International Journal of Computer Applications Technology and Research
Volume 4– Issue 11, 829 - 837, 2015, ISSN:- 2319–8656

Effects of Object-Oriented Programming on Modern Software Development

Onu Fergus U.
Ebonyi State University, Abakaliki
Nigeria

Okorie Kingsley
10 Jideofor Street
Thinkers Corner
Enugu – Nigeria

Ugwa Chioma U
43/45 Unity Street off Nike Road Trans-Ekulu,
Enugu – Nigeria

Abstract: Object oriented Programming (OOP) entered the world of programming in the 1980s, but it actually became very popular from 1990s till date. What changes has it actually introduced into the programming world when placed in a balance with other programming techniques? This paper answers this question by taking a detailed and analytical look into the history and evolution of programming Languages. The paper unraveled the comparative advantages of OOP over a few earlier programming techniques. We collected data from both primary and secondary sources, analyzed the data and found out that stakeholders in software development industry have felt the tremendous impact of OOP on modern software development process. We have also noted that despite all the good and desirable features offered by Object Oriented programming, it is obvious that stakeholders in software development still expect easier and more flexible features than those the Object Oriented Programming currently presents.

Keywords: Object- based programming, procedure-based programming, software development, GSD

1. INTRODUCTION

Herbsleb and Moitra (2001), declared that software has become a vital component of almost every business in recent times. Success in all businesses increasingly depends on using software as a competitive weapon. This fact has made software development a construction of strategy for business success. Software development is a serious engineering concept that needs team work. The team members may be collocated or dispersed depending on where the needed expertise is found. The team-based approach to software development has made it a global concern. This is what Herbsleb and Moitra termed Global Software Development (GSD). In GSD distance is a major issue leading to coordination, communication, and management issues. To survive the problems posed by the globalisation of software development, collaborative tools among other technologies will play a key role concludes Herbsleb and Moitra (2001). Object Oriented Programming presents this collaborative feature and hence poses the tendency to provide great success in modern software development.

2 REVIEW OF RELATED LITERATURE

2.1Brief History of Computer Programming

Computer programming has undergone both tumultuous and turbulent moments which subjected programmers to rough learning and application terrain. It used to be so bad that one programmer may not and (most times) cannot comprehend what another programmer presented with their codes. To make the situation worse, the programmer who authored a set of codes would end up forgetting completely what the codes were to do few years later. The consequence of was a software or application which is very difficult (if not impossible) to maintain. Going down the memory lane, let us recall that programming the computer started with the novel works of the following: John Von Neumann, Charles Babbage, etc. The Assembly language, other procedural-based programming of 1950s, Structured programming techniques of late 1960s, Modular programming concepts 1970s and Object-Oriented programming of late 1980s have taken their turns to influence the computer programming and software development landscape.

Assembly Language: Assembly language as stated by Wikipedia, is a programming Language that can be used to directly tell the computer what to do. An assembly language is almost exactly like the machine language that a computer can understand, except that it uses words called *mnemonics* in place of numbers. A computer cannot really understand an assembly program directly. However, an assembler can easily change the program into machine code, replacing the mnemonic with the binary patterns they stand for. It is a programming language that is a close approximation of the binary machine code (Mifflin, 2009).

Procedure-Based programming: Many of today's programmers may not be used to some words such as routines, subroutines in programming as they are with functions and objects. These are what procedural programming paradigm, derived from structured programming which is built on the concept of the procedure calls are based on. In procedural programming, program codes are organized into small "procedures" that use and change our data. In ColdFusion for example, we write our procedures as either custom tags or functions. These functions typically take some input, carry out a process, and then produce some output. Ideally your functions would behave as "black boxes" where input data goes in and output data comes out. The key idea here is that our functions and procedures have no intrinsic relationship with the data they operate on. As long as you provide the correct number and type of arguments, the function/procedure will do its work and faithfully return the expected result. So in a procedural system our functions/procedures use data "given" to them (as parameters) but also directly access any shared data they need. (KevanStannard, 2011). Procedural programming presents a list or set of instructions telling a computer what to do step by step and how to move from the first line of code to the second line of code. Some examples include BASIC, Fortran, Pascal, C and Go.

