IOSR Journal of Business and Management (IOSR-JBM)
e-ISSN: 2278-487X, p-ISSN: 2319-7668. Volume 19, Issue 3. Ver. I (Mar. 2017), PP 87-95
www.iosrjournals.org

Computer Programming Course for Non-MIS Business Students: Curriculum, Perception and Enrichment

Samer Y. Al-Imamy
MIS department, College of Business Administration,Prince Mohammad bin Fahd University,Saudi Arabia

Abstract: *Computer programming is introduced to different disciplines in many universities and schools around the world for several objectives, including the Information Technology competency and curriculum requirements. Prince Mohammad bin Fahd University (PMU) in Saudi Arabia set six distinctive competencies as learning outcomes to its undergraduate programs. As a fulfillment to PMU's technology requirement, programming is introduced as an elective course served by the Management Information Systems (MIS) department to the College of Business Administration (COBA) students within most of its majors. The reasons of teaching such a course to majors like Accounting, Finance and Business Administration are sometimes unclear to students. The confusion leads to lack of interest, which, in combination with the coding complexities, relinquishes the objectives of the course within the curriculum. Therefore, a new approach is needed to make this course well perceived and utilized by the attendees of different disciplines. The new approach involved introducing the programming logic through graphical tools before dealing with the complicated coding. This research proves that such an approach is easy, useful and versatile. It was also found to be positively related to students' performance and conformance with their expectations.*

Keywords*: First programming language, visual logic, logic first, teaching tools, visual environment, IT competency, curriculum, teaching programming to non-IT students.*

I. Introduction

Modern systems require business people involvement in the development process (Shaw, 2000). This involvement may span from the business and system requirement collection in one end to the involvement in the development process in the other end. The National Academy of Sciences report (Committee_on_Information_Technology_Literacy 1999) requires students to be "Fluent with IT" (FIT). Several universities employed a programming course in their curriculum as part of IT competency. The curriculum requirements and the need for the ability to utilize the technology in handling the automated processes and participating in the introduction of innovative new solutions are discussed next. The second subtitle of this section deals with the importance and need for programming to prepare the graduates from different business majors for innovative IT competitive market.

I.1 Curriculum Requirements

To acquire FITness, three types of knowledge are needed (Urban-Lurain, &Weinshank, 2001), these are:

- Ability to use computer applications.
- Understanding the computing concepts.
- Ability to use the technology to solve new problems.

The first point is the easiest to achieve. Most university students come with a thorough knowledge in different applications. However, technical applications such as Microsoft Excel and Access need to be introduced formally to enhance their skills.

Using applications (1st bullet) is not enough to use the technology in solving problems (3rd bullet). Therefore understanding computing fundamental concepts is necessary for the students to be able to add innovation into the problems they face and may even be able to introduce new solutions and applications. Despite its importance, there were debates about the point of understanding the computing concepts (2nd bullet).

Cuzdialfrom University of Michigan, argued that teaching coding to non-IT students using general purpose programming languages wouldn't be of benefit to students towards the real goal of using technology to solve new problems (Soloway 1993). On the other hand, his colleagues from different universities had different viewpoint which generally supports teaching programming to non-IT students. For its importance, their opinions are included starting with Clancy and Linn from University of California, Berkeley who suggested that instead of rejecting the programming courses, we should introduce case studies relating to students' majors. The same approach is still being used with students of diverse problem-solving and analytical capabilities who are pursuing different majors (Crabtree and Zhang 2015). These discipline related case studies create power users of applications that evident the fun of IT future. DiSessa from University of California, Berkeley, emphasized on the need to make programming easier in learning and use. They also highlighted the requirement for programs expressiveness and usefulness. Miller from Carnegie Mellon University highly supports programming, he quoted “Programming is here to stay and programming will be stretched, sanitized, tailored, and generally subtended to the needs of countless disciplines and activities. People who program are empowered”. Lastly Resnick and Papert from MIT also criticized the way of teaching programming isolated from the student’s field of interest and proposed teaching the students programming to express themselves.

The conclusion from the above opinions is that “understanding the computing concepts” (2nd bullet) is necessary for all disciplines to strengthen the students’ power and “ability to use computer applications” (1st bullet) and offer them the “ability to use the technology to solve new problems” (3rd bullet). This will meet the FITness requirement and make students achieve the information technology competency.

