Measuring and relating self-efficacy in java programming among computer science undergraduates in the South-West, Nigeria

Dr. Josiah Owolabi
National Open University, Victoria Island, Lagos

Abstract
The role that self-efficacy plays in the choice, effort as well as perseverance of a learner makes it very important to measure the self-efficacy scores in JAVA programming (SESJP) of computer science undergraduates and also to investigate its relationship with some selected variables (background in general programming language (C++), number of programming courses taken and institution types). This study made use of survey design of a correlational type. Two hundred and fifty-four (254) computer science undergraduates from four universities in the South-West, Nigeria, were selected using purposive sampling. Data was collected using a Self-Efficacy in Java programming scale (SEJPS) that contains thirty-two (32) items. The instrument was validated using Cronbach alpha approach that yielded a coefficient of 0.96. Descriptive statistics, Correlation and regression were used for data analysis. The mean and standard deviation scores for SEJPS were found to be $\bar{x}$(134.89) and SD (45.39) respectively. The mean of 134.89 out of the maximum obtainable score for the scale which is 224,translates to 60.22% and this is above average. The number of programming courses taken before JAVA programming class as well as their background in C++ were found to significantly predict self-efficacy in JAVA programming. It was therefore recommended that computer undergraduates be made to take more prerequisite courses (C++ inclusive) before exposure to JAVA programming.

Key Words: Self-Efficacy, JAVA Programming, Measuring, , Computer Undergraduates

Introduction

Learning how to write programs is vital in the training and development of computer professionals. Therefore, computer science undergraduates are expected to take and pass some programming courses during the course of their study. According to Jenkins (2001), programming is "the process of taking a problem specification written in plain language, understanding it, devising a solution, and then converting the solution into a correct computer program (usually expressed in some special-purpose programming language)". Java has been chosen for this study because it is one of the programs that is still relevant in the industries and also works on the web browser.

Programming cannot be done except with great efforts and perseverance. A programmer therefore must be persevering and committed. People's belief and judgement about their capabilities are better at predicting perseverance and commitment compared with their actual capability of accomplishing the task. This belief and judgement about one's capability to perform a task is what is called self-efficacy. Bandura (1977) submits that self-efficacy is an important psychological construct that requires attention in research because of its influence on (i) how individuals choose the activities they take part in; (ii) the extent to which they will make effort in the performance of a task and (iii) the length of perseverance in the face of difficulties encountered in completing that task. High self-efficacy has been seen to enhance motivation, make people to set themselves higher goals, put in more efforts, show more resilience and persistence more than those with low self-efficacy (Cassidy & Eachus, 2002; Schwarzer, 2004).

Computer self-efficacy (CSE) is an off-shoot of the general concepts of self-efficacy which refers to an individual's perception of his ability to use computer to perform a computing task successfully (Bandura, 1996).According to Hassan (2003), computer self-efficacy (CSE) is key in the determination of computer related ability (including programming). Since self-efficacy has been proven to be a determinant of computer ability, the need to carry out an accurate measurement of the concept of self-efficacy becomes imperative. Java Programming Self-efficacy, which is a special type of self-efficacy relating to programming using JAVA language was investigated in this study.

Sometimes, the problem of measurement is not necessarily with statistical analysis but with operationalisation. This is especially when a quality of interest cannot be measured directly. The concept of self-efficacy is an example, as there is no way it could be measured directly. In this study, it has been operationalised in such a way that respondents are required to rate their confidence on a list of JAVA programming related tasks using a scale of 1(meaning not confident at all) to 7 (meaning absolutely confident).

Self-efficacy is influenced by several other variables as evidenced in literature (Beas & Salanova, 2006; Cassidy & Eachus, 2002). These variables include: the number of programming courses, C++ background and institutional type. On number of programming courses, Jegede (2009), found no significant relationship between Java programming self-efficacy and the number of programming courses taken. He, however, established that Java programming self-efficacy is significantly predicted by the number of programming courses offered by students. In an earlier study, Ramalingan, La Belle and Wiedenbeck (2004) established that programming experience (which could also result from the number of programming courses earlier taken) influenced self-efficacy for programming.This confirmed the claim of (Bandura, 1986), that self- efficacy perceptions are developed gradually with the attainment of skills and experience.

