Semantic Analysis of Object-Oriented Programming Languages: Survey

Abdulkadir Abubakar Bichi[1], **Abdulrauf Garba Sharifai**[1], **Saud Adam Abdulkadir**[1]

Department of Computer Science, Northwest University, Kano Nigeria

Abstract: ***Object-oriented programming (OOP) is a programming paradigm that uses a special data structure called objects which encapsulate the data fields and procedures together with their interactions for writing a computer program. This paper is a survey of ten (10) Object oriented programming languages: C++, C#, Java, Modula, Delphi, Ada, Eiffel, VB.Net, Python and Smalltalk. The work involved a comparative semantic analysis of the ten programming languages with respect to the following criteria: support of inheritance, encapsulation, operations and method overloading among others. Smalltalk and Eiffel are pure OOP languages, but Eiffel is more powerful in terms of inheritance since it supports both single and multiple inheritance and support class variable/method. C++ and Java are hybrid languages; they support most of the OOP features but not all. However, Java has higher degree of objectivity since it supports total objectivity of user defined and have a good technique for garbage collection, C++ also has many powerful features like operation and method overloading, flexibility in binding and multiple inheritance support, though difficult to use and prone to errors, it is powerful and complex at the same time. Python can also be considered as hybrid since it lack the feature of total object of operation message but has higher degree of objectivity compare to both C++ and Java. C# is like an improved version of C++ that solve many complications of the later language. C# behaves similar to Java but support of more features like method overloading.***

Keywords: Abstract Data Type, Inheritance and Polymorphism

1. Introduction

Object-oriented programming (OOP) is a programming paradigm that uses a special data structure called objects which encapsulate the data fields and procedures together with their interactions for writing a computer program [1]. Though it was invented with the creation of the Simula language in 1965, and further developed in Smalltalk in the 1970s, it was not commonly used in mainstream software application development until the early 1990s[2]. Manytodays modern programming languages support the features of OOP. The degree of objectivity of the language is measure in terms of functionality features such as: Abstract data type, inheritance, Polymorphism. All pre-defined data types, operations performed by sending messages to Objects and all user-defined data types [3].

2. Types of OOP Languages

The OOP languages can be classified based on its degree of object orientation;this can be put as:

1) **Pure:** A Pure Object-Oriented language is the one that satisfied all the above six features [4]. The languages treateverything in them consistently as an object, from primitives such as characters and punctuation, all the way up to whole classes, prototypes, blocks, modules, etc. They were designed specifically to facilitate, and even enforce OO methods. Examples: Smalltalk and Eiffel.
2) **Hybrid:** The hybrid language usually refers to the one that satisfied the first three features but may lack some of the last three. Languages designed mainly for OO programming, but with some procedural elements. Examples: C++, C#, Java, Ada, Python.
3) **Partial:** Languages usually supports encapsulation (abstract data type) and one more feature, but not all features of object-orientation, they are sometimescalled object-based languages. Examples: Modula-2 and VB.NET.

3. General Features of OOP Languages

Object oriented programming involves three fundamental concepts: Abstract Data Types (ADT), Inheritance and Dynamic binding.

1) **Abstract Data Type (ADT)** is a set of values (the carrier set) and a collection of operations to manipulate them. Every programming language provides some built-in data types (like integers and floating-point numbers) that can be instantiated as needed. Objects may either be statically or dynamically instantiated. Statically instantiated objects are allocated at compile-time and exist for the duration that the program executes. Dynamically instantiated objects require run-time support for allocation and for either explicit deallocation or some form of garbage collection [5].
2) **Inheritance** is a means by which a new abstract data type can inherit the data and functionality of some existing type and is also allowed to modify some of those entities and add new entities.
   a) Single Inheritance: if a new class is a subclass of a single parent class.
   b) Multiple Inheritance: if a new class has more than one parent class.

The process of deriving a whole new class from the base class has various advantages: the execution time would get reduced, the possibilities of errors will be less as we are using the same attributes which have been compiled by the system and, they are error free [6]. With the help of the existing class, we can create a new class using same function. The existing class will be used as a reference class for the derived class. To organize our grammar or language in a systematic way in the program, we must use inheritance. The reusability of the same grammar in the new sub-class would reduce execution and compilation time [7]. Reusing the attributes, methods make the task of programming easy and the reusability of the code from our existing class to our new derived class is very feasible for the users [8] By making the use of inheritance, we save the execution time and the programming complexity.

