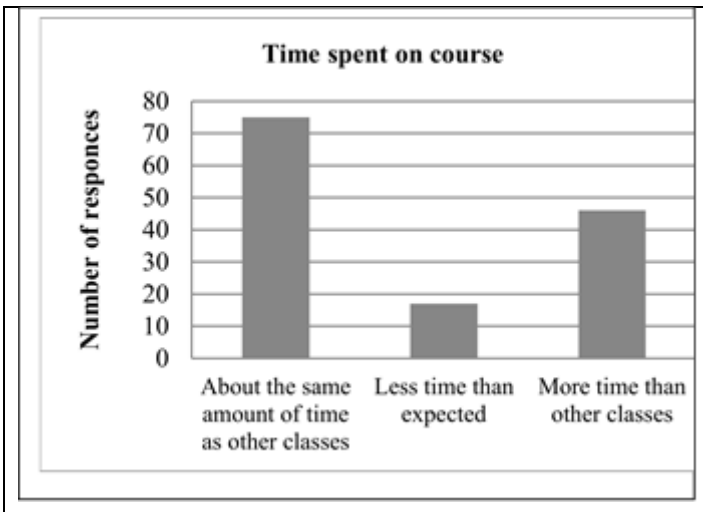

Figure 4. Percentage of Time Spent on Course

The students were asked several open ended questions. The first was: **What surprised you the most while learning to write a computer programming?** An in depth analysis of their answers provided a true picture of their attitudes about taking a programming course. The answers were analyzed and patterns of themes arose. Some sample comments from the students for this question are as follows:

- "How much each little piece of code matter in order to make the program run. I didn't know you had to be so precise to write even the simplest code"

- "How alike different languages are"
- "How easily one tiny error like a single character being in the wrong position could stop your program from running properly"
- "Just how involved it can be, having to know all the different syntax rules to make a program work"
- "The fact a computer actually makes "logical decisions" leaves me shocked"
- "It surprised me how exact you have to be in what you do"
- "The extreme level of detail the PC required in order to be able to run it"

Table 5 contains a summary of many of the themes found with the analysis of the student comments for this question.

Table 5. Themed student responses to what surprised students the most about programming course

THEME	#RESPONSES
How precise the code must be (detailed)?	20
Logical decisions computer makes	15
How complex programming is (hard…difficult)	14
How computers work?	13
More than one way to solve a problem	12
Various languages have the same constructs	11
Time consuming work	9
Did not think I could write a program	9
How easy programming was	8
Liked writing code	6
The way computers do math	6

Of course, students who have never programmed before are surprised that it takes so much detail to create a running program. That many felt that it was difficult is also not surprising. But the course also achieved a goal of exposing the students to a new way of thinking…the way a computer "thinks". The fact that the computer actually makes logical decisions was a very new concept for many. Along the same line of thinking, they also reported that how a computer works as well as how it does math is something new and exciting to them.

Another open-ended question asked was **"What did you like least about learning to write a computer program?"** Some sample comments from the students for this question are as follows:

- "The ridiculous attention to detail required"
- "How challenging it can be to get the syntax correct"
- "Sometime you make the simplest of mistakes and it's hard to find it"
- "Having to leave a buggy program for a while because you need to eat and sleep"
- "The time it takes to write the code and get it to work. took a great deal of my day just to complete one java homework"
- "The frustration when you couldn't figure out an error in your code and sometimes the error didn't make any sense at all"

A summary of many of the themes found with the analysis of the student comments for this question is represented in Table 6.

Table 6. Themed student responses to what was *liked least* about the programming course

THEME	#RESPONSES
Nothing	20
Time consuming and frustrating	18
Course was online	10
Assignments	10
Very difficult	8
Exams	6
How detailed you have to be	6

The response level to this question was interesting. Students either did not answer this or as seen in table 6 the theme with the most responses is "nothing". Often students will find something to complain about in a course, but 20 students actually took the time to enter comments to the effect that "all was good". The next biggest theme found is that the course was time consuming and frustrating. Programming is both of these and these authors do stress this on the first day of the course. Students are also encouraged to try to find their errors when coding, but to not spin their wheels too long before asking the instructor for help. A number of students also did not like taking the course online. Learning to program is difficult and doing that in an online course can be even more so.

