Abstract

Object oriented programming paradigm in software development is one of the most popular methods in the information technology industry and academia as well as in many other forms of engineering design. Software development is a field of engineering that came into existence owing to the various problems that developers of software faced while developing software projects. This paper analyzes some of the most important technological innovations in object oriented software engineering in recent times. The advancements in Object technologies that have been analyzed here include object cloning, introspection and reflection, class co-evolution, global software development contexts, interfaces, namespaces, query-able source codes, meta-model for generating design alternatives, magic methods, design pattern detection, auto-active functional verification, cohesion, coupling and Separation of Concerns (SoC). From our analysis we predict further advancements in object technologies towards game development, metrics for software design analysis, addition to fundamental Object oriented programming language features and distributed software development.

Keywords: Application, Object, Oriented, Paradigm, Software, Development, Concepts

1.1 Introduction

Object oriented programming (OOP) is like an element of nature in the computing programming world where the multidiscipline domain of life and professional fields of engineers, educationist, scientist, project management, programmers, system analyst and field researchers has been involved. In recent time, OOP and Software development industries are gaining advantages where the two phenomena are emerging and converging to produce efficiency in industries software that helps to solve specific organizational and personalization problems. Take for instance, a lot of sophisticated apps are being produced and release in the market today through the application of OOP. Almost desktop apps are being converted to mobile apps through Java, C++, PHP & MySQL, R, Python etc platform which form the testimony of OOP in the software industries. Software developer has been revolving using procedural language for the past decade before the advent of OOP. The relationships between them is that procedural languages focus on the algorithm but OOP focuses on the object model itself, therefore, making the software programming more visible and increasing the efficiency and usability of the existing object thereby reducing cost. The significant aspect of object oriented objects is character, attributes and their modeling of the software system, since, internal structure is time-consuming and often imposes requirement barriers for development. Objects is like blood capsule in OOP, it map character with behaviour, thus enable software programmers to map a real problem implementation and execution in the problem domain. The concepts of OOP [1]–[7]such as inheritance, encapsulation and polymorphism enable the successful implementation of objects access and modification. Therefore, the phenomenon by which programming method code is designed based on the functions and attributes of the objects is classified as object oriented programming [9]. Object oriented programming bridges the creation of intellectuals object that model a business problem we trying to solve [10]. In the same vein, the process by which of programming approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of that modules on demand by the user is referred to as object programming language [2]. This programming style revolves round the generation of programming language, from 1st – 5th generations. Figure 1 summarizes the generation of computer programming language [11], [12]:
Computer Languages 1st – 5th generations

a. First Generation Languages (1GL): This is the machine languages era which is consist of 0s and 1s corresponding to the instruction set that is hard wired into the security of a microprocessor. Examples: ALGOL 58, FORTRAN 1

b. Second Generation Languages (2GL): This is the era of assembly languages or low level language corresponding 1 to 1 basis to a machine language instruction. Examples: ALGOL 60, FORTRAN II, COBOL

c. Third Generation Languages (3GL): This is the High level languages dated as earlier as 1950’s. The improvements of 1st and 2nd generation resulted to 3rd generation which is highly used in business and scientific applications. Examples: C, C++, Java, JavaScript, Visual Basic

d. Fourth Generation Languages (4GL): This is high level in nature. It consists of statements similar to statements in a human language. It eliminates many of the strict syntax and semantics grammar rules which imposed barriers in 3rd generation languages. They are also high level languages. Examples: PHP, SQL, Perl, Python, Ruby

e. Fifth Generation Languages (5GL): This contains visual tools to help develop a program. This is aspect solely dedicated in solving problems. It does not follow algorithm written by a programmer but follow the constraints given to the program. Examples: Prolog, Mercury, OPS5.

The essence of the new paradigm is to solve software complexity (such as lines of code, number of classes, number of modules, module interconnections and dependencies, and time to understand[14] through the needs to alter the style of programming. The essence is to: (i) produce software efficiency and reliability (ii) develop reusable software modules (iii) production cost reduction (iv) reduce cost of maintenance (v) completion time of software development necessitated.

The knowledge of object-oriented programming paradigm in software development is numerous. OOP structuring provides a closer representation of reality than other programming techniques thereby solving complex software systems. A proper analyzed OOP system can implement changes at the class level without affecting the other level in the system development, thereby reducing maintenance cost. This is form a formidable computing power and advantageous of OOP in the industries software development problem solving. The concepts of OOP, the polymorphism and inheritance, allows the reuse individual components, hence, reduces the development time of OO software. OOP also helps in reducing rate of work involved in revising and maintaining the system because during the design state, problems are easily debugged [1].

2.1 The Generic of Software development Project

Software Development Project is not an easy process which is made up of many phases. Each phase requires a lot of sketch and report in all the development initiative processes. The expert and the novice do not think alike in software industry with respect to code generation during implementation. Each software development project has to go through about eight phases[15] as follows:


Figure 1 depicts a typical development process for a new product. Multiple cycles might run during the testing phase as software testers/developers find problems, bugs and fix them before a software product is fully deployed in the market. Figure 1 clearly shows the various phases and the interaction
with the customer. The software release runs through the customer to support and move down to new features, etc. Each phase is slightly dependent on other phases for the software development project to be achieved efficiently.

2.2 Super Model of Software development

Software development should always follow a particular software development life cycle model like the waterfall model and so on in the development of an application from its initial feasibility study down to its deployment and maintenance in the field. There are many other models that could be used to explain various approaches during SDLC process. The SDLC model is widely used during software development because it makes it easier to describe the steps that are followed within the life-cycle framework.

The SDLC targeted in producing high-quality software that meets or exceeds customer expectations, reaches completion within time, and cost estimation, and is directly related to the customer as well as organizational satisfaction. Three major phases are identified in super model of software development. From figure 2; the first phase is design and communication, after analysis requirements are collected and design. The second phase starts with coding. The programmer transforms the design into a programming notation. The developed code is tested to ensure it meets the usability and software requirement specification. Once, the deployment test result meets the customer’s expectation then it moves to the final phase. The final phase is the third phase which propels the software to be released in the market based on the nature of the stakeholder involved by the project manager. All processing phases supported by time-boxing and smart project management[16]. The super model diagram is depicted in figure 2 below:

Figure 1: Software Development Projects

Figure 2: Software Development Super Model
2.3 Features of Object-Oriented Programming

The fundamental features or concepts of object-oriented programming [17][9], [18], [19]:


(i) **Data abstraction**: is the concept of OOP that focuses on the vital attribute and behaviour of an object. Abstraction focuses only on complex ideas and ignores irrelevant details that would confuse us when handling real world problems. It shows only the relevant data and hides unnecessary details of an object from the operator. From the figure 3, the features of the car are the abstraction. In this context, an attribute could be any character of an object that affects the state or data as well as the operation reflecting a method (or behavior or function). For example, the features a car, such as the load the car is carrying, the medical history of the manager traveling in the car, and the working mechanism of the car engine are the vital feature of the car.

![Figure 3: Features of a car in terms of driver view point](image)

In real world problem solving, the abstraction specifies necessary and sufficient descriptions rather than implementation details. It results in separation of interface and implementation.

(ii) **Encapsulation**: From the user’s point of view, a number of features are packaged in a capsule to form an entity. The entity provides several services on the surface by linking up and hiding the implementation details. The term encapsulation is used to describe the keeping off of the details used in implementation.

Information hiding is about keeping away the implementation process of a system from the user. Data hiding is the private usage of data by few individuals. The essence of this encapsulation is further explained with the user knowing how to use the system without knowing how it was implemented. This implies that the hiding of the driving mechanism of any system from the user of the system refers to information hiding. Whatever interface created for human interactivity of a system is applicable to every other system that the user does not know how the system is built. The form created that accepts input from the user to the data base is a typical example of an interface. The point of interactivity is visible to the user but not the implementation process and its function is not affected by the change in the implementation by a different type of steering mechanism such as power steering. The user’s interface is an integral part of the system but not the implementation process and this implementation process will not change the human interface. Therefore, the process, or mechanism, by which you combine code and the data it manipulates into a single unit, is commonly referred to as encapsulation. With Encapsulation in place, there is always a measure of security and data protection from the user of the system.

(iii) **Inheritance**: Inheritance allows the extension and reuse of existing code, without having to repeat or rewrite the code from scratch. Inheritance involves the transfer of the parents attribute to the children. Creating new classes, also called derived classes, from existing classes (base classes). This creation of new classes from the old classes enables the existence of a hierarchy of classes that transfer attribute to other smaller classes called the subclass in the concept of the real world.

Inheritance is often times applicable when there is need to transfer attribute from one class to another. Extension uses inheritance to transfer attribute from one class, the existing classes to new ones by adding new features. Specialization is another term used to refine the behavior of a general class through inheritance.

(iv) **Polymorphism**: This allows an object to be processed differently by data types and/or data classes. In other words, it allows different objects to respond in different way when faced with same challenge. Again, this allows a single name or operator given to an object in a class to be associated with different operations in that class or other classes. This action depends on the type of data passed, and gives the ability to redefine a method within a derived class.

(v) **Concurrency**: This is represented in OOP such as Java through threading, synchronization, and scheduling. Using concurrency allows additional complexity to the development of applications, allowing more flexibility in software applications.

(vi) **Genericity**: This is a technique for defining software components that have more than one interpretation depending on the data type of parameters. Thus, it helps to solve complex problem by keeping off unnecessary details from the user of data items without specifying their exact type. These unknown (generic) data types are resolved at the time of their usage (e.g., through a
(vii) Persistence: This is the concept by which an object (a set of data) outlives the life of the program, existing between executions.