It is also concluded that most researches recognized the need for programming to satisfy the IT competency, but introducing programming as a list of statements to solve problems unrelated to the students’ discipline will have adverse effect and demolish the purpose of the course. Therefore communicating the importance and benefits of programming to non-IT students’ career and its future success is essential. This attempt can be translated practically as versatile programs that solve their daily needs in their disciplines.

I.2 Importance of programming to Business majors

It is obvious from the above discussion that programming is important for the students to understand applications and to use the knowledge in creating or suggesting new solutions. However, this cannot be achieved without addressing (solving) the famous question asked by non-IT (not Computer Science or MIS students) about the reasons for studying programming. Computer Science department in Florida state university introduced Computer programming courses for non-majors as a course that add valuable skills to make students more attractive to potential employers regardless of the degree program (FSU, retrieved 2014). Computer Science Department in the University of North Carolina at Chapel Hill emphasized the redefinition of all fields to take advantage of the technological enhancement (UNC, retrieved 2014).

The involvement of business people in the software development process including programming became indispensable part of the IT learning outcome of the curriculum at business schools (Roussev 2003). IT is now one of the essential component in sustaining companies’ businesses among the huge competition of the twenty first century. Looking back at the last twenty years; many businesses got out of the market due to falling behind the competitors within the business field (Chesbrough 2013). One of the major reasons of success and failure hides behind the advance in IT. For example, the main communication during the eighties was through land and air mails. With the heavy increase in computer use at the beginning of the nineties several companies outsourced their communication with customers using data posted daily by tapes or lately through the networks. Since the internet became public at the end of nineties, a majority of customers prefer accessing their bills using the internet (Pearlson& Saunders 2004), therefore, businesses need to use the internet to stay competitive. Students should be educated about this fact and the importance of IT for success. Other example is the use of transactional data in making decisions and the difference between the old managers and those who build their decisions based on the data and information. Students may not be expected to be expert in programming, but they should know and understand the potential of adding script behind Excel, the utilization of access through programming, or the possibility of creating a new application that better suit their businesses. They are not necessary to be the programmers, but as managers or responsible staff know what to discuss and request from the IT department of the company (Pijpers and van Montfort 2005).

II. Background

Programming, similar to mathematics courses, improves general problem solving abilities and logical thinking. However, some researchers are against this believe (Mayer, Dyck et al. 1986). It is reported that programming doesn’t develop the students’ problem solving skills and further study are needed to make programming a useful tool in the process of students’ mental development (Kurland, Pea et al. 1986). Others emphasized the need for critical thinking skills as an essential factor of success in programming courses Bailey and Mentz (2015). On the other hand, a new course called "Introduction of Algorithmic Thinking" was developed as a general education course (Sarawagi 2010). It is built on examples drafted from (Crews and Murphy 2008) and (Farrell 2012) books using Visual Logic as a teaching tool. “Computer Thinking CT” would be the fundamental skill of the world by the middle of the 21st Century (Wing 2006). CT can be useful for every discipline as for English language reading, writing and mathematics. Many universities have already incorporated CT in their computer programming introductory courses (Qualls and Sherrell 2010).

Most universities offer a computer programming course to non-Information Technology majors (Ali & Smith 2014). College of Business Administration (COBA) in Prince Mohammad bin Fahd University (PMU) in Saudi Arabia is no exception. It offers programming course to COBA majors served by the Management Information Systems (MIS) department. Offering programming to non-IT majors is a fulfillment to the PMU learning outcomes. PMU technological competence is defined as “prepare specialized candidates in various fields of human knowledge through utilizing modern technologies in the education process (PMU, retrieved 2016).

PMU and other universities' information technology competency is based on the international standard as stated by the National Academy of Sciences report (Committee_of_Information_Technology_Literacy 1999), that require students to be "Fluent with IT" (FIT). Programming is introduced to Saudi students in the university with no expectation of prior preparation in the subject from the school level, as with some other countries (Bailey &Mentz, 2015; Moreno-León, Robles et. al. 2016). Therefore, introducing the programming subject as a list of statements to solve problems unrelated to the students' discipline will have adverse effect (Soloway 1993; Tillberg&Cohoon 2005), therefore new types of programming tools are needed.