Finally, the open ended survey question that produced some exciting results is **"What did you like most about your programming course?".** Some sample comments from the students for this question are as follows:

- "The finished product; nothing beats the sense of accomplishment when a program finally works"
- "The fact that I was able to create and execute a working computer program of my own was the greatest feeling"
- "The feeling of actually understanding a concept, and being able to apply it"
- "Opened my understanding about computer programming because I had no prior knowledge whatsoever"
- "Looking back at having no understanding of programming to understanding everything I am doing in assignments"
- "I liked the challenge, and the satisfaction of figuring a code out"
- "Challenging myself to get a program to run. Makes you think logically"
- "The sense of accomplishment when you put so much time and effort into it"
- "when my program ran and the sense of relief"
-

Table 7 contains a summary of many of the themes found with the analysis of the student comments for this question.

Table 7. Themed student responses to what was liked most about programming course

THEME	#RESPONSES
Writing a program…sense of accomplishment, getting program to run successfully	38
Learning how a computer works	25
Help from professor	16
Coding, lab work	15

As can be seen in table 7, many of the students reported a sense of accomplishment, pride, etc. in getting their program to successfully execute. Those of us who teach programming have experienced this from our students and it is gratifying to see these results. Students also reported that they liked learning how computers work. This is a goal of this course. This is the main reason why programming should not be taken out of the Information Systems

required curriculum. Even if students never program on a computer again in the future, the course provides an opportunity to learn how the computer works. In this age of computers being everywhere, some basic understand is essential to all.

WHAT WORKS IN A BEGINNING PROGRAMMING COURSE

Will a student take a programming course if it is not required? There is no denying that writing a computer program is not easy. And, for the first time programmer, it can be a daunting task. If one sees that programming is important, and there are difficulties in teaching/learning the skill, then there should be ways to help make programming for the first time, a successful experience for the student.

Order of Topics taught is crucial
In one study it was found that the beginning programmer can create very large projects by the introduction of the programming concepts as they are needed for the project [4]. In the course, the students were first given a number of small programs to write that highlighted the basic building blocks of programming; input, output, variables, math operations, logical (if) statements, loops and control structures. Then a programming project was begun that required these concepts as well as the use of new programming concepts as each phase of the project was assigned. This method of teaching programming showed a great deal of success with 60% of the students receiving an A for the course.

Speed of the Course
Of course there are always a few students who will not be able to understand the topics in the course. This can be said of any course in a College curriculum. It is always difficult for the instructor to balance their teaching style when there are students at many different levels in the course. The marginal student may need additional actions taken to facilitate their success, such as extra help sessions and personal guidance during office hours. When especially difficult topics are introduced, such as methods/functions and class definition, extra class sessions could help.

Tutoring
Tutoring on campus is also something that is extremely beneficial. Good students should be encouraged to sign up to be tutors. We should use these advanced students as experts. Interestingly, it is a wonderful way for the student who is doing the tutoring to increase their own coding skill set. Debugging someone else's code is difficult and helps both students on each side of the tutoring process.

Cohort approach
Creating a cohort of students to go through it together is another approach that has shown success in other areas of IS education. In one study, IS Doctoral students who worked in a cohort for the three years of their study reported that the main reason they felt they were successful was because of the cohort approach [6]. The students reported that they were in it "together" and would do whatever they could to insure that all "made it". With a cohort approach, multiple programming courses with continuity from one course to the next would be easily accomplished. Students would be cheerleaders for each other.

Group Projects
Another possible method of teaching programming that may be successful is allowing students to work in groups or teams on a coding project. This approach is often used on lab assignments by these authors. Students working together and helping each other can definitely have great benefits. But, for writing code, care must be taken to ensure that everyone is helping with the code and it is not just one person who is taking on most of the work. A way to ensure this is to test the students on all concepts that are required in the specific programming assignment. One instructor has all students explain their semester project code on a final exam, to insure that they actually wrote the code [5]. The student will not be able to adequately explain the code if they did not write it or help to write it.

Institution Support
Successfully teaching programming does require more effort on the instructor's part. A great deal of student follow-up is needed for the beginners. Because faculty may spend a lot of extra time helping the first time programmer, it

is essential that the University provide support in areas such as class size, tutoring and course release for extra office hours or help sessions. If the support is not there, the success of any such course cannot be assured [11].

CONCLUSIONS

Let us return to the title of this paper: "PROGRAMING NOT REQUIRED? DID THE IS-MODEL CURRICULUM GET IT RIGHT?" The authors are sending a resounding **"NO"** as the answer to this question. Even if a student will never program again in the future, learning to program provides an increase in their knowledge/skill set that is liken to learning about other intellectual areas in a College core curriculum, such as literature, science, math and history. But, since we interact with computers on many levels in all of our daily lives, learning a little more of how they operate is not only a good idea, it is an essential idea.