Modular programming: Modular programming is a software design technique that emphasizes the separation of the functionality of a program into independent, interchangeable segments called *modules*. Theoretically, modules represent a separation of concerns, and improve maintainability by enforcing logical boundaries between components. Modules are typically incorporated into the program through interfaces. In the modular programming methodology, one of the most important principles is encapsulation, whereby the data contained within a module is accessible to the rest of the program only via behaviours of the module. This makes it much easier to control what happens when data is modified. Modular programming is a method for designing software by way of breaking up components of a large software program into manageable pieces. Those pieces, or "modules," can then be independently developed, tested, and refined. It's a process that generally helps shorten development time and avoid replicating code (WiseGEEK)

Structured Programming: Structured Programming is a technique for organizing computer program codes in a hierarchy of modules/segments showing how they are used. Each segment has a single entry and a single exit point. At the entry point, control is passed downward through the structure to higher levels of the structure without unconditional branches. Three types of control flow are used namely: sequential, test/selection, and iteration. Structured programming which includes modular programming, is a subset of procedure based programming enforces a logical structure on the program being written to make it more efficient and easier to understand and modify (Rouse, 2014)

Object-Oriented programming: Object Oriented Programming arrangements are in Classes and in Objects. An instance is classes of Human beings. These classes can have sub-classes; the sub-class can be Ebonyi girls. Classes inherit properties and behaviours so, Ebonyi girls like money because Ibo girls like money; Ibo girls like money because human beings like money. Object-oriented methodologies are an extension of modular ones, the additional ingredient being inheritance. With this ingredient, the code can refer directly to kinds of interactive things. The Object Oriented Programming ideals are. In object-oriented programming, concepts are directly molded in code employing the ideas of classes and inheritance. A distinct member of a class is called an object.

2.2 Features of Object-Oriented Programming

The concept of Object-Oriented Programming is very interesting and creates a lot of ease in the programming process. It includes the concept of:

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism

Objects: Objects are the basic run-time entities in an object-oriented system. Programming problem is analyzed in terms of objects and nature of communication between them. When a program is executed, objects interact with each other by sending messages. Different objects can also interact with each other without knowing the details of their data or code.

Classes: A class is a collection of objects of similar type. Once a class is defined, any number of objects can be created which belong to that class.

Data Abstraction and Encapsulation: Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes. Storing data and functions in a single unit (class) is encapsulation. Data cannot be accessible to the outside world and only those functions which are stored in the class can access it.

Encapsulation: Once an Object is created, knowledge of its implementation is not necessary for its use. In older programs, coders needed understand the details of a piece of code before using it. Objects have the ability to hide certain parts of themselves from programmers. This prevents programmers from tampering with values they shouldn't. Additionally, the object controls how one interacts with it, preventing other kinds of errors. For example, a programmer (or another program) cannot set the width of a window to -400. (Popyack, 2012)

Inheritance: Inheritance is the process by which objects can acquire the properties of objects of other class. In object oriented programming, inheritance provides reusability, like, creating additional features to an existing class without modifying the class. This is achieved by deriving a new class from the existing one. The new class will have combined features of both the classes.

Polymorphism: Polymorphism means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation. Polymorphism is extensively used in implementing Inheritance.

2.3 Inherent Pros of Object-Oriented Programming

Code Reuse and Recycling: Objects created for Object Oriented Programs can easily be reused in other programs.

Design Benefits: Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with fewer flaws. In addition, once a program reaches a certain size, Object Oriented Programs are actually easier to program than non-Object Oriented ones. (Jeffrey L. Popyack)

Software Maintenance: Programs are not disposable. Legacy code must be dealt with on a daily basis, either to be improved upon (for a new version of an exist piece of software) or made to work with newer computers and software. An Object Oriented Program is much easier to modify and maintain than a non-Object Oriented Program. So although a lot of work is spent before the program is written, less work is needed to maintain it over time. .(Jeffrey L. Popyack)

2.4 Procedure-Based Versus Object-Based Programming

Procedural Programming lays emphasis on identification and specification of a set of steps to solve a given task and the precise way to execute it to reach the desired outcome. For example, if you want to calculate the month-end closing balance of the departmental imprest, the steps would follow thus:

- Assign the initial monthly-allocation
- Sum up all the expenses within the month
- then subtract the sum of the expenses from the initial monthly allocation
- the subtraction will give you the month-end closing balance

A procedure which can be a subroutine or a function contains an ordered list of instructions to be carried out. A procedure can be called at any time during the execution the program by any other procedure or by itself. Let us note at this point that Procedure-based and Object-based programming are two ways of showing problems to be solved and how to go about solving them.