Studies on the influence of background in C++ on JAVA programming self-efficacy seems to be rare. Historically, JAVA programming language was developed from the C++ programming language. This therefore could explain the reason why Sconberg and Dewar (2008) asserted that JAVA should not be introduced as an introductory programming course, It was expected to be taken after the students have had some experiences in prerequisite courses like the C++ for which it was an offshoot. In some universities in the South-West Nigeria, students are required to take it before taking JAVA programming as a course. These are universities who share the same opinions with Sconberg and Dewar (2008). Some other universities however do not. The implication is that background in C++ varies among the computer undergraduates who take JAVA programming as a course of study.The study therefore sought to test relationship between background in C++ and JAVA programming self-efficacy.

The type of institution was found in literature to affect the student's' academic self-efficacy. In a study carried out by Gafoor (2012) on the influence of school on academic self-efficacy, findings showed that the private secondary schools sampled have significantly higher self-efficacy when compared with their counterparts in public secondary schools. Bututcha (2013) in a related study found beginning teachers in private schools to be more self-efficacious in instructional strategies as well as overall self-efficacy when compared with their counterparts in the public schools. Capa (2005), in another study, found a similar result. This study therefore sought to study the relationship of the institution type and self-efficacy in a specific domain (JAVA programming).

Statement of the problem

Programming requires great efforts and perseverance. Great efforts and perseverance in a task like programming could be exhibited only when one has the belief in his capability to achieve success in it. While there are many computer users, computer programmers responsible for software development appear scarce. The scarcity is not because enough computer scientists are not trained computer scientists but probably not many of them take to programming as a profession after graduation. The possible explanation for this is the fact that they may not have confidence in their capability to write complicated programs. This confidence in their capability, otherwise known as self-efficacy is what this study sought to investigate. The study sought to specifically measure the JAVA programming self-efficacy possessed by the undergraduate computer science students in the South-West Nigeria and also to determine its relationship with a number of programming courses taken earlier, background in C++ and institution type.

Research Questions

The answers to the following research questions were provided by the study:

1. What are the measures of Self-efficacy in JAVA programming possessed by computer science undergraduates?
2. What type of relationship exist among background in C++, the number of programming courses taken before entering Java programming class, type of institution and Self-efficacy in Java programming?
3. How much of the variance in Self-efficacy in Java programming possessed by computer science undergraduates is accounted for by background in C++,the number of programming courses taken before entering Java programming class and the type of institution?
4. How much of the Java programming self efficacy of computer science undergraduates is associated with background in C++, number of programming courses taken before entering Java programming class and institution type?

Methodology

Purposive sampling was used for the selection of participants for the study because the researcher is interested in those public universities in the South-Western Nigeria where Java programming language is taught the undergraduate level. At the time of the study, five universities met the criteria stated above. One of the five was used for the validation of the instruments while the remaining Universities were used for the main study. All the computer science undergraduates that had been taught Java programming that were available and willing to participate in the study participated in the study. After careful scrutiny, a total of 254 questionnaires that were properly filled were used for the study.

Data was collected using Self-Efficacy in Java programming scale developed by Askar and Davenport (2009), by adapting the C++ developed by Ramalingam and Wiedenbeck (1998). The scale consisted of 32-items presented in 7- Likert format from 1 (Not confident at all) to 7 (Absolutely confident); the participants were expected to rate their level of confidence in the items highlighted using the 7-Likert format. The original reliability coefficient found by Askar and Davenport (2009) was found to be 0.99. The reliability coefficient found from the pilot in this study however was 0.96. Data was analysed using descriptive statistics (Mean and Standard Deviation), Pearson Product Moment Correlation (PPMC) Coefficient as well as Linear Multiple Regression.

RESULTS

Research Question One: What are the measures of Self-efficacy in JAVA programming possessed by computer undergraduates?