The discussion of the above concepts with the students, during the first week of study formed a base of satisfactory answer about their concerns of the course and its relationship to their field of study. However, further empirical testing is required. To encourage the students to use the programming as a problem solving tool within their disciplines, we should introduce the course through an easy to use, useful and versatile tool that also leads to a better performance. Such objectives can hypothetically be achieved if the coding dilemma is solved. That will give the students an opportunity to investigate solutions to their problems using a tool with the above mentioned characteristics.

II.1 Learning Enhancement

Both (Simon 1978) and (Lewis and Olson 1987) claim that learning programming will not be handled by novice minds as it does for the experts. Experience and knowledge give experts the ability to filter the broad range of options into a smaller set under which to apply particular principles. Novice programmers lack the experts' ability and treat each problem as if they are unique rather than applying known patterns. This makes learning programming a difficult task for the beginners, especially if we add the challenge of learning the language's syntax and its constraints.

One of the first attempts in using patterns (Al-Imamy, Alizadeh et al. 2006) developed an application to help students writing code syntax automatically through graphical constructs. In four empirical studies (Chesbrough 2013), syntax of most programming languages including Ruby, Java, Perl, Python and others proved to be a significant barrier for novice learners. Instructors are constantly exploring new ways to make the course clear and interesting to students. It was found that understanding the syntax and the fundamental logic simultaneously discourages and distracts the students (Chesbrough 2013). Subsequently this mix of logic and syntax is overwhelming and discourages students (Kelleher and Pausch 2005), therefore focusing on the logic only brings clarity (Olivieri 2009).

Several tools were developed to make the programming logic clear. Algorithm visualization is used as a teaching tool for introductory programming course (Avancena and Nishihara 2015). On a study by (Nour, Al-Imamy et al. 2007) an attempt to reduce the syntax complexity has produced a positive impact on several factors including students' interest and performance. Visual Logic© (www.visuallogic.org) is another useful tool created to address this challenge and help the learners to focus on (almost) syntax-free logic to reduce students' frustration and enhance learning.

Since most students are visual learners (Cardellini 2002), Visual logic as a graphical interactive tool is more suitable for the learning of introductory programming course. Based on an experiment conducted in three different colleges, (Chesbrough 2013) concluded from course evaluation that visual logic is extremely user-friendly and simple to learn. (Hughes 2012) tried to reduce the coding complexity through the use of Alice, Python, and Visual Logic in Lyndon State College. She found that Visual Logic proved to be the superior approach. Visual logic was also a subject of examination of simplicity, power and versatility by (Pijpers and van Montfort 2005).

Previous researches didn't use systematic research tools to measure the impact of Visual Logic approach on the learning process. Therefore, we try in this work to prove that focusing on logic before coding using an easy and useful tool will increase students' involvement and versatility in expressing themselves that consequently lead to a better performance. These factors are empirically tested through the hypotheses listed in the next section.

III. Research Methodology

This work consists of three research studies; the first subsection addresses observations to a class of university students where Visual Logic is introduced for about a third of the semester period. In the second subsection, a survey is analyzed to test four hypotheses. A comparison between the Exams results for students attending the class which utilized VL (the experiment group), against other conventional classes (start with Java syntax) is analyzed in the third subsection to measure the students' performance and compare it to the survey results.

III.1 Class Observation

In a typical conventional first programming course, program structure, compilation, and output statements are required for the learners to be able to write “Hello World” code. A minimum of two lectures are needed to introduce the concepts of variables, types, declarations, initialization, constants and assignments. Introducing arithmetic operations and input statement will give the learners more power in writing interactive simple “useful” program. Much more lectures in decisions, looping and nesting enrich the learners with the basic constructs needed for most programming languages.

The above sequence takes a long time for the students to understand the basic concepts, apply them and fix the errors. The time is probably be longer for non-IT majors, especially if they don’t understand the reasons behind such strict coding that they found themselves sunk in.

Contrary to the conventional scenario, we observed that as soon as Visual Logic was introduced, students started writing programs and investigating solutions to problems they have in their mind that are related in some way to their field of study or interest. The students managed to write programs prompting the user to enter a value; the value is processed using a formula (like convert USD amount to SAR, Celsius to Fahrenheit, or any other conversion they like to test). Some of them moved further to calculating the Future value of an asset. It was astounding when some of them were found investigating the “if statement” during the first lab. This is possible because the tool helps them to select a construct from a combo box. Fig. 1 shows how programs an be constructed by selecting the appropriate construct from the list.