The key difference between Object Oriented Programming and Procedural Programming is that the focus of Procedural Programming is to break down the programming task into a collection of variables and subroutines while, the focus of Object Oriented Programming is to break down the programming task into objects, which encapsulate data and methods. Most notable difference could be that while Procedural Programming uses procedures to directly operate on data structures, Object Oriented Programming will bundle the data and methods together so that an object will operate on its own data (KevanStannard, 2011).

3. METHODOLOGY

We studied the effect of OOP on modern software development with data from two main sources as discussed below.

Primary source: We conducted oral interviews with a total of 125 interviewees out of which 10 were Lecturers, 15 were freelance indigenous software developers in Enugu Nigeria and 100 were final year students of computer Science. These respondents were selected from three different Universities namely: Ebonyi State University Abakiliki - Nigeria, Nnamdi Azikiwe University, Awka - Nigeria and Enugu State University of Science and Technology, Enugu - Nigeria. The questions sought the views of the above named groups of persons' on the effects of Object-Oriented Programming on modern software development.

Secondary source: we consulted existing documents like computer science journals, text books, laboratory manuals and manuscripts, etc. The Internet was a major source of the secondary data source. Most of our journal articles were sourced through the internet. We studied various reviews and comments from people of all walks of life.

4.0RESULTS PRESENTATION & SUMMARY OF FINDINGS

Table 1 shows the occupational distribution of the interviewee. A total of 125 respondents' opinions were sampled and responses collected and analyzed on a 5-point *linker type* scale as shown subsequently.

S/n	**Respondents' Occupation**	**No.**	**Percentage**
1.	Software developers	15	8%
2.	Lecturers	10	12%
3.	Final year students	100	80%
	Total	**125**	**100%**

Table 1: Occupation distribution of interviewed respondents

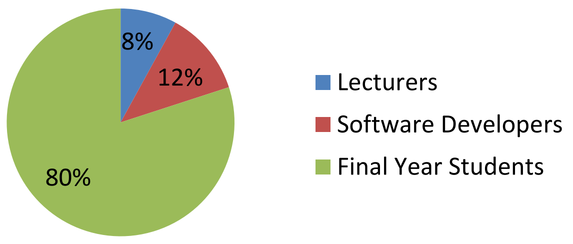

Figure 1: Pie Chart Showing the Occupational distribution of interviewed respondents

Table 2 presents the opinion of the respondents to the questions presented by the interviewer. These questions were presented to indigenous software developers, Lecturers and final year students of Computer Science selected from three universities mentioned earlier.

Table 2: Questions and responses by respondents

S/N	Questions	x	F	fx	χ (mean)	%
1	Object oriented programming languages enable programming in modules?					
	Strongly Agree	5	100	500	4.80	80.00
	Agree	4	25	100		20.00
	Undecided	3	0	0		0.00
	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
2	Program can be divided into units of task and tested differently in OOP?					
	Strongly Agree	5	83	417	4.60	66.67
	Agree	4	33	133		26.67
	Undecided	3	8	25		6.67
	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
3	Object oriented programming languages have the concept of objects, class, data abstraction, inheritance, and polymorphism?					
	Strongly Agree	5	100	500	4.73	80.00
	Agree	4	17	67		13.33
	Undecided	3	8	25		6.67
	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
4	It is easy to debug programs written with object oriented programming than procedure oriented programming?					
	Strongly Agree	5	83	417	4.53	66.67
	Agree	4	25	100		20.00
	Undecided	3	17	50		13.33
	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
5	Tasks are broken into smaller units in object oriented programming?					
	Strongly Agree	5	17	83	3.40	13.33
	Agree	4	17	67		13.33
	Undecided	3	92	275		73.33