(viii) Events handling: An event can be considered a kind of interrupt: they interrupt your program and allow your program to respond appropriately. In a conventional, non object-oriented language, processing proceeds literally through the code: code is executed in a 'top-down' manner. The flow of code in a conventional language can only be interrupted by loops, functions, or iterative conditional statements. In an object-oriented language such as Java, events interrupt the normal flow of program execution. Objects are connected to one another and can pass information and control in chains from themselves to another object, and so on.

(ix) Delegation: This is an alternative to class inheritance. Delegation is about the connection amongst objects which allows object to function like the inheritance. In delegation, two objects are involved in handling a request: methods can be delegated by one object to another, but the receiver stays bound to the object doing the delegating, rather than the object being delegated to. This is comparable to the relationship of a child and parent where the child classes could send requests to parent classes.

3.1 Technological Advancement in Object Oriented Programming Paradigm for Problem Solving

Discussed briefly are some technological advancement in object oriented programming paradigm for problem solving during software development.

a. Design Pattern Detection
Design patterns play a key role in software development process as it was created to document reusable architectural proposals for producing high-quality designs, help document. [20] Therefore, design pattern detection is described as advancement in OOP paradigm introduced to improve software quality and accelerate software development. It is an important concept used in the field of reverse engineering used in identifying design patterns used in the implementation of a software system. By detecting design patterns from reusable software, it is easier to recognize and verify those reusable parts in software based system or figure out in the form of pre-developed components, or build them as reusable product [21].

b. Object Cloning
The object cloning is a way to create exact copy of an object by transferring all its attribute to that same object (the cloned object). The method within an Object class is used to clone an object. In software development, object cloning technique saves the extra processing task for creating the exact copy of an object.

c. Class Co-evolution
It can be described as an implicit inter-dependence among classes which, when understood, can be valuable in guiding subsequent evolution of the system in question. This co-evolution relation can be used as the basis for advice on maintenance activities during software development [22]. [23] With co-evolution, patterns are formed, in addition to the members of the co-evolving clusters, report the nature of the interdependency between the co-evolving classes.

d. Query-able Source Code
It is advancement in object oriented software development processes where source code encode a query able and modifiable model of the applications and utilizable by IDEs in these aspects [24].

e. Interfaces
This is the input point between the user and the system which serves as a class blueprint. It also works with the object as defined(data types) which determines exactly what can be done with the class, without getting know the details of the implementation-specific details. This allows software developers to group classes that share some functionality but do not necessarily share a parent class.

f. Meta-model for generating design alternatives
A meta-model specifies the structure, the semantics, and the constraints for a family of models in a certain domain [25]. At times in most software development activity, there are extensively used for prediction, sensitivity analysis, pattern recognition, and design optimization [26]. Therefore, in software development, meta-model serves as an alternative to most common code-based development techniques.

g. Namespaces
Namespaces help you to more effectively manage your code base by compartmentalizing various libraries and classes according to context [27].

h. Introspection and Reflection
Introspection refers to making a program to look into the object's attribute, like its name, parent class (if
any), properties, and methods. Reflection is about making a computer program to x-ray the structure of a program as well as the modification of its behavior (specifically the values, meta-data, properties and functions) of a program at runtime.

i. Magic Method
Magic Methods are vital concepts in Object Oriented programming, allowing you to respond to specified circumstances when using a specific object. It allows you to tweak calls to methods or properties of a class and update the object state when particular operation occur.

j. Cohesion, Coupling and Separation of Concerns
Cohesion is the sticking together of responsibilities within a module or in other words its complexity. Coupling captures the complexity between connecting different classes. While, Separation of Concerns (SoC) is the idea that a software system must be split into parts that do not overlap in functionality.

k. Auto-Active Functional Verification
Auto-active Functional verifiers are OOP concepts that provide an intermediate level of computerization(automation) between fully automatic systems and the interactive users supply code with annotations as input while benefiting from a high level of automation in the back-end [28].

These technological advancement in OOP helps to translate our thoughts to a program. It makes possible for a problem to solved exactly the same way human being could solve it in the a real world. It is possible to construct large reusable components using these object-oriented technologies.

4.1 Conclusion
Therefore, the increasing demand of software in the market enterprise today due to numerous problem facing the human being and organization is continuously creating gaps for development of new technologies in OOP. Based on customer and user request, software developers are rapidly finding solution to solving problems. In effort to solve these regenerating issues, new advanced technologies in the field of OOP are continuously being created and additional technologies are emerging. Software developers are faced with code usability, refractory, maintainability, enhancement of software features, developing from the scratch and many more research are emerging in order to find considerable approach to this clarion call in the software engineering domain. Hence, this necessitated this paper focused on technological advancement in object oriented programming paradigm for problem solving during software development. The knowledge of OOP paradigm when utilize will increase problem solving abilities of OOP by matching the problem with real world object then translated to code.

References


[14] Yan W. T., (2016), Software Development Lifecycle Model, The Hong Kong University of Science and Technology