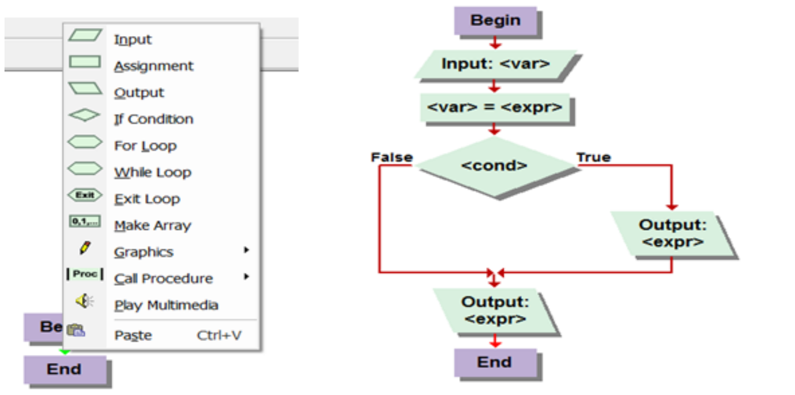

Fig. 1: Visual Logic tool showing how it guides the user to construct a runnable program.

Few students even looked at the looping structure and nesting the constructs. All these observations were noticed in the very first lab. This means Visual Logic tool is promising to create learners who think about achievable objectives instead of spending time in coding on the expense of understanding of logic. Students were noticed enthusiastic and challenging each other through questions they create that are either related to the daily processes or to their field of study.

III.2 Research Hypotheses

As we noticed in previous sections, introducing programming syntax causes a confusion that made students spending their efforts in putting words in correct sequence instead of focusing on the logic behind the code. The reason for using Visual Logic is to eliminate the overwhelming syntax building using a runnable graphical flowchart.

To get the students’ feedback about the use of Visual Logic as a programming learning environment, fifteen questions used in a survey grouped to measure four hypotheses.

III.2.1 Hypotheses List

The complexity of coding makes the students spend most of their time focusing on the syntax and fixing the syntax errors. They ignore the logic behind the code due to their heavy involvement with the code and its punctuations. This makes the process of solving the problem in hand a difficult task. Visual Logic environment is expected to ease complication of the learning process. Hence, the first null hypothesis addresses the problem of complexity is stated as:

H10: *Visual Logic is hard to use.*

The usefulness of Visual Logic as a learning tool is a key consideration of this study. This will address my and other researchers’ claims that using the graphical tools would be useful for the students in understanding

the problem logic that can be translated to code at a later stage. The second null hypothesis therefore can be stated as:
H20: *Visual Logic is not useful tool for learning process.*

Having a tool that can formulate and address problems related to students needs within their own majors and interest as a running visual program will add versatility to the subject. Instead of limiting the examples to few unrelated conventional problems, students may solve problems related to their study such as wages calculations for accounting and Time Value of Money for finance. The third null hypothesis that deals with the flexibility and adaptability to students' need is stated as:
H30: *Visual Logic is not versatile.*

The fourth and last hypothesis looks at students' impression about the impact of using Visual Logic on their performance. Starting with the logic only to understand the sequence of program execution will increase students' performance even when they move later to coding.
My claim of better performance is tested through the following null hypothesis:
H40: *Visual Logic has no positive impact on students' performance.*

The above hypothesis would also be tested through another instrument to compare the students' performance when the learning starts with Visual logic environment and move to coding after understanding the logic compared to the conventional approach when learning starts directly with coding. The mentioned instrument is introduced in the following section.

III.3 Exams Results

As well as the hypothesis test, students' performance is also measured through exams. Students' results for the experiment group (using Visual Logic up to the first exam) are compared with the results of the previous semester (based on the conventional syntax-based teaching method with a little Visual Logic introduced after the first exam). All students are male and have the same general level with no previous programming knowledge. Learners in both sections took three exams in total; the first was after the first third of the semester period. The second was after two thirds of the semester period and the final was at the end of the semester. Both courses were taught by one instructor who prepared all the exams to be similar in content and level. Though, in the experiment group both conventional first and second exams materials are combined together using Visual Logic platform only prior to EXAM1 as observed "Class Observation" subsection and explained below.

