Evolution of Programming Paradigm and novel approach towards development

[1]Apurv Agrawal,[2]Sonali D. Mali, [3] Snehal Chaudhary,[4]Akash Yadav, [5]Priyanka Paygude

[1,4]U.G. Student, Department of Information Technology, Bharati Vidyapeeth (Deemed To Be University),College of Engineering, Pune, Maharashtra, India,

[2,3,5]Professors, Department of Information Technology, Bharati Vidyapeeth (Deemed To Be University),College of Engineering, Pune, Maharashtra, India

Abstract : The Procedure-Oriented programming paradigm (POP) marks the beginning of programming era. Later on, the Object-Oriented paradigm (OOP) succeeded it and provided a new viewpoint. Now Aspect-Oriented paradigm (AOP) and Context-Oriented paradigm (COP) are again revolutionising the way we see this world. This paper discusses about the above-mentioned programming paradigms analysed from different perspectives. It provides an unbiased comparison of these paradigms in terms of their evolution, need and strengths. Considering a real-world example for this evaluation and traverse through these paradigms identifying their strengths and necessity as the development journey proceeds. This paper also provides information regarding various tools and modules available for the programmers that provide proper implementation of core concepts of the programming paradigms discussed.

IndexTerms **– (Context|Object|Aspect|Procedure) Oriented Programming**

I. INTRODUCTION

In the early days of programming, all the systems intended to provide functionality and that was the only motive kept in the mind while developing a system. Providing computational capability was the only requirement of systems developed in the inception phase of the programming era. Procedure-Oriented programming paradigm (POP) fulfilled this, which followed the strategy of developing functionalities to achieve business requirements. A paradigm is the way of perceiving the world and the way we interpret it while developing a system. It is not about how we program but how we choose to design a system so that it is logically close to the real world scenario and reflects the real world itself in the system.

The next step was to make the system easier to develop with simplified object management. Thus came the Object-Oriented programming paradigm (OOP), which bundled the data into one unit and provided functionalities over them. This supported a much more understandable design of the system. The real world problems can now be viewed and designed with ease and thus provided better understanding towards the designing of the system. The data and functions related to the data were grouped together into one single entity called as class. Objects of these classes are created as instance of the class and achieve the business functionality.

Later in time the importance of differentiating the aspect based on system requirements was realised which lead to introduction of Aspect-Oriented programming paradigm (AOP), which follows the concept of modularising and associating different concerns and aspects related to a system. Aspects can be understood as functionality that is required by the system to achieve optimism in terms of performance and computational correctness. These aspects may be required from the end user perspective or can be necessary to be considered to maintain the integrity of the system.

Now a days the importance of a context aware system and the scope and extent it is capable of handling is realised. A context aware system is capable of dynamically changing the behaviour of the system to achieve harmony with the context it is working in. The term context here can be understood as the state of the environment in which the system is used. Having a system responsive to its environment provides and results in better performance. Thus came the Context-Oriented Programming Paradigm (COP).

In this paper, the evolution of all the paradigms introduced above are discussed in detail by observing the need and strength of these programming paradigms by developing a Banking system. To have a better understanding while developing the system the paper briefly talks about the development from a programmer's perspective. The terminologies of C and Java are considered while discussing from the programmer's perspective also.

II. PROCEDURE ORIENTED PROGRAMMING PARADIGM

Procedure-Oriented Programming Paradigm deals with a procedural way of implementing the functionalities of a system. The Procedure in POP is defined as the specific series well-structured steps and function defined in its programming context. This can be understood as the programming style that views a system from the agenda of identifying and separating the functionalities that are required to be developed.

In POP, the functionalities are implemented with the help of what is called as procedures or simply functions. These procedures contain the well-structured steps defined in the definition. A separate procedure is responsible for providing the way of invocation of other procedures. This means that a procedure may also queue or schedule another procedure and thus group of procedures form the steps to be followed to achieve the required functionality.

The procedures defined above are usually developed in separate modules that aim to provide separate functionality, which promotes de-coupling. Before the POP was introduced, the entire coding consisted of machine level coding which dealt with memory

and registers with a sequence of statements to be executed. With the introduction of POP, the coding became much easier and it provided modularity. Having modularity in the system code enabled parallel development of exclusive modules and thus could be tested separately (unit testing). Code readability was increased which boosted the development phase.

Banking System

Considering the Banking system from the Procedure-Oriented Programming Paradigm perspective, functionalities that this system must provide needs to be identified. From that perspective identifying the data structure to be used, deposit/withdrawal/transfer procedure, open/close account procedure, interest calculation procedure, low balance procedure etc. are the important procedures that we identify.

Thus, these will be the procedures should be considered while developing, each in separate modules. Considering the C programming for POP we can conclude that the above identified procedures would be coded as functions. Array of Structures would be required to carry the data required to process the Banking related activities. Then each of the different module can be tested separately and will be tested again after integrating. Thus, a simple menu driven system can be developed using this approach, which will serve the above mentioned requirements.

However, there are some drawbacks in the system that need to be addressed. One of the main drawback is that anyone who has access to the system i.e. CRUD operations can be easily performed by anyone who gains the access to the system which becomes a major concern while considering the sensitivity of the system. Therefore, in section 2 the OOP is discussed, which will help us in resolving this issue.

III. OBJECT ORIENTED PROGRAMMING PARADIGM

OOP is completely different from the POP. The way we perceive the real world in OOP is very different from how we perceived it in POP. While in POP, the focus is on the functionality and procedural part of the system, in OOP we define the real world in terms of classes and associate the functionalities to these classes. This provides better understanding of the system and what it represents or corresponds to in the real world. The object in OOP is defined as *the instance of a class capable of holding the data to be processed along with the actions that can be performed on the object of that class.*

From the above definition it can deduced that the real world objects are considered as classes in OOP. This class contains attributes that corresponds to the property of the object in the real world. The functions associated or defined in a class reflect the operations that can be performed on the real world object which ultimately result in the change of the state of that object. Thus, to summarize the above discussed it can be said that real world objects (things) are reflected as classes in the OOP.

The term Object in the OOP is symbolizing the object of the real world and not the one that is defined in the system. The term object while coding symbolizes the instance of that class, i.e. an object in programming domain represent a specific object in the real world and not the generally defined object. For eg. In reference of our Banking system, an account is a real world object (thing) which will become a class when we try to implement it in the system. However, an account with specific details such as account number, amount, customer name, type of account etc. are the properties which are represented by the attributes in the programming domain; also when we declare a new object, we will assign values to these attributes as well.

Banking System

Considering the Banking System from OOP perspective, it should completely be revised the way we it was defined in POP. From OOP perspective real world objects will become the classes of our system such as account, customer, bank, administrator, accountant etc. will be our classes. Each of these classes will contain attributes, which will represent the properties of the real world object such as account number, account type, balance etc. for account object (real world). The functions associated with these classes will be the actions we perform on these real world objects in real world such as deposit/transfer/withdrawal, open/close, interest calculation etc. for account. It should be noted that all the functions we consider for the system are actually capable changing the values and properties of a real world object, which will be reflected in the system.

This system can also be divided into different modules which will promote the parallel development as well as unit testing. This also helps in achieving de-coupling of exclusive modules which will improve the quality of the system. Since these modules are separated thus cohesiveness will also come into picture so that different modules can work in collaboration. Such modular architecture brings a long list of benefits.

One observation to be made is to understand that the pillars of OOP are Abstraction, Polymorphism, Inheritance and Containment helps us handle the concern of security which was discussed in the previous section which is very important to maintain the integrity of the system. Yet, one can identify that this system is still simply serving the functionality required. It is not contributing anything additional towards improving the design of the application.

IV. ASPECT ORIENTED PROGRAMMING PARADIGM

AOP is a paradigm that can be applied on top of other programming paradigms. It is a concept that talks about different functionalities that are required for a system to work properly without affecting or compromising the integrity of the system. Aspect can be understood as the concern of different stakeholders having different priorities that are spread across various modules of the system. These concerns are often termed as cross cutting concerns as these are not the core functionalities but the functionalities that are required to maintain the integrity and consistency of the system.

Above discussion clearly indicates a well-structured and modular system with a much better software metrics. As the system is modular it signifies that it would be easier to handle and accommodate changes suggested in the future development of the system. This also promotes de-coupling of functional and service level functionalities which in turn is a proper way to develop a system.

To find the essence of AOP, the terminology related is necessary to understand.

- *Aspect* is the cross cutting concern which serves the peripheral non-functional requirements.
- *Advice* are the statements that are coded to serve the concerned aspect.
- *Join-Point* is the actual instance in the executing program where the advice is applied.
- *Point-Cut* tells about the join point to which the advice must be applied to.
- *Target* is the application object on which the advice is implemented upon.
- *Proxy* is the object provided by the framework after implementing the advice.
- *Weaving* is the process of integrating these advice with the help of point-cuts on the specific join-point returning the target application object.

Banking System

The banking system developed in the previous section was developed from the perspective of OOP, which did provide the functionality that were required. However, from the AOP perspective one needs to identify the aspects from the existing system and must separate and develop them in a separate module. The prominent aspects for the banking system would be logging, transactional consistency, security, authentication, accessibility etc. Though these functionalities can be provided by the OOP but these will be scattered over all the modules of the entire system which couples this code to the main code. Having these aspects separated in the form of advices implemented with the help of join-point and point-cut will be more manageable.

Still there is a lack of sense of dynamicity in the banking system. Though the aspects have been separated from the executing code and have weaved it to the join-points, the code weaved cannot be changed at run time and the actual code would be required to tamper with to accommodate changes. The system would behave only in the specified way only. It is simply insensitive to the changes in the environment it is being executed. This problem of dynamicity is addressed in the next section.