The dynamic binding also sometimes called dynamic dispatch is the process of linking procedure call to a specific sequence of code (Method) at run time. In other words, Dynamic Binding means that a block of code executed with reference to a procedure call is determined at run time. The code usage gives the programmer to utilize the time and get fast results [9]. We could define only one singular code for many different types of variables to perform the action of execution quickly. By allowing common functions to work for various objects we avoid the complexity and bind objects together. Binding of objects makes the task of giving definition easy, which is very necessary if we have to write long codes for programs [10]. By giving access to users on defining their objects limits to certain classes, the user gets the leverage to use the same object at different positions with different functions. It joins the derived classes together which gives us a clear picture of what polymorphism and its importance in OOP [11].

4. Related Works

This section reviews some related works pertinent to Survey on Concepts of Object Oriented Programming Language, Maya Hristakeva, RadhaKrishnaVuppala[12]. Discusses Object-oriented programming and how it become important programming paradigm of our times. From the time it was brought into existence by Simula, object-oriented programming has seen wide acceptance. Object-oriented programming languages (OOPLs) directly support the object notions of classes, inheritance, information hiding (encapsulation), and dynamic binding (polymorphism). This paper in detailed look at some of the concepts considered fundamental to object-orientation, namely Abstract data type, inheritance and polymorphism. Different aspects of inheritance and polymorphism are implemented in various popular OOPLs. This work concludes with the observation that there is still lot of work to be done to reach a common ground for these crucial features of OOPLs. This survey presents a detailed comparison of 7 Object Oriented Programming Languages Java, Smalltalk, C++, C#, Eiffel, Ruby and Python in terms of their inheritance and polymorphism implementations. The paper also presents a compilation of the observations made by several earlier surveys

OscarNierstrasz [5] discussed various concepts of object-oriented programming language and the importance of those mechanisms in the real world environment. OOP becomes popular not only by the programmers who use it to solve their problem but also because of its flexibility over user interfaces, operating systems and databases. Smalltalk effort is a main reason for the popularization of the OOP which utilizes both the classes and inheritance concept. Simula was the first programming language which uses an object as a programming construct. The creation of the object is not only flexible for programmers; it is also useful for prototyping and application development. The most important concepts that are supported by the OOP are encapsulation and inheritance. Encapsulation is used to hide the behaviour of objects, whereas the behaviour of properties can be shared among objects through the inheritance. This work studies mainly about the reusability properties of objects, its various types and its concurrency properties.

5. Comparative Semantics Analysis

S/N	Language	Encapsulation	Inheritance	Polymorphism
1	C++	Parentless class: yes Generic class: yes Class Variables/ Methods: yes Garbage collection: none Total objectivity of user defined: no Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: yes Inheritance based: class	Binding; both Operator overloading: yes Method overloading: yes
2	Java	Parentless class: no Generic class: no Class Variables/ Methods: yes Garbage collection: Mark and Sweep or Generational Total objectivity of user defined: yes Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: virtual Inheritance based: class	Binding: static Method overloading: yes Operator overloading: no
3	Python	Parentless class: yes Generic class: no Class Variables/ Methods: no Garbage collection: Reference Counting Total objectivity of user defined: yes Total objectivity of pre-defined type: yes Total objectivity of operation message: no	Single inheritance: yes Multiple inheritances: yes Inheritance based: both	Binding: dynamic Method overloading: no Operator overloading: yes
4	Smalltalk	Parentless class: No Generic class: no Class Variables/ Methods: yes Garbage collection: Mark and Sweepor Generational Total objectivity of user defined: yes Total objectivity of pre-defined type: yes	Single inheritance: yes Multiple inheritance: no Inheritance based: class	Binding: dynamic Method overloading: no Operator overloading: yes

