# Comparative Study on Object Oriented Approach and Aspect Oriented Approach for the Software Development

**Dishek J. Mankad**
B.R.Patel Institute of Computer Applications, Gujarat Technological University, Gujarat, India

**Abstract**
This contents present a research where a new programming technique is applied to an established branch of the software development. The main purpose of this content was to check whether or not how easily aspect – oriented programming could be used in software development. The challenge in today's technological environment is to keep evolving the older systems so that they are truly compatible with the real world technological environment. The most common approach was to migrate your code into the object oriented code. However there are many various paradigms that a software development might adopt. Aspect oriented technology is today's most emerging concept for the software development that is receiving considerable attention from research and practitioner communities alike. The approach reverse engineering activities to abstract an object – oriented model from any code. The methodology was to migrate it incrementally by decomposing existing system into the notable sets of components, and each of this potentially implements an object. In this paper we are focus on the work done in evolving system using object oriented approach, then we analyze the impact of object oriented technology and aspect oriented technology on system development and the environment that is required to implement the two paradigm.

**Keywords**
Object – Oriented Programming, Aspect – Oriented Programming, Software Development, Paradigm

## I. Introduction

Software Engineering is defined as the establishment and use of sound engineering principles in order to obtain software which is reliable and works efficiently on real time machines. Presently software is like clay, it is soft for a while, but then it hardens and becomes brittle and can't be changed easily for use in new contexts. But software should be more like gold, malleable for life and able to adapt and change its shape in response to new needs. Also as the size of software applications expand at an exponential rate, their complexities are also growing. There are a variety of ways for an application's complexity getting increased. One feature of this complexity is the repetition of functionality like as security, memory management, resource sharing, and error, memory crash and failure handling throughout an application. This complexity is realized when programmers change an often-repeated feature for an updated or latest version of particular application. It is very hard and quite complex for programmers to find each and every instance of such a feature in millions of lines of code. Failing to find and change every occurrence of this feature promotes bugs in the application.

To overcome from issue like this, software developