I.J. Modern Education and Computer Science, **2016, 9, 28-34**
Published Online September 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2016.09.04

Survey on Adverse Effect of Sophisticated Integrated Development Environments on Beginning Programmers' Skillfulness

Alaba T. Owoseni
Department of Computer Science, Interlink Polytechnic, Ijebu Jesa, Osun State, 233114, Nigeria
Email: atimothyowoseni@yahoo.com

S. A. Akanji
Department of Mathematics and Statistics, Interlink Polytechnic, Ijebu Jesa, Osun State, 233114, Nigeria
Email: deerious1@gmail.com

Abstract—Integrated development environment as a software system that aids programmers in developing software applications quickly and effectively has been perceived to also serve as an inappropriate tool for beginning programmers when it is specially developed with some complex features. This complexity in features as perceived leaves the programmers with a double role of studying complexity found in the environment and the semantics with syntaxes of the concerned programming language. This paper categorizes few of the available integrated development environments based on program building tools that are integrated in them and also considers an experimental survey on their adverse effects on novice programmers by sampling programmers' opinions using closed ended questionnaires. The population was randomly selected from some tertiary institutions in Nigeria. The opinions were statistically analyzed using chi square and based on the analysis, beginning programmers learning strengths are found greatly influenced by the type of integrated development environment used.

Index Terms—IDE, integrated development environment, compiler, intelliSense, interpreter, effect of integrated development environment on programmers.

I. Introduction

Computer programming is a difficult and much challenges-demanding task executed by students [1][2] and even by any person. The challenges involved in programming might not be a point of discussion to an experienced programmer due to some motivational factors and also the level of skillfulness but these are always points to pound on when there is a population of novice under study. There have been various views on various challenges been faced by new programmers at their earliest years of programming. Some of these challenges are, learning language syntaxes, gaining access to computer systems or networks, learning language semantic structure, learning other program constructs such as comments, control structures, data types and so on, learning sub-programs, designing a program to solve a task, debugging, lack of competent tutors, lack of technical textbooks, and the complexities involved in integrated development environments. Many researchers before now had considered some of these challenges and the level of their effects on new programmers but, little or no work had been done on scientific confirmation of adverse effect of sophisticated development environments on learning strength of the new programmers. Therefore, this paper provides a survey on the adverse effect of sophisticated development environments on novice programmers.

Integrated development environment (IDE) is a software system that provides comprehensive facilities to computer programmers for software development. During software development, there are many program building tools that a programmer uses in developing software applications. These program building tools are all integrated into an environment that is referred to as integrated development environment.

Some IDEs are graphically enriched while some are not. Based on the number of program building tools that are integrated together to form an environment, an IDE may be categorized as sophisticated or non-sophisticated.

Sophisticated development environment is an IDE that integrates many of the program building tools (debugger, compiler, editor and so on) and its features can only be easily understood by an experienced user (programmer) who is keen in using it for software development. Some of the integrated development environments in this category include; Microsoft Visual Studio, Netbeans, and so on. It is noted that majority of the graphical-based integrated development environments are sophisticated and support more than one language. However, the programming skills of some experienced programmers are heavily enhanced by these sophisticated IDEs while to the beginners, they serve as tools that do not enhance good programming skills due to their complexities. These complexities have always leave new programmers in a state where a single role of learning semantic and syntactic structures of programming language is doubled

with the cost of comprehending the complexities found by the IDEs to use.

Non-sophisticated integrated development environment integrates few of the program building tools and its features can be easily understood by a user more so a beginner who wants to use it for software development.

II. Literature Review

A. Program Building Tools

The integrated development environment as a system is made up of some program building tools seen as its components and among these are:

1) Source Code Editor

Source code editor is a text editor program designed specifically for editing source codes of computer programs by programmers [3]. Source code on the other hand is a file of computer instructions written in a particular programming language. Examples of source code editors are notepad, JEdit, notepad++ among others. Editors have auto-complete feature that automatically completes keywords or reserved words and also syntax highlighting feature that highlights reserved words or bugs in different colors as configured by the user.