	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
6	Object oriented programming languages have graphical interface that allow for the design of a more attractive program interface than in procedure oriented programming?					
	Strongly Agree	5	25	125	3.33	20.00
	Agree	4	42	167		33.33
	Undecided	3	17	50		13.33
	Disagree	2	33	67		26.67
	Strongly Disagree	1	8	8		6.67
7	Object oriented programming aids team work better than procedure oriented programming?					
	Strongly Agree	5	67	333	4.53	53.33
	Agree	4	58	233		46.67
	Undecided	3	0	0		0.00
	Disagree	2	0	0		0.00
	Strongly Disagree	1	0	0		0.00
8	Object oriented programming makes it easy to modify existing codes for a re-use?					
	Strongly Agree	5	67	333	4.33	53.33
	Agree	4	42	167		33.33
	Undecided	3	8	25		6.67
	Disagree	2	8	17		6.67
	Strongly Disagree	1	0	0		0.00
9	Objects created for object oriented programs can easily be reused in other programs?					
	Strongly Agree	5	58	292	4.13	46.67
	Agree	4	42	167		33.33
	Undecided	3	17	50		13.33
	Disagree	2	0	0		0.00
	Strongly Disagree	1	8	8		6.67
10	In object oriented programming, once an object is created, knowledge of its implementation is not necessary for its use?					
	Strongly Agree	5	50	250	3.87	40.00
	Agree	4	33	133		26.67
	Undecided	3	17	50		13.33
	Disagree	2	25	50		20.00
	Strongly Disagree	1	0	0		0.00

4. DISCUSSION

We can see from table 2 that all the questions/factors listed for the opinion of the respondents were turned-in with high arithmetic mean. Exceptions to these were two factors namely: *presence of graphical interface that allow for the design of a more attractive program interface than in procedure oriented programming* and *breaking of tasks into smaller units*. These factors have arithmetic mean of 3.33 and 3.40 respectivey. That shows that respondents do not see these factors as a positive impact that OOP has brought to software development. Those factors really do exist in the previous programming techniques.

All the other factors that were tested had high positive impact as they presented with high arithmetic mean. These showed that Object-Oriented Programming has made a very remarkable positive impact on modern software development. The impact has brought about resultant increase in the production of millions of software on daily basis all over the world. Object-Oriented Programming has been very successful and possibly more successful than other conventional programming approaches.

The later popularity of OOP notwithstanding, a few issues affected its popularity in its early stage. These issues were: technological shortcomings such as disc space and energy/time dissipated in program planning and design. OOP has brought a lot of changes into computer programming in particular and the world of software development in its entirety. Suffice it to say in a nut shell that Object Oriented Programming has brought the dawn of a new epoch in the software development world.

Encapsulation, inheritance, instance, abstraction the ***reuse property,*** etc were some of the key characteristics of OOP that made it stand out from the earlier programming techniques. These properties were so unique and interesting that their usage in software development heralds a revolution in the software industry. Deploying Object Oriented Programming paradigm in software development saves a lot of code through the reusable of components, frameworks and designs. Object Oriented Languages made available generic templates and saved the time and space needed for code duplication.

5. CONCLUSIONS

There is so much software now than has ever been in the world of software development and in the world in general. Is it possible that this is as a result of time and chance? What do we attribute these daily churning of software to? We attribute these changes and the ease of software development to object oriented programming and all the easy manipulations of data and functions. A glimpse at our oral interview result shows in every area that Object Oriented Programming is preferred by all the three groups of respondents. In view of the above we affirm that the effects of Object Oriented Programming on modern software Development is very positive and makes software development very fast to development.