Table 1: Student's Self-Efficacy in JAVA Programming

		1	2	3	4	5	6	7	Mean	S.D
1.	I could write syntactically correct Java statements.	10 (3.5)	26 (9.0)	40 (13.8)	49 (17.0)	68 (23.5)	36 (12.5)	60 (20.8)	4.69	1.72
2.	I could understand the language structure of Java and the usage of the reserved words.	14 (4.8)	19 (6.6)	37 (12.8)	46 (15.9)	52 (18.0)	66 (22.8)	55 (19.0)	4.79	1.78
3.	I could write logically correct blocks of code using Java	18 (6.2)	26 (9.0)	40 (13.8)	35 (12.1)	67 (23.2)	51 (17.6)	52 (18.0)	4.62	1.80
4.	I could write a Java program that displays a greeting message.	15 (5.2)	15 (5.2)	11 (3.8)	26 (9.0)	33 (11.4)	55 (19.0)	134 (46.4)	5.59	1.80
5.	I could write a Java program that computes the average of three numbers.	11 (3.8)	14 (4.8)	19 (6.6)	30 (10.4)	32 (11.1)	56 (19.4)	127 (43.9)	5.54	1.76
6.	I could write a Java program that computes the average of any given number of numbers	15 (5.2)	19 (6.6)	23 (8.0)	28 (9.7)	88 (30.4)	25 (8.7)	91 (31.5)	5.06	1.79
7.	I could use built-in functions that are available in the various Java applets.	54 (18.7)	37 (12.8)	48 (16.6)	50 (17.3)	49 (17.0)	21 (7.3)	30 (10.4)	3.64	1.91
8.	I could build my own Java applets.	84 (29.1)	50 (17.3)	55 (19.0)	28 (9.7)	31 (10.7)	19 (6.6)	22 (7.6)	3.06	1.92
9.	I could write a small Java program given a small problem that is familiar to me.	25 (8.7)	21 (7.3)	22 (7.6)	85 (29.4)	34 (11.8)	36 (12.5)	66 (22.8)	4.57	1.87
10.	I could write a reasonably sized Java program that can solve a problem that is only vaguely familiar to me.	33 (11.4)	29 (10.0)	40 (13.8)	55 (19.0)	65 (22.5)	29 (10.0)	38 (13.1)	4.14	1.84
11	I could write a long and complex Java program to solve any given problem as long as the specifications are clearly defined.	46 (15.9)	29 (10.0)	37 (12.8)	66 (22.8)	45 (15.6)	36 (12.5)	30 (10.4)	3.91	1.89
12.	I can organize and design my program in a modular manner.	51 (17.6)	36 (12.5)	35 (12.1)	50 (17.3)	57 (19.7)	40 (13.8)	20 (6.9)	3.78	1.88
13.	I understand the object-oriented paradigm.	26 (9.0)	27 (9.3)	41 (14.2)	57 (19.7)	55 (19.0)	46 (15.9)	37 (12.8)	4.29	1.80
14.	I can identify the objects in the problem domain and declare, define, and use them.	19 (6.6)	35 (12.1)	30 (10.4)	68 (23.5)	43 (14.9)	43 (14.9)	51 (17.6)	4.43	1.83