2) Compiler

A compiler is a system software that translates a computer program written in a particular language (usually high level language) into an equivalent program in another language (machine language). Translation of programs here, is always done in its entirety before execution.

3) Interpeter

Interpreter is a software system that translates a computer program written in a particular high level language into an equivalent program in another language while the translation is done line by line or statement by statement.

4) Documentation Tool

This a program building tool that is used for documenting the source codes. It uses to make source code more understandable and clearer to programmer for proper debugging and for future referencing most especially in this generation of code reuse.

5) IntelliSense

IntelliSense is an integrated development environment feature that helps in automatic generation of code in the code editor. It helps to reduce time spent in typing code elements by programmers and helps to prevent the introduction of typographical errors in code.

6) Build Automation Tool

A tool that helps in scripting or automating a wide variety of compilation tasks, packing tasks, documentation tasks and development tasks that software developers do frequently[4].

B. Overview of the Benefits of integrated Development Environment

The benefits of integrated development environment as contained in [5] are:

1) Maximization of programmer's productivity with less effort

IDE provides some features that assist programmers in maximizing their productivities. Little input or effort is expected from the programmers and this little input yields maximized products with the help of the features contained in the IDE.

2) Reduction in software development time

With the available program building tools, development time becomes shorter. With this tools, many of the needed codes are auto-generated thereby reducing the lines of code to be edited manually by the programmer.

3) Enforcement of project or company standards

Standards may be enforced if the IDE offers predefined templates or blueprints or if code libraries are shared between different team members working on the same project.

4) Reduction in development stress

The stress passed through by programmers when developing complex software applications may be reduced with the help of IDEs.

C. Related Work

An exploratory study was carried out in [2] by computing department staff of National University of Samoa to investigate the most common errors students made in Java programming classes. There was an analysis of program code from undergraduate Java programming classes and results of analysis were used to form recommendation to inform courses' development and improve teaching practices.

In [6], a work on the review of literature relating to the psychological or educational study of programming for the purpose of identifying various problems experienced by novices was carried out. The problems include issues relating to basic program design, algorithm complexity in certain language features, fragility of novice knowledge and others but there seems to be little or no consideration on the complexity in IDEs. The researchers were later able to make some speculative observations and note possible topics for future work.

Survey on difficulties faced by students in learning programming were considered in [10]. These difficulties involved the IDEs that were used by the programmers under study. The result of the research according to their responses showed that IDEs have great influence on the learning curve of programming students. However, the research does not specify the type of IDE and the class of

influence it has on programming students. More so, the population under study in the research was specific to the concerned population.

As researched in [7], modern integrated development environments have tools that make recommendations and automate common tasks, such as refactoring, auto-completions, and error corrections. However, these tools present little or no information about the consequences of the recommended changes. Having to compute the consequences always puts an extra burden on the developers. But, the researchers developed a technique that could reduce the burdens encountered by the developers (who are believed to be experienced developers). This techniques informs developers of the consequences of any code transformation.

[8] conducted a one-on-one interviews with 22 freshmen who were taking their first Java programming courses with the objective of investigating not only which programming concepts or constructs most students had difficulties with, but why they found them difficult. The focus was on fundamental object oriented concepts and Java programming constructs such as, classes vs. objects, static data members vs. constant data members, constructors, access modifiers, syntax for method calls parameter passing, method overloading, inheritance, polymorphism, foreach vs. for loops, and the use of standard Java libraries. Students' misconceptions and missing conceptions in each of these concepts/constructs were described in detail in the research.

A research [9] addressed the problem of IDE interface complexity by providing a single window graphical user interface. The main goal of the research was to create a usable graphical interface design for IDE without tool views with the aim that it will reduce the number of open tool views functionality into the text editor. However, there was only a partial implementation of this single-window design in KDevelop IDE and there was a need for future work on suability testing of single window interface to find whether usability problems are solved or at least reduced compared to traditional IDE graphic user interface.

In [10], a research investigated and analyzed the problems faced by computer programming students at the

University of Tabuk with two main objectives (finding out whether the students at the University of Tabuk faced problems in computer programming similar to the ones faced by the students in different corners of the world as reported in the literature and studying the impact of sociocultural and environmental factors on learning computer programming skills by the students of the University. The researchers designed questionnaires (that contain questions pertaining to educational facilities such as curriculum and teaching materials, lab equipment and class rooms' environment, and the adequacy and proficiency of the professors and teaching assistants) to sample opinions and results from analysis of the questionnaires provided insight into the environmental and socio-cultural effects and the difficulties experienced while learning and teaching programming.

Collaborative learning and its potential positive effect on the learning outcomes of programming students was investigated in [11] while [12] described an investigation into the nature of the academic problems that face novice programming students. The later research analyzed the results of a survey given to students enrolled in an introductory programming unit across three campuses at Monash University in 2007. The survey focused on student perceptions of the relative difficulty in understanding and implementing both low level-programming concepts, such as syntax and variables, and

high level concepts, such as OOP principles and efficient program design. An analysis of the approximately 150 responses in the study indicated that a significant percentage of students experienced difficulties in high-level concepts.

To this point in time, it appears that little work has been done to directly consider the effect of IDEs most especially those with complex facilities on new programming students.

III. Materials and Method

A. Integrated Development Environment Classification

In this paper, the IDEs are classified into two based on the number of program building tools integrated into them. The two classes are non-sophisticated and sophisticated development environments.

1) Non-sophisticated development environment

It is an IDE that integrates few of the program building tools and its features can be easily understood by a new programmer who wants to use it for software development. An understanding of the technical knowhow of it requires little or no effort from the programmers who want to use it. Examples of these IDEs are:

a) Turbo Pascal IDE

Turbo Pascal IDE is an IDE that is capable of running on some operating systems among which are windows based, DOS, Macintosh [13]. It integrates tools like text editor, compiler, linker and few other tools. It needs little or no training from new programmers before its usage.

b) Qbasic IDE

It is also a simple to use IDE that does not require extra training before a new programmer can begin exploring its features and used for maximizing productivity. Few of the program building tools are integrated in it.

2) Sophisticated Development Environment

IDE in this class integrates many of the available program building tools and its features and usage can only be comprehensive to experienced programmers who are keen in using it for software development. It usage demands for extra training because without training, many of its features cannot be maximally utilized. Some IDEs in this class are:

a) ActivatdeState Komodo

ActivateState Komodo was developed by ActiveState [15]. It greatly supports Ruby, javaScript, XUL, Perl, Python, XHTML, CSS3 and Tcl. It is an IDE that is capable of running on some platforms such as Windows, Mac and Linux. It contains some features like auto-completion, syntax coloring as configured by users and source code control integration with subversion, Bazaar, Git, Mercurial, CVS and Perforce.

b) Netbeans

Netbeans is a multi-language IDE for developing primarily with Java, but also with some other languages, in particular, PHP, C/C++, and HTML, Ruby and so on. This IDE is written in java and can run on many operating system that are available in the market such as Windows, Linux, Solaris. Since it is a java application, it can run on any platform that supports java virtual machine [14]. This IDE, integrates many program building tools (code editor, intelliSense, and code refactoring tool, designer for designing GUI applications, web designer, class designer and database schema designer etc.) and its enhanced use needs training on how to use the available tools found in the environment. It is provides tools for developing applications for the three editions of java that is, standard, enterprise and micro editions.

c) Microsoft Visual Studio

This is an IDE developed by Microsoft Corporation [15]. It is used for developing both console and graphic based applications, Windows forms applications, web applications and web services applications like as we have in Netbeans. It includes program building tools like code editor, intelliSense, and code refactoring tool, designer for designing GUI applications, web designer, class designer and database schema designer [16]. It is multi-language integrated development environment that provides support for some languages among which are C, C++, VB. NET, C# and F# [15].

d) Eclipse

Eclipse is a multi-language IDE developed by free and open source software community and it is written in java [9]. It runs on some operating systems such as Linux, Mac OS X, Solaris and Windows. It supports development in languages like Ada, C, C++, COBOL, FORTRAN, Haskell, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy and Scheme [17].

e) Oracle JDeveloper

Oracle JDeveloper is an integrated development environment developed by Oracle Corporation. It is written in java language and it is a freeware type IDE [10]. The features provided by JDeveloper helps to support developments of software applications in some programming languages such as Java, XML, PL/SQL, JavaScript, SQL and PHP among others [18]. With JDeveloper, Oracle has aimed to simplify application development by focusing on providing a visual and declarative approach to application development in addition to building and advanced coding environment [18]. It integrates with the Oracle Application Development Framework, an end-to-end Java EE based framework that furthers simplifies application development for programmers [18].