For the control groups, Java constructs are introduced starting from the first class and students are expected to program and understand Java code and logic throughout the entire semester. However, simple coding including input, output, data declaration, assignment and formulae statements are covered and tested in the first exam. Other basic constructs such as decisions and looping are tested in the second exam. All the materials covered are tested in the final comprehensive exam.

For the experiment group, all materials covered in the first and second exams of the control group are covered and tested in the first exam. This deviation from the conventional approach is due to students' enthusiasm produced from the logic understanding spawned from VL tool. However, all the focus was exerted on the logic using visual constructs with no programming syntax.

IV. Research Results

IV.1 Survey Results

Learning through Visual Logic as a first programming teaching environment was tested a sample of 98 students. The research data was collected through questionnaire, with a five-point Likert scale, involving 15 items. These items were factored into four groups related to the perception of learning through Visual Logic. The groups were mapped to the following four hypotheses discussed in "Research Hypotheses" subsection.

H10: *Visual Logic is hard to use.*
H20: *Visual Logic is not useful tool for learning process.*
H30: *Visual Logic is not versatile.*
H40: *Visual Logic has no positive impact on students' performance.*

The above factors have been measured through a number of survey questions and have been tested for reliability using Cronbach's Alpha measurement. The reliability scale was between 0.818 and 0.860 indicating that the reliability of the data is very good.

This section provides the survey analysis results. Summary of statistics are given first, results of the hypothesis tests are given next and discussion about the results is presented in the last subsection.

IV.1.1Exploratory Statistics

Table 1 shows the sample demographic data. Based on the students' number and the major program requirement, the majority (43.9%) of the surveyed students are for Business Administration where programming must be taken as an elective, followed by MIS where it is a required course. For Finance, one elective must be selected as either programming or telecommunication. 15.3% of the students selected programming. Accounting major can select between either Finance or MIS courses as electives, this explains the 2% students taking programming as an elective. About 60% of the surveyed students were male and 40% female.

Table 1: Demographic Statistics

MAJOR				**GENDER**	
BA	**MIS**	**FINA**	**ACCT**	**MALE**	**FEMALE**
43.9%	38.8%	15.3%	2%	59%	59%

Table 2 shows the statistical summary for the four variables. All of them have a minimum score above the 3 (Neutral) and average around 4 (Agree) with a little less for VERSATILITY which is of less concern to students because it deals with future use of the tool. Most scores indicate relatively favorable ratings for all the variables. Standard deviation is fairly consistent across the variables.

Table 2: Descriptive Statistics

VARIABLE	**N**	**MINIMUM**	**MAXIMUM**	**MEAN**	**STD. DEVIATION**	**RELIABILITY (C. ALPHA)**
EASEOFUSE	98	3.73	4.05	3.89	0.801	0.853
USEFULNESS	98	3.84	4.14	3.99	0.745	0.860
VERSATILITY	98	3.53	3.85	3.69	0.805	0.841
PERFORMANCE	98	3.86	4.19	4.02	0.813	0.818

IV.1.2 Hypotheses Tests

The hypotheses tests are listed below. The first one tests whether Visual Logic is an easy tool to use or not (EASEOFUSE). The second hypothesis gets the students opinion about the tool's (USEFULNESS). The third hypothesis deals with the tool's (VERSATILITY). The last one shows students experience with the tool on regards to its effect on their (PERFORMANCE). Table 3 shows the test results for the four hypotheses, which are all rejected at the p-value <.05.

Table 3: Hypotheses Test Results

HYPOTHESIS NO	**VARIABLE**	**D.F.**	**T-STATISTICS**	**MEAN DIFFERENCE**	**SIG.**
1	EASEOFUSE	97	11.001	0.89	.000
2	USEFULNESS	97	13.149	0.99	.000
3	VERSATILITY	97	8.469	0.69	.000
4	PERFORMANCE	97	12.472	1.02	.000

IV.1.3 Discussion and Implications

Visual Logic has provided students with an easy to use graphical tool that was useful in shielding the overwhelming complex coding from the programming logic. Students learn the problem solving and logic before dealing with the complexity of coding and its syntax. As with Unified Modeling Language (UML), students set up and learn the general programming logic regardless of any programming language. This will give the learners a broad knowledge about programming in general that makes the transition to languages like Visual basic, Java, C++, C# and others an exercise of translation. Therefore, learning the logic will give the students a chance to understand multi-languages. The data analysis proved that all the four factors have been significantly affected by the use of Visual Logic as we hypothesized. The t-statistics shown in table 3 indicate that a positive improvement resulted from the factors. The rejection of the first hypothesis proves that the tool is easy to use by learners in our attempt to reduce the coding complexity. The rejection of the second hypothesis proves that VL is a useful tool in solving the problems. The rejection of the third hypothesis proves that using VL will make the learning process versatile that could be related to the learners' field of study currently and in future. The approach also proves that performance of students has been improved. The performance issue will be also tested using another instrument as explained next.