15.	I can make use of a pre-written function, given a clearly labeled declaration of the function.	27 (9.3)	26 (9.0)	32 (11.1)	62 (21.5)	51 (17.6)	56 (19.4)	35 (12.1)	4.36	1.80
16.	I can make use of a class that is already defined, given a clearly labeled declaration4.39 of the class.	21 (7.3)	27 (9.3)	35 (12.1)	62 (21.5)	50 (17.3)	45 (15.6)	49 (17.0)	4.47	1.81
17.	I can debug (correct all the errors) a long and complex program that I had written and make it work.	23 (8.0)	28 (9.7)	34 (11.8)	66 (22.8)	42 (14.5)	57 (19.7)	39 (13.5)	4.39	1.80
18.	I can comprehend a long, complex multi-file program.	37 (12.8)	41 (14.2)	24 (8.3)	67 (23.2)	47 (16.3)	46 (15.9)	27 (9.3)	4.01	1.86
19.	I could complete a programming project if someone showed me how to solve the problem first.	23 (8.0)	18 (6.2)	34 (11.8)	63 (21.8)	44 (15.2)	57 (19.7)	50 (17.3)	4.58	1.80
20.	I could complete a programming project if I had only the language reference manual for help.	24 (8.3)	29 (10.0)	29 (10.0)	52 (18.0)	61 (21.1)	57 (19.7)	37 (12.8)	4.44	1.79
21.	I could complete a programming project if I could call someone for help if I got stuck.	19 (6.6)	16 (5.5)	25 (8.7)	61 (21.1)	59 (20.4)	51 (17.6)	58 (20.1)	4.76	1.74
22.	I could complete a programming project once someone else helped me get started.	19 (6.6)	24 (8.3)	35 (12.1)	64 (22.1)	42 (14.5)	63 (21.8)	42 (14.5)	4.54	1.77
23.	I could complete a programming project if I had a lot of time to complete the program.	18 (6.2)	20 (6.9)	20 (6.9)	56 (19.4)	49 (17.0)	60 (20.8)	66 (22.8)	4.88	1.79
24.	I could complete a programming project if I had just the built-in help facility for assistance.	23 (8.0)	18 (6.2)	39 (13.5)	69 (23.9)	43 (14.9)	55 (19.0)	42 (14.5)	4.47	1.76
25.	I could find ways of overcoming the problem if I got stuck at a point while working on a programming project.	28 (9.7)	30 (10.4)	38 (13.1)	70 (24.2)	50 (17.3)	42 (14.5)	31 (10.7)	4.16	1.77
26.	I could come up with a suitable strategy for a given programming project in a short time.	33 (11.4)	41 (14.2)	33 (11.4)	71 (24.6)	52 (18.0)	34 (11.8)	25 (8.7)	3.93	1.77
27.	I could manage my time efficiently if I had a pressing deadline on a programming project	30 (10.4)	30 (10.4)	29 (10.0)	91 (31.5)	48 (16.6)	43 (14.9)	18 (6.2)	4.03	1.67
28.	I could mentally trace through the execution of a long, complex, multi-file program given to me.	50 (17.3)	33 (11.4)	22 (7.6)	69 (23.9)	38 (13.1)	50 (17.3)	27 (9.3)	3.93	1.94

29.	I could rewrite lengthy confusing portions of code to be more readable and clear.	42 (14.5)	33 (11.4)	36 (12.5)	56 (19.4)	47 (16.3)	41 (14.2)	34 (11.8)	4.01	1.92
30.	I can find a way to concentrate on my program, even when there were many distractions around me.	35 (12.1)	29 (10.0)	31 (10.7)	58 (20.1)	48 (16.6)	44 (15.2)	44 (15.2)	4.26	1.92
31	I can find ways of motivating myself to program, even if the problem area was of no interest to me.	32 (11.1)	25 (8.7)	25 (8.7)	53 (18.3)	60 (20.8)	47 (16.3)	47 (16.3)	4.43	1.89
32	I could write a program that someone else could comprehend and add features to at a later date.	28 (9.7)	25 (8.7)	34 (11.8)	43 (14.9)	66 (22.8)	49 (17.0)	44 (15.2)	4.66	4.45

Key: 1-Not at all confident; 2-Mostly not confident; 3-Slightly confident; 4-50/50; 5-Fairly confident; 6-Mostly confident; 7-Absolutely confident

Table 1 displayed the measures of undergraduate computer science students' self-efficacy in various JAVA programming tasks. The table showed that the undergraduate computer students are confident in some JAVA programming tasks, not confident in some and moderately confident in some . This is evident from the percentages and means obtained in their self-efficacy in various tasks.

For the first categories, their means are above 4.00 while their percentages for fairly confident, mostly confident and absolutely confident put together were above 50.0%. These include the following tasks: (i) Writing syntactically correct JAVA statements; (ii) Understanding the language structure of JAVA and the usage of reserved words; (iii) Writing logically correct blocks of code using JAVA; (iv) Writing a JAVA program that displays a greeting message; (v) Writing a JAVA program that computes the average of three numbers tasks recorded. The five (5) tasks listed above that the students appeared to be confident in were all simple tasks. Aside simple tasks, the measures in the table above also showed that they were confident in tasks that they received assistance. These include items 19-21 which are as follows: (i) Completing a programming project if someone showed how to solve the problem first; (ii) Completing a programming project if he/she had only the language reference manual for help; (iii) Completing a programming project if he/she could call someone for help when stuck.