3) Advantages of Sophisticated IDE on Experienced Programmers

The advantages of sophisticated IDE on experienced programmers include:

- Its effectiveness in developing complex software projects
- Its involvement in the reduction of software development time
- Its maximization of programmer's productivity with little effort
- Its involvement in stress reduction during software development
- Its appropriateness as the right choice for an experienced programmer who has vast knowledge with its use

4) Advantages of Non-sophisticated IDE

The following are few of the advantages of the non-sophisticated IDE:

- Its easiness with technical knowhow since few tools are integrated.
- Its right choice for a beginner
- Its power of computer resources' consumption in terms of memory, processors' cycle and so on.

5) Disadvantages of non-sophisticated IDE

To an experienced programmer, the following are the disadvantages of non-sophisticated environment:

- Its less effectiveness for complex software projects
- Its limitation in operation since few features are implemented in it.

B. Materials

During the course of studying the adverse effects of sophisticated IDEs on beginning programmers, we designed four hundred and twenty closed-ended questionnaires that contained questions as represented in table 1. These questionnaires were used in obtaining data from their primary sources (students and lecturers in programming related disciplines).

C. Methods

1) Data Collection

Data collection in this paper has been done through questionnaire. The population for study was carefully selected and it comprised of computer science and computer engineering undergraduates and lecturers of some tertiary institutions in Nigeria. These institutions

are Federal University of Technology, Akure (FUTA), Ladoke Akintola University of Technology, Ogbomoso (LAUTECH), Usmanu Danfodiyo University, Sokoto (UDUS), Interlink Polytechnic, Ijebu Jesa (IPI), Osun State College of Technology, Esa Oke and Federal Polytechnic, Ede (FEDPOEDE). The population of our study were of various academic levels who have in one form or the other made use of IDEs that fall into the categories of IDEs under study.

2) Data Presentation and Analysis

a) Data Presentation

Three hundred and eighty questionnaires were returned and this data as collected from the population were presented using simple pie chats as shown in fig 1. and fig 2. The collected data has been analyzed using a statistical tool called chi square as illustrated shortly.

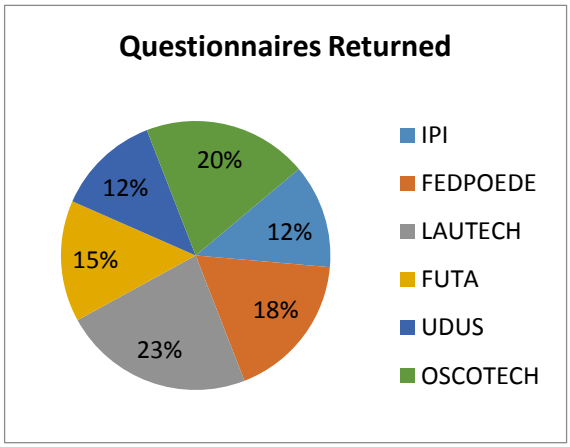

Fig.1. Presentation of returned questionnaires

Table 1. Results of responses as contained in questionnaires.

Questions (What are issues you have found difficult in learning programming so far?)	Question Code	Number of Students
The use of sophisticated or complex featured IDE	Q1	223
Learning language syntaxes	Q2	50
Gaining access to computer systems or networks	Q3	10
Learning language semantic structures	Q4	41
Designing a program to solve a task	Q5	30
Debugging	Q6	08
Lack of competent tutors	Q7	04
Lack of technical textbooks	Q8	05
Learning other program constructs such as comments and so on.	Q9	07
The use of non-complex featured IDE	Q10	02

b) Data Analysis

The levels of significance (α) in this analysis are 1% and 5% while the hypothesis are:

Hypothesis Tested

H_0: Sophisticated IDEs do affect learning strength of new programming students.