IV.2 Performance through Exams

It was not easy to compare the performance gained from the use of Visual Logic tools through the comparison of results because the tool has been introduced as a pilot for the last two years. Students were surveyed during the mentioned period to test the hypotheses (as shown in "Survey Results" subsection).

Since the exam results are available, a preliminary performance through exams can be examined. Though, a thorough and comprehensive test can be conducted for bigger samples in a future work. For this test, we selected two sections taught by the same instructor who introduced the material to male students only of (on average) same level of ability and a mix of different COBA majors. These test sections were the control section (Fall 2014-2015) and experimental section (Spring 2014-2015).

In the control group, the instructor introduced the basic structure and constructs that taught the students how to write a simple Java program. Materials covered are including input, output, variables, statements and related issues. An introduction to "if statement" has also been introduced prior the first test (Exam1). Second test (Exam2) included all the mentioned topics as well as a thorough knowledge about decisions and the different types of looping. For the control group Visual logic was introduced to help the students in understanding the mentioned constructs at this period (prior to Exam2). The rest of the semester's time was spent in practicing all the constructs in moderate problems that include nesting and other related issues as a preparation to the final exam.

The experimental section was different from the control section in the early introduction of Visual Logic during the first period up to Exam1; no Java structure and coding were practiced. Students of this section where very enthusiast (see III.1) and excited in developing their own programs and execute them using the tool. All the materials covered throughout the entire semester for the control section were practiced during the first period of this experimental section. Above that several graphics applications were practiced using functions and procedures.

The difference in the introduced constructs and applications in the different periods for the control section against the experimental section makes the comparison not easy, but worth looking at the difference between the various exams to gain understanding of the tool's benefit.

As a general performance overview, Fig. 2 compares the average performance (based on the results) for the two sections:

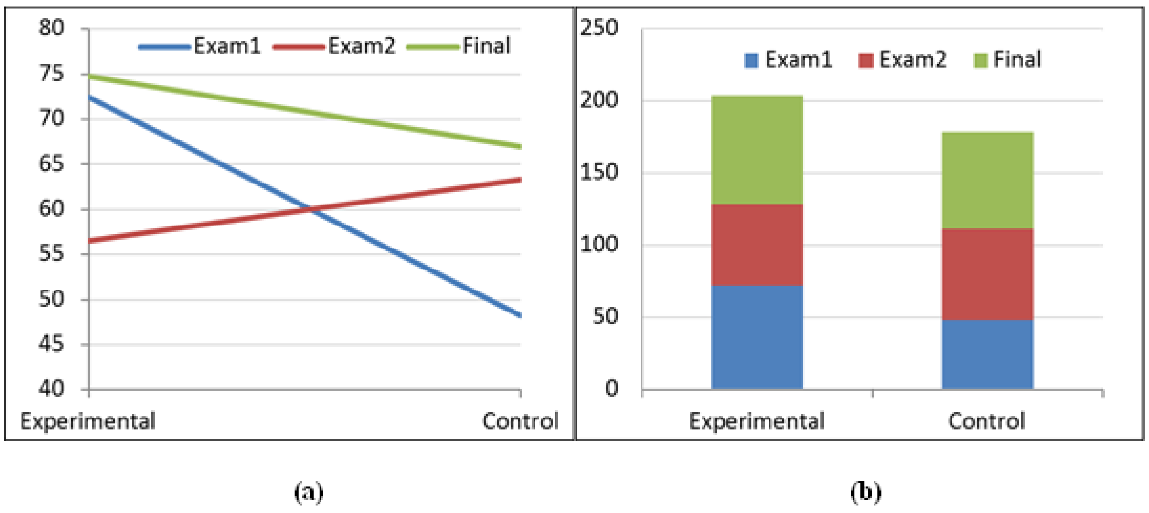

Fig. 2: Compression of results between the Experimental and the Control groups.