We all need to adopt a new attitude: everyone can do it. The web site code.org supports a non-profit organization dedicated to promoting computer science (specifically computer coding) as a requirement for all students. (http://code.org) Their vision states that "every student in every school has the opportunity to learn how to code." Additionally, the organization supports the view that "computer science and computer programming should be part of the core curriculum in education, alongside other science, technology, engineering, and mathematics (STEM) courses, such as biology, physics, chemistry and algebra". Leaders in all phases of Industry and Education advocate for all students learning to code.

According to Wang, programming requires thinking with abstract concepts, which is difficult for novices. Secondly, programming includes many different tasks, such as problem solving, algorithm and data structure design, programming language comprehension, testing, and debugging. [14] The student who has never programmed before feels an enormous sense of accomplishment when the code all comes together and produces correct results. Students are very proud of their work. The Authors have taught programming for many years and from their experience it is clear that programming students show success in both the areas of increased technical skills and personal growth. In addition to the students' positive experiences, watching the transformation of the students from not understanding a simple output statement to writing a programming project encompassing many methods/functions is extremely rewarding to the instructor. Learning how the computer operates gives the student a view into that black box…how it is programmed, how it makes decisions and generally, *how it works*. It is no longer such a mystery to the student. This research has proved that after taking a programming course, the students have a greater appreciation for how computers aid us in our daily lives.
"Programming is exciting, stimulating, fun and develops new ways of thinking". (http://www.code.org) We, as instructors, are charged with helping to successfully prepare our students for the digital future.

REFERENCES

1. Aasheim, C., Shropshire, J., Li, L., Kadlec, C. (2012). Knowledge and Skill Requirements for Entry-Level IT Workers: A Longitudinal Study, Journal of Information Systems Education. Summer2012, Vol. 23 Issue 2, p193-204
2. Accreditation Board for Engineering and Technology (ABET) http://www.abet.org
3. Apigian, C.H. and Gambill, S.E. (2010). Are We Teaching the IS2009 Model Curriculum? Journal of Information Systems Education, Vol. 21(4), p411-420.
4. Baugh, J.M., Kovacs,P. (2012). Large Programming Projects for the beginning programmer, Issues in Information Systems, Volume 13, Issue 1, pp. 85-93, 2012
5. Baugh J. M., Davis G., Kovacs, P., Scarpino, J., Wood, D. (2009). Employers And Educators Want Information Systems Graduates To Be Able To Communicate, Issues in Information Systems, Vol X No 1. pp 198-207, 2009
6. Baugh,J. , Kohun,F. (2005) Factors that Influence the successful completion of a Doctoral Degree, IACIS Conference Presentation, Atlanta GA
7. Couger, J. D., Davis, G. B., Dologite, D. G., Feinstein, D. L., Gorgone, J. T., Jenkins, M., Kasper, G. M. Little, J. C., Longenecker, H. E. Jr., and Valachic, J. S. (1995). IS'95: Guideline for Undergraduate IS Curriculum, MIS Quarterly (19:3), 1995, pp. 341-360.

8. Davis, G., J. T. Gorgone, J. D. Couger, D. L. Feinstein, and H. E. Longenecker. (1997). IS'97: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. ACM SIGMIS Database, 28(1).
9. Gallivan, Michael; Truex, Duane; Kyasny, Lynette, 2004, "Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising", ACM SIGMIS Database, Volume35, Issue36.
10. Gorgone, J.T., Davis, G.B. Valacich, J., Topi, H., Feinstein, D.L. and Longenecker. H.E. (2003). IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. Data Base 34(1).
11. Gopalakrishnan, A. (2006). "Supporting Technology Integration in Adult Education: Critical Issues and Models", Adult Basic Education: An Interdisciplinary Journal for Adult Literacy Educational Planning, v16 n1 p39-56 Spr. 18 pp.
12. Gruba Paul, Al-Mahmood Reem, (2004). "Strategies for communication skills development", ACE '04: Proceedings of the sixth conference on Australasian computing education - Volume 30
13. Topi, H., Valacich, J., Wright, R.T., Kaiser, K.M., Nunamaker, J.F., Sipior, J.C., and Vreede, G.J. (2010). IS 2010 Curriculum Guidelines for Undergraduate Degree Programs in Information Systems, Association for Computing Machinery (ACM), Association for Information Systems (AIS)", retrieved July 14, 2012: http://www.acm.org/education/curricula/IS%202010%20ACM%20final.pdf
14. Wang, X. (. O. P. H. I. E. (2010). Teaching programming skills through learner-centered technical reviews for novice programmers. Software Quality Professional, 13(1), 22-28