		Total objectivity of operation message: yes		
5	C#	Parentless class: No Generic class: no Class Variables/ Methods: yes Garbage collection: Mark and Sweep or Generational Total objectivity of user defined: yes Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: virtual Inheritance based: class	Binding: both Method overloading: yes Operator overloading: yes
6	VB.net	Parentless class: No Generic class: no Class Variables/ Methods: no Garbage collection: Reference Counting Total objectivity of user defined: yes Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: virtual Inheritance based: class	Binding: static Method overloading: yes Operator overloading: yes
7	Delphi	Parentless class: Yes Generic class: Class Variables/ Methods: Garbage collection: reference counting Total objectivity of user defined: no Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: virtual Inheritance based: class	Binding: both Method overloading: yes Operator overloading: yes
8	Modula-3	Parentless class: yes Generic class: yes Class Variables/ Methods: yes Garbage collection: none Total objectivity of user defined: no Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: no Inheritance based: class	Binding: dynamic Method overloading: Operator overloading:
9	Eiffel	Parentless class: Yes Generic class: yes Class Variables/ Methods: no Garbage collection: Mark and Sweep or Generational Total objectivity of user defined: yes Total objectivity of pre-defined type: yes Total objectivity of operation message: yes	Single inheritance: yes Multiple inheritance: yes Inheritance based: class	Binding: dynamic Method overloading: no Operator overloading: yes
10	Ada	Parentless class: no Generic class: yes Class Variables/ Methods: yes Garbage collection: none Total objectivity of user defined: no Total objectivity of pre-defined type: no Total objectivity of operation message: no	Single inheritance: yes Multiple inheritance: virtual Inheritance based: class	Binding: both Method overloading: yes Operator overloading: yes

The degree of objectivity is not enough to conclude one language is better than the other without extensively discussing the other factors like: generic class, inheritance mechanism, support of multiple inheritance, Operator overloading,Method overloading and technique of garbage collection. Smalltalk and Eiffel are pure OOP languages, but Eiffel is more powerful in terms of inheritance since it support both single and multiple inheritance and less complication in binding since it use static binding but unlike Smalltalk does not support class variable/method. C++ and Java are hybrid languages they support most of the OOP features but not all. However, Java has higher degree of objectivity since it supports total objectivity of user defined and good technique of garbage collection, but C++ has many powerful features like operations and method overloading, flexibility in binding and multiple inheritance support thus make it powerful and complex and the same time; is good for complex task but difficult to use and prone to errors. Python too can be considered as hybrid since it lacks the feature of total object of operation message but has higher degree of objectivity compare to both C++ and Java. C# is like an improve version of C++ will solve many complications of the later language is works very similar to Java but support of more features like method overloading

6. Conclusion

Generally, OOP languages are sophisticated and more difficult compare to other programming paradigms but has powerful features that gives you a handleover complex tasks. There is no direct answer of which language is the best among the OOP languages but generally the pure OOL are easier and more efficient to implement object-oriented algorithms. C++ has a lot of features that make the language powerful but with many complications and prone to errors, thus make it difficult to work with. Java and C# appear to be an improvement over C++ to solve the complications but compromised many of the powerful features.

References

[1] Oscar Nierstrasz, A Survey of Object-Oriented Concepts, University of Geneva†

[2] Makkar, G., J.K. Chhabra, and R.K. Challa. Object oriented inheritance metric-reusability perspective. In Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on. 2012.

[3] Mernik, M., et al. The template and multiple inheritance approach into attribute grammars. In Computer Languages, 1998. Proceedings. 1998 International Conference on. 1998.

[4] Rathore, N.P.S. and R. Gupta. A novel coupling metrics measure difference between inheritance and interface to find better OOP paradigm using C#. In Information and Communication Technologies (WICT), 2011 World Congress on. 2011.

[5] Milojkovic, N., et al. Polymorphism in the Spotlight: Studying Its Prevalence in Java and Smalltalk. In Program Comprehension (ICPC), 2015 IEEE 23rd International Conference on. 2015.

[6] Rountev, A. and A. Milanova. Fragment class analysis for testing of polymorphism in Java software. in Software Engineering, 2003. Proceedings. 25th International Conference on. 2003.

[7] Hang, Z., H. Zhiqiu, and Z. Yi. Polymorphism Sequence Diagrams Test Data Automatic Generation Based on OCL. In Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for. 2008.

[8] Maya Hristakeva, RadhaKrishnaVuppala. A Survey of Object Oriented Programming Univ. of California, SantaCruz