We can see from Fig. 2 diagrams that the Experimental group had good overall performance over the control one. For better analysis, statistical figures were collected together in table 4.

Table 4: Exams statistical Results

EXAM TYPE	**R SQUARE**	**F**	**T-STATISTICS**	**SIG.**
Exam1	.476	34.565	5.879	.000
Exam2	.022	.874	-.935	.356
Final Exam	.117	5.034	2.244	.031
Average	**.147**	**6.527**	**2.555**	**.015**

Table 4 tells us that the Experimental group scored significantly higher (t=5.879) at the $p < 0.05$ significance interval than the Control group in Exam1. 47.6% of the variance between the two groups can be explained by using Visual Logic.

In Exam2, no significant difference is noticed at the $p < 0.05$ confidence interval.

We noticed at the end of the semester that the Experimental group performed significantly better (t=2.244) at the $p < 0.05$ significance interval than the Control group. I interpret this as solid logic has been built during the first period, a problem faced with coding during the second period due to transition, but at the end of the third period, learners of the Experimental section managed to link their logical knowledge with the coding knowledge to perform well in Java programming.

For the Control group, learners struggled during the first period to understand the syntax, get control on the coding during the second period supported by a little broader logic backing from the tool. But, however, the combined knowledge demonstrated in the final Exam is still less than those of the Experimental section that built a concrete logic at the beginning of the semester.

The above comparison between sections is based on the exams (Exam1, Exam2 and final Exam) that show significantly high overall performance with (t=2.555) for the Experimental group at the $p < 0.05$ significance performance interval compared to the Control group in spite of the difference in the contents (less for the latter). Please note that each exam comprises three parts; multiple choice questions, analysis & conversion and code writing. Visual Logic has affected the overall (average) difference. Kruskall-Wallis non-parametric test also proves the significant difference and the overall performance of the Experimental over the Control group can be seen in Fig. 2 (b).

V. Conclusions

Contrary to the conventional approach, introducing logic before coding boosted the students' interest. We observed that as soon as Visual Logic was introduced, students started writing programs and investigating solutions to problems they have in their mind that are related in some way to their field of study or interest. It was astounding to me when some of them started to investigate some non-basic constructs during the first lab. This means Visual Logic tool is promising to create learners who think about achievable objectives instead of spending time in coding on the expense of understanding of logic. Students were noticed enthusiastic and challenging each other through questions they create that are either related to the daily processes or to their field of study.

Introducing Visual Logic as a teaching tool was useful in shielding the overwhelming complex coding from the programming logic. Students learn the problem solving and logic before dealing with the complexity of coding and its syntax. The data analysis proved that all the four factors have been significantly affected by the use of Visual Logic as we hypothesized. The rejection of the first hypothesis proves that the tool is easy to use by learners in our attempt to reduce the coding complexity. The rejection of the second hypothesis proves that VL is a useful tool in solving the problems. The rejection of the third hypothesis proves that using VL will make the learning process versatile that could be related to the learners' field of study currently and in future. This will enhance their ability to use the technology in solving new problems and create learners fluent with IT (FIT). The approach also proves that performance of students has been improved.

The performance issue has also been tested through Exams. It was noticed that as much as Visual Logic involved, students' performance increase.

Further research might be needed to find a better way to make the transition from pure logic to coding easier. One of the possible approaches is the use of the automatic code generation from VL flowchart as a syntax learning tool.

References

[1] Ali, A., & Smith, D. (2014). Teaching an introductory programming language in a general education course. Journal of Information Technology Education: Innovations in Practice, 13(6), 57-67.

[2] Al-Imamy, S., Alizadeh, J., & Nour, M. A. (2006). On the development of a programming teaching tool: The effect of teaching by templates on the learning process. *Journal of Information Technology Education, 5*(2006), 271-283.

[3] Avancena, A. T., & Nishihara, A. (2015). Usability and pedagogical assessment of an algorithm learning tool: a case study for an introductory programming course for high school. *Issues in Informing Science & Information Technology, 12*, 21-44.

[4] Bailey, R., &Mentz, E. (2015). IT Teachers' Experience of Teaching–Learning Strategies to Promote Critical Thinking. *Issues in Informing Science & Information Technology, 12*, 141-153.