6. REFERENCES

1. Abadi, Martin; Cardelli, Luca (1996). A Theory of Objects. Springer-Verlag New York, Inc.
2. Ambler, Scott (1st January 1998). "A Realistic Look at Object-Oriented Reuse". www.drdobbs.com.
3. Armstrong, Joe. In Coders at Work: Reflections on the Craft of Programming. Peter Seibel, ed. Codersatwork.com.
4. Booch, Grady (1986). Software Engineering
5. Boronczyk, Timothy (11 June 2009). "What's Wrong with OBJECT ORIENTED PROGRAMMING", zaemis.blogspot.com.
6. Brooks, Fred P. (April 1987). "No Silver Bullet — Essence and Accidents of Software Engineering".
7. C. J. Date, Hugh Darwen. Foundation for Future Database Systems: The Third Manifesto (2nd Edition)
8. C. J. Date, Introduction to Database Systems, 6th-ed., Page 650
9. Cambridge: Prentise Hall International Series in Computer Science. p. 23.
10. Cardelli, Luca (1996). "Bad Engineering Properties of Object-Oriented Languages".
11. Computation Center and Research Laboratory. p. 88f. "In the local M.I.T. patois, association lists [of atomic symbols]
12. Conway, Richard (1978). A primer on disciplined programming using PL/I, PL/CS, and PL/CT. Winthrop Publishers.
13. Dr. Alan Kay on the Meaning of "Object-Oriented Programming"". 2003.
14. Graham, Paul. The Emerald Programming Language". 2011-02-26
15. 1995 Reviewers Guide to Visual FoxPro 3.0: DFpug.de
16. Hoare, C. A. (Nov 1965). "Record Handling". ALGOL Bulletin (21): 39–69. doi:10.1145/1061032.1061041.
17. Holmevik, Jan Rune (1994). "Compiling Simula: A historical study of technological genesis". IEEE Annals of the History of Computing 16.
18. Jacobsen, Ivar; Magnus Christerson; PatrikJonsson; Gunnar Overgaard (1992). Object Oriented Software Engineering. Addison-Wesley ACM Press. pp. 43–69.
19. James, Justin (1 October 2007). "Multithreading is a verb not a noun". Techrepublic.com. Archived from the original on 2 January 2013.
20. John C. Mitchell, Concepts in programming languages, Cambridge University Press, 2003, p.278

21. Kay, Alan. "The Early History of Smalltalk". September 2007.

22. Kindler, E.; Krivy, I. (2011). Object-Oriented Simulation of systems with sophisticated control**.** *International Journal of General Systems*. pp. 313–343.

23. Lewis, John; Loftus, William (2008). Java Software Solutions Foundations of Programming Design 6th ed. Pearson Education**.**

24. M.Trofimov, OBJECT ORIENTED PROGRAMMING - The Third "O" Solution: Open OBJECT ORIENTED PROGRAMMING. First Class, OMG, 1993, Vol. 3, issue 3, p.14.

25. Meyer, Bertrand (1988). Object-Oriented Software Construction.. p. 105. "Object - a synonym for atomic symbol" Meyer, Second Edition, p. 230

26. Michael Lee Scott, Programming language pragmatics, Edition 2, Morgan Kaufmann, 2006, p. 470

27. Neward, Ted (26 June 2006). "The Vietnam of Computer Science". Interoperability Happens.

28. Pierce, Benjamin (2002). Types and Programming Languages. MIT Press. section 18.1 "What is Object-Oriented Programming?"

29. Poll, Erik. "Sub typing and Inheritance for Categorical Data types".

30. Potok, Thomas; MladenVouk; Andy Rindos (1999). "Productivity Analysis of Object-Oriented Software Developed in a Commercial Environment". Software

31. Rich Hickey, JVM Languages Summit 2009 keynote, Are We There Yet? November 2009.

32. Robert Harper (17 April 2011). "Some thoughts on teaching FP". Existential Type Blog.

33. Ross, Doug. "The first software engineering language". LCS/AI Lab Timeline:.

34. Shelly, Asaf (22 August 2008). "Flaws of Object Oriented Modeling". Intel Software Network.

35. Shelly, Asaf (22 August 2008). "HOW TO: Multicore Programming (Multiprocessing) Visual C++ Class Design Guidelines, Member Functions". support.microsoft.com.

36. Stepanov, Alexander. "Stlport: An Interview with A. Stepanov". Sutherland, I. E. (30 January 1963). "Sketchpad: A Man-Machine Graphical Communication System"