The result also showed that the students showed confidence in JAVA programming tasks when they have enough time for the tasks. This is evident in their responses to items 23 as follows: (i) completing a programming project if they had a lot of time to complete the project. In conclusion, the students were confident in tasks that are either simple, done when help is available or done when enough time is given for completion.

The result of the percentages and means also showed that students did not show adequate confidence in fairly difficult tasks and when they have a short time to deliver. In all of these tasks, their means were less than 4.00 and their percentages for fairly, mostly and absolutely confident put together were below 50.0%. The tasks for fairly difficult tasks include: (i) Using built-in functions that are available in various JAVA applets; (ii) Building their own JAVA applets; (iii) Writing a reasonably sized JAVA program that can solve a problem that is only vaguely familiar; (iv) Writing a long and complex JAVA program to solve any given problem as long as the specifications are clearly defined. They also showed lack of adequate confidence in tasks that require a short period of time to complete. Such tasks include items 26 and 27 as follows: (i) Coming up with a suitable strategy for a given programming project in a short time; (ii) Managing their time efficiently if they had pressing deadline on a programming project.

Research Question Two: What type of relationship exist among background in C++,the number of programming courses taken before entering Java programming class, type of institution and Self-efficacy in Java programming?

Table 2: Inter correlation Matrix of the independent variables and Java programming self-efficacy

Var	X1	X2	X3	X4
X1	1.000			
X2	0.208*	1.000		
X3	0.628*	0.292*	1.000	
X4	0.350*	0.266*	0.431*	1.000
Mean	1.60	1.63	1.24	139.55
SD	0.49	1.05	0.26	44.15

Key: X1 = Background In C++; X2 = Number of Programming Course; X3 – Institution Type; X4 = Java Programming Self Efficacy Scores * P < .05

From table 2, it can be observed that all the relationships between the three independent variables and Self-Efficacy in JAVA programming are positive and statistically significant at *p* < .05. The table shows that Self-Efficacy in JAVA programming has positive relationship with background in C++, number of programming courses and Institution type. The Institution type has the highest relationship (r = 0.431, *p* < .05) with Self-Efficacy in Java programming. This is followed by background in C++ (r = 0.350, *p* < .05) while number of programming courses has the least (r = 0.266, *p* < .05). All the relationships are also statistically significant.

Research Question Three: How much of the variance in Self-efficacy in Java programming possessed by computer undergraduates is accounted for by all of background in C++, the number of programming courses taken before entering Java programming class and the type of institution?

Table 3a:Multiple Regression of the Independent Variables on Self-efficacy in Java programming

Parameter	Value
Multiple Regression	0.177[a]
R-Square	0.031
Adjusted R -Square	0.020
Standard Error of Estimate	15.417
Sig (two-tailed)	0.047

[a] Predictors: (Constant), background in C++, the number of programming courses taken before entering Java programming class and type of institution

Table 3b: Multiple Regression ANOVA for JAVA Programming Self-Efficacy

Model	Sum of Squares	Df	Mean Square	F	Sig (2-tailed)
Regression	130327.509	3	43442.503	27.790	0.000
Residue	390804.621	250	1563.218		
Total	521132.132	253			

[a] Predictors: (Constant), background in C++, the number of programming courses taken before entering Java programming class and type of institution

[b] Dependent variable: JAVA programming Self-Efficacy score

Table 3 presents the multiple regression of the independent variables (background in C++,numberof programming courses taken before entering Java programming class and type of institution) and dependent variable (Self-Efficacy in Java programming) among computer science undergraduates. The multiple regression coefficient (R) showing the linear relationship between the three **independent variables** (background in C++, the number of programming courses taken before entering Java programming class and type of institution) and dependent variable (Self-Efficacy in Java programming) among computer undergraduates is 0.18. The adjusted R square value is 0.02; this implies that the variation in **Java programming** self-efficacy accounted for by the stated **independent variables** (background in C++, the number of programming courses taken before entering Java programming class and the type of institution) when combined, among computer undergraduates was 2.0 %.