[5] Cardellini, L. (2002). AN INTERVIEW WITH RICHARD M. FELDER ENTREVISTA CON RICHARD M. FELDER. *Journal of Science Education, 3*(2), 62-65.

[6] Chesbrough, H. (2013). *Open business models: How to thrive in the new innovation landscape*. Harvard Business Press.

[7] Committee_on_Information_Technology_Literacy (1999). Being fluent with information technology, National Academy Press.

[8] Crabtree, J., & Zhang, X. Recognizing and Managing Complexity: Teaching Advanced Programming Concepts and Techniques Using the Zebra Puzzle.

[9] Crews, T., & Murphy, C. (2008). *A Guide to Working With Visual Logic*. Cengage Learning.

[10] Farrell, J. (2012). *Just Enough Programming Logic and Design*. Cengage Learning.

[11] FSU (, retrieved 2014). Florida State University ,Computer Science Department, Computer Programming Courses for Non-majors"." Retrieved October, 2014, from http://service.cs.fsu.edu/prog.html,.

[12] Hughes, D. S. (2012, March). Introducing programming logic in a one-credit course. In *Proceedings of the 50th Annual Southeast Regional Conference* (pp. 48-52). ACM.

[13] Kelleher, C., &Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR), 37*(2), 83-137.

[14] Kurland, D. M., Pea, R. D., Clement, C., &Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, *2*(4), 429-458.

[15] Lewis, C., & Olson, G. (1987, December). Can principles of cognition lower the barriers to programming?. In *Empirical studies of programmers: second workshop* (pp. 248-263). Ablex Publishing Corp..

[16] Mayer, R. E., Dyck, J. L., &Vilberg, W. (1986). Learning to program and learning to think: what's the connection?.*Communications of the ACM*, *29*(7), 605-610.

[17] Moreno-León, J., Robles, G., &Román-González, M. (2016). Code to Learn: Where Does It Belong in the K-12 Curriculum?. *Journal of Information Technology Education: Research*, *15*, 283-303.

[18] Nour, M. Al-Imamy, S., &Alizadeh, J. (2007). An Empirical Investigation of Student Perceptions of the Effect of a Blended Learning Environment on the Learning Outcomes. *International Journal of Excellence in e-Solutions for Management 1 (2): 27-39*

[19] Olivieri, L. M. (2009). Using Visual Logic© to teach programming logic in an introductory CS course. *Journal of Computing Sciences in Colleges*, *24*(6), 146-148.

[20] Pearlson, K., & Saunders, C. S. (2004). *Managing and using information systems: A strategic approach*. New York, NY: Wiley.

[21] Pijpers, G. G., & van Montfort, K. (2006). An investigation of factors that influence senior executives to accept innovations in information technology. *International Journal of Management*, *23*(1), 11.

[22] PMU (, retrieved 2016). "Prince Muhammad Bin Fahd University. *Retrieved October, 2016, from*http://www.pmu.edu.sa/About/Uni_Mission.aspx.

[23] Qualls, J. A., &Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, *25*(5), 66-71.

[24] Roussev, B. (2003). Teaching introduction to programming as part of the IS component of the business curriculum. *Journal of Information Technology Education*, *2*, 349-356.

[25] Sarawagi, N. (2010). A general education course-Introduction to Algorithmic Thinking-using Visual Logic©: poster session. *Journal of Computing Sciences in Colleges*, *25*(6), 250-252.

[26] Simon, H. A. (1978). Problem solving and education, *Carnegie-Mellon University, Department of Psychology.*

[27] Shaw, M. (2000). Software engineering education: A roadmap. *22nd Int'l Conference on Software Engineering*, Limerick, Ireland.

[28] Soloway, E. (1993). Should we teach students to program?.*Communications of the ACM*, *36*(10), 21-25.

[29] Tillberg, H. K., &Cohoon, J. (2005). Attracting women to the CS major. Frontiers: A Journal of Women Studies, 26(1), 126-140.

[30] UNC (, retrieved 2014). University of North Carolina at Chapel Hill, Computer Science Department. Retrieved October, 2014, from http://cs.unc.edu/academics/undergraduate/courses-for-non-majors/.

[31] Urban-Lubrain, M., &Weinshank, D. J. (2001). Do non-computer science students need to program?.*Journal of Engineering Education*, *90*(4), 535.

[32] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.