The methodology section outline the plan and method that how the study is conducted. This includes Universe of the study, sample of the study,Data and Sources of Data, study's variables and analytical framework. The detailsare as follows;

V. CONTEXT ORIENTED PROGRAMMING PARADIGM

COP is the newly introduced programming paradigm, which provides dynamicity to the applications in a way that systems implementing the concept of COP are context aware and are able to respond according to the context in which they are deployed. It focuses on separating the cross-cutting concerns that affect the context of the system. The term context here refers to the environment in which the system is working. The system environment can change and may require the system to respond accordingly.

Following are the basic terminologies of COP:

- *Context* can be defined as any form of information that is computationally available and is indulged in behavioural variations.
- *Scoping* is a common term in COP, which provides information regarding the space in which a given context will be activated.
- *Layers* are grouped contexts that are behaviour specific variations and can be used on top of any programming model as they are first class entities.

To understand what a context is and how context aware models work, consider the following example of mobile phone. Network connectivity can be identified as an Aspect for a mobile application. Considering this mobile application to be contextually aware, as soon as the battery percentage crosses the critical threshold, network connectivity of various applications will now be updated based upon the current environment (context) conditions. This way we identify low battery as a context in which mobile application must perform and system responds to this context by updating the network requirements of individual application.

Banking System

From the AOP perspective, different aspects were identified that were scattered over the system and were then grouped together to form one single aspect. In the same way different contexts are required to be identified to make the system dynamic and respond to context variations. Contexts can be power outage, inconsistent transactions, development of sink, fraud activities etc. These are the contexts that the system must be able to handle. For example, during power outage, the transactions must be rolled back when the system reboots. Recovery procedure would start. During the recovery period the fund transfer transactions should be aborted. In case of detection of fraudulent activities, all the related accounts must be freezed.

So in this way it is observed that system developed now is capable of handling the dynamic changes happening in real time and it also changes behaviour according to the situation providing dynamicity to the system.

VI. THE BIG PICTURE

The central idea of this section would be to discuss the overall journey of evolution of programming paradigms as a whole. In section 1, the basic introduction to all the paradigms was discussed. The section 2 introduced the POP, which had the idea of developing the system considering the functionalities it needs to serve. The system was developed only to process the inputs and was stateless. The section 3 introduced the OOP, which completely revolutionised the way systems were developed. The systems now developed were more secure, modular and state-full as well. The data was now handled more appropriately. The section 4 & 5 introduced ways to develop the systems more properly so that it becomes easier to manage, handle and accommodate changes even at a later stage in development.

One main point to note here would be to realise that systems can be developed using POP or OOP but AOP and COP can be applied individually as well as in collaboration too. AOP and COP are simply ideologies that help to develop a system more accurately. *These ideologies can be applied to both POP and OOP*. The core concept of AOP is to identify the aspects and provide advice to them, which can be applied to any paradigm i.e. POP and OOP. Same is the case with COP, which argues that systems should be aware of the environment in which they are working and should respond to those changes accordingly by computing the variables which represent the context and thus these computations can be computed in POP and OOP as well.

Though more modules of AOP and COP are available in the market for OOP languages than compared to POP, still they are ideologies and can be applied to any of the programming paradigm. The reason number of modules available for OOP is higher because of the success of OOP over POP. Yet, there is no such recommendation/suggestion/preference about choosing between POP and OOP; one can choose any paradigm to develop any system based upon the features that the system demands.

VII. CONCLUSION

The POP is the programming paradigm which focused on the functionality while OOP focused on the data as well. AOP and COP are the ideologies that can be applied on top of the other programming paradigm as their core concepts are abstract. The first two discussed paradigms provide a way to develop any system while the rest two provide upgradation technique to make the system perform better from different aspects and in different contexts.

REFERENCES

[1] Ayed, D., Delanote, D., Berbers, Y. (2007). *Computer Science*, V. 4635/2007 of *Lecture Notes in Computer Science*, Springer, Berlin / Heidelberg, chapter MDD Approach for the Development of Context-Aware Applications, p. 15-28.

[2] Chantal Taconet, Zakia Kazi-Aoul. Building Context-Awareness Models for Mobile Applications.

[3] Malte Appeltauer, Robert Hirschfeld, Michael Haupt, Hidehiko Masuhara. ContextJ: Context-oriented Programming with Java.

[4]Jongmyung Choi, Rosa I. Arriaga, Hyun-Joo Moon, and Eun-Ser Lee. A Context-Driven Development Methodology for Context-Aware Systems.

[5] Malte Appeltauer, Robert Hirschfeld, Michael Haupt, Jens Lincke, Michael Perscheid. A Comparison of Context-oriented Programming Languages.