Further verification using multiple regression ANOVA produced F-ratio = 2.686, *p* < .05. This implies that there is a significant linear relationship between the identified independent variables (background in C++, the number of programming courses taken before entering Java programming class and the type of institution) and Self-Efficacy in Java programming.

Research Question Four: How much of the variance in Self-efficacy in Java programming possessed by computer undergraduates is accounted for by each of background in C++, the number of programming courses taken before entering Java programming class and the type of institution?

Table 4: Coefficients Indicating Relative Effects of the identified Independent Variables and Java programming self-efficacy among computer undergraduates

	Unstandardized Coefficients		Standardized Coefficient			
Model	B	Std. error	Beta	t	Sig.	Remark
Constant	0.929	7.003		0.133	0.895	
INS	3.553	2.999	0.097	1.185	0.237	NS
BCPP	5.039	2.540	0.159	1.984	0.048	S
NPCS	1.153	0.580	0.129	1.988	0.048	S

[a] Dependent variable: Java programming self-efficacy
S: significant at 0.05 alpha levels; N.S: Not Significant at 0.05 alpha levels

Table 4 gives the individual contributions of the identified independent variables (background in C++ the,number of programming courses taken before entering Java programming class and type of institution**) to the dependent variable (Self-efficacy in Java programming) among computer undergraduates**. Background in C++ contributed the most to **Java programming** self-efficacy **(**B= 0.159; t = 1.984; *p* < .05). This is followed by that of number of programming courses taken before entering Java class **(**B= 0.129; t = 1.988; *p* < .05). The contribution of institution type was positive but insignificant (r = 0.095, t = 1.185; *p* > .05).

Discussion

Self-efficacy in JAVA programming possessed by computer science undergraduates was found in this study to be above average but with moderately high standard deviation. This suggests that there is room for improvement and also there is the need to work towards boosting the self-efficacy of some of the undergraduates.

Background in C++ related positively and significantly to Java programming self-efficacy. The implication is that those that had experience with CPP had higher self-efficacy in their ability to program using Java. This is in agreement with Sconberg and Dewar (2008) that argued that Java should not be introduced as an introductory programming course. It is also noteworthy that Java is an offshoot of C++, therefore it is reasonable to conclude that experience in C++ would boost their self-efficacy in Java programming.

Number of programming courses also related positively and significantly to Java programming self-efficacy. Jegede (2009) found that Self-efficacy in Java programming has no significant relationship with the number of programming courses taken. He however established that the number of programming courses offered by students significantly predicted their Java programming self-efficacy. Both studies established that number of programming courses predicts self-efficacy but disagrees on relationship between the two variables. This contrast could be a result of the differences in the participants of the two studies. For Jegede (2009), engineering undergraduates from one university participated while in the current study, computer undergraduates across a geopolitical zone were participants.
Institutional type was also found to relate positively and significantly to Java

programming self-efficacy. This is in agreement with Gafoor (2012); Bututcha (2013) and Capa (2005). A study carried out by Gafoor (2012) on the influence of school-image on academic self-efficacy, findings showed that Private secondary schools had significantly higher self-efficacy when compared with their counterparts in public secondary schools. Bututcha (2013) in a related study, found beginning teachers in private schools to be more self-efficacious in instructional strategies as well as overall self-efficacy when compared with their counterparts in the public schools. The possible reason for the above trend could be because the class size in private schools is smaller compared to the ones in public schools. Capa (2005), in another study found a similar result. This could be because of smaller class size in private than public secondary schools in Ethiopia.

Conclusion

The level of self-efficacy possessed by the computer undergraduates, though above average, should be improved upon. The predictor variables (Background in C++, number of programming courses taken before entering the Java class and institution type) related positively and significantly to Java programming self-efficacy. Number of programming courses and institution type also significantly predicted Self-efficacy in Java programming.

Recommendation:

The following recommendations were made:

(i) Background in C++ was found to relate significantly with JAVA programming self-efficacy score, therefore C++ programming language should be made a prerequisite for Java Programming course.

(ii) Number of programming courses related significantly with JAVA programming self-efficacy scores, more relevant programming courses should be included in the curriculum of semesters preceding the semester when Java Programming is to be taught.