$H_{1:}$ Sophisticated IDEs do not affect learning strength of new programming students.

E= (380/10) =38;

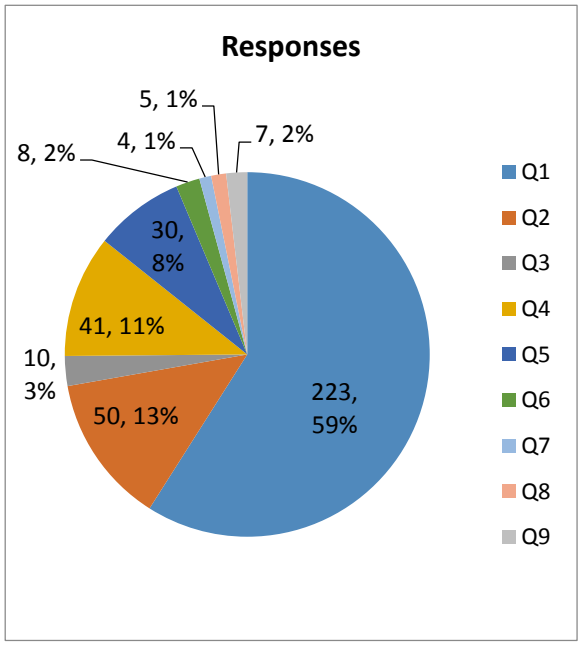

Fig.2. Responses as collected through questionnaires

$$X^2 = \sum ((O-E)^2/E) = 4.320247$$

From the table 2 of the Chi-square distribution,

$$D.f = X^2\,\alpha\,(n-1) = X^2\,0.01,\ 9 = 21.67$$
$$D.f = X^2\,\alpha\,(n-1) = X^2\,0.05,\ 9 = 16.92$$

Where,

O= observed frequency
E= expected frequency
α= level of significance
(n-1)= degree of freedom
X^2= goodness of fit, and
H_0 and H_1 are null hypotheses.

c) Discussion

Decision rule says, reject H_0 if $X^2_{calculated} > X^2_{tabulated.}$ With the two levels of significance (i.e. 1% and 5%) and tabulation done so far, since $X^2_{calculated}$ is less than $X^2_{tabulated}$ hence, we do not reject H_0 but conclude that Sophisticated IDEs do affect learning strength of new

programming students. According to the views of the population understudy, some of the adverse effects of sophisticated IDEs on novices among many include, difficulties with its technical knowhow, time wasted in understanding the complexity of the IDEs, a source of hindrance for learning new languages more so when they also require different IDEs, huge computer system resources consumed and multiple diversification of attentions to studying the IDE concerned and the work to execute.

Table 2. Tabulation of X^2

Problems	O	E	(O-E)	$(O-E)^2$	$(O-E)^2/E$
Q1	223	38	185	34225	153.4753
Q2	50	38	12	144	2.88
Q3	10	38	-28	784	78.4
Q4	41	38	3	9	0.219512
Q5	30	38	-8	64	2.133333
Q6	08	38	-30	900	112.5
Q7	04	38	-34	1156	289
Q8	05	38	-33	1089	217.8
Q9	07	38	-31	961	137.2857
Q10	02	38	-36	1296	648

IV. Conclusions

This academic work researched the adverse effect of integrated development environments that are built with complex program development tools on programmers who have just ventured into programming. This research has been done by sampling the opinions of some students and lecturers of programming related disciplines who are believed to be directly concerned. The data were sampled using questionnaires and chi square statistical tool was used for analyzing the collected data.

At the end of the research, it could be drawn that the integrated development environment whose purpose of development is to aid the programmers in developing effective and quick software applications might do the reverse if the programmers do not go for the right option. New beginning programmers are encouraged to be using non-complex IDEs at their elementary years of programming while complex IDEs can now be introduced at later years when they have experiences of programming. It will be a word of an advice to recommend a better development of other category of IDE that will bridge the gap between sophisticated and non-sophisticated IDEs. This is assumed to provide novices with little technical features and not advanced features as contained in the sophisticated IDEs.

Acknowledgment

We wish to acknowledge the effort of our academic mentor who is also our brother, Owoseni, Joshua O., Lecturer in the Department of Applied Geology, Federal University of Technology, Akure, Nigeria, whose ways of academic life challenge us a lot. One will be indebted if the indefatigable effort and support of Mrs C. O. Akanji of Plant Science and Biotechnology, Adekunle Ajasin University, Akungba, Nigeria is not acknowledged. We also wish to thank our Parents, Siblings, Computer Science Department Coordinator (Interlink Polytechnic, Nigeria), NDII and HNDI Computer Science Students of the above named department and to all that might have contributed directly or indirectly towards the success of this paper. Above all, to God be the glory.

References

[1] I. T. Chan Mow, "The Effectiveness of Cognitive Apprenticeship based Learning Environment (CABLE) in Teaching Computer Programming". *Unpublished PHD dissertation, University of South Australia*, 2006.

[2] I. T. Chan Mow, "Analysis of Student Programming Errors in Java Programming Courses," *Journal of Emerging Trends in Computing and Information Sciences*, (2012), Vol 3, No 5, Page 739-749.

[3] Source Code Editor, http://en.m.wikipedia.org/wiki/Source_code_editor.

[4] Build Automation, http://en.m.wikipedia.org/wiki/Build_automation.

[5] http://salfaris25.wordpress.com/2010/12/22/advantage-and-disadvantage-of-using-ide/

[6] A. Robins, J. Rountree and N. Rountree, "Learning and Teaching Programming: A Review and Discussion," *Computer Science Education*, (2003), Vol 13, No 2, Page 137-172.

[7] K. Muslu, Y. Brun, R. Holmes, M. D. Ernst and D. Notkin, "Speculative Analysis of Integrated Development Environment Recommendation," *OOPSLA' 12 Proceedings of the ACM international conference on object oriented programming systems languages and applications*, Page 669-682, ACM New York, NY, USA

[8] C. Chen, S. Cheng and J. Mei-Chuen Lin, "A Study of Missconceptions and Missing Conceptions of Novice Java Programmers," http://weblidi.info.unlp.edu.ar/worldcomp2012-mirror/p2012/FEC2866.pdf, retrieved on 1/07/2015.

[9] I. Ruchkin and V. Prus, "Single-window integrated development environment," http://arxiv.org/1207-1493.pdf, retrieved on 1/2/2015.

[10] M. M. Mhashi and A. M. Alakeel, "Difficulties Facing Students in Learning Computer Programming Skills at Tabuk University," *International conference, 12th, education and educational technology, recent advances in modern educational technologies*, (2013), Page 15-24.

[11] D. Teague and P. Roe, "Collaborative Learning-towards a solution for novice programmers," *ACE' 08 Proceedings of the tenth conference on Australian computation education*, Vol 78, Page 147-153.

[12] M. Butler and M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", *Proceedings of the 24th ascilite Conference*, (2007), Page 99-107.

[13] Turbo Pascal, http://en.m.wikipedia.org/wiki/Turbo_Pascal

[14] NetBeans, http://en.m.wikipedia.org/wiki/NetBeans

[15] http://www.microsoft.com/visualstudio

[16] Microsoft_Visual_Studio, http://en.m.wikipedia.org/wiki/Microsoft_Visual_Studio

[17] Eclipse, http://en.m.wikipedia.org/wiki/Eclipse_(software)

[18] JDeveloper, http://en.m.wikipedia.org/wiki/Oracle_JDeveloper

Authors' Profiles

Mr. A. T. Owoseni, is currently a lecturer at department of Computer Science Interlink Polytechnic, Nigeria. His research interests include multi-valued logic, artificial intelligence (AI), information retrieval, programming logic, software engineering, and database management system. He is currently a member of International Association of Engineers; International Association for Computer Science and Information Technology; and few societies of International Association of Engineers.

Mr. S. A. Akanji, an environmental statistician, currently lecturing at the department of mathematics and statistics, Interlink Polytechnic, Nigeria,
His research interest includes, Modelling Vehicular Emission in Comparative analysis of Statistical Neural Network and Classical Regression.