(iii) Computer departments in federal government-owned institutions should improve on the instructional modes in Java Programming class towards enhancing the rise in computer undergraduates' self-efficacy.

References

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming among engineering students. *Turkish Online Journal of Education Technology 8* (1), 26-32.

Bandura, A. (1977). Self-efficacy: Towards a unifying theory of behaviour. *Psychological Review, 84* (2), 191-215. Retrieved from http//www.ncbi.nih.gov/pubmed/847064 .

Bandura, A. (1986). *Social foundation of thought and actions: A social cognitive theory.* Englewood cliffs, New Jersey: Prentice Hall.

Bandura, A.(1996). *Assessing self-efficacy beliefs and academic outcome: The case for specific city and correspondence.* A paper presented at the annual meeting of the American Educational Research Association, New York, NY.

Beas, M. I., & Salanova, M. (2006). Self-efficacy beliefs, computer training and psychological well being among Information and Communication Technology workers. *Computers in Human Behaviour, 22,* 1043-1058.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming Among Engineering Students. *Turkish Online Journal of Education Technology 8* (1), pp 26-32.

Bandura, A. (1977). Self-efficacy: Towards a Unifying Theory of behaviour. *Psychological Review, 84* (2), 191-215. Retrieved from the World Wide web:http//www.ncbi.nih.gov/pubmed/847064.

Bandura, A. (1986). Social foundation of thought and actions*: A social cognitive theory*. Englewood cliffs, New Jersey: Prentice Hall.

Bandura, A (1996). *Assessing self-efficacy beliefs and academic outcome: The case for specific city and correspondence.* A paper presented at the annual meeting of the American Educational Research Association, New York, NY.

Beas, M.I and Salanova, M (2006). Self-efficacy beliefs, computer training and psychological well being among Information and Communication Technology Workers. *Computers in Human Behaviour, 22,* 10431058.

Bututcha K. G. (2013). Gender and school type differences in self-efficacy in teaching. Sky Journal of Educational Research (4), 23 - 31, Available online http://www.skyjournals.org/SJERhttps://www.spsstests.com/2015/02/how-to-test-validity-questionnaire.html

Capa, Y. (2005). Factors influencing first-year teachers' sense of efficacy. Unpublished doctoral dissertation, The Ohio State University, Retrieved November 2, 2008, from http://www.ohiolink.edu/etd/send-pdf.cgi/199apa%20Yesim.pdf?acc_num=osu1110229553.

Cassidy, S. & Eachus, P. 2002. Developing the Computer User Self-Efficacy (CSUE) Scale: Investigating the relationship between Computer Self-efficacy, Gender and Experience with Computers. *Educational Computing Research, 26* (2), 133-153.

Gafoor, A. K. (2012) Influence of school-image on academic self efficacy beliefs. *Innovations and Researches in Education 2*(1), 51-58.

Hassan, B. (2003). The influence of specific computer experiences on computer self efficacy beliefs. *Computers in Human Behaviours, 19,* 443-450.

Jegede, P. O. (2009). Predictors of Java Programming self-Efficacy among engineering Students in Nigeria University. *International Journal of Computer Science and Information security* (IJCSIS), available at http:/site.google.com/site/ijcsis/ 4 (1 & 2).

Jenkins, T. (2001). *The motivation of students of programming*. A Thesis Submitted to the University of Kont at Canterbury in the Subject of Computer Science for the degree of Master of Science, (1- 5)

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-efficacy Scale and Group Analysis of Novice Programmer Self-efficacy. *Journal of Educational Computing Research, 19* (4), 367-381.

Ramalingam, V., LaBelle, D., and Wiedenbeck, S. (2004). *Self-efficacy and mental models in learning to program.* The 9th Annual SIGCSE conference on Innovation and Technology in Computer Science. Leeds, United Kingdom, *pg 171-175.*

Schwarzer, B. (2004). General Perceived Self-Efficacy in 14 cultures “user page fuberlin.de/~ health/selfscal.htm.

Sconberg, E. & Dewar, R. (2008). A principled Approach to Software Engineering Education or JAVA considered harmful. Ada User Journal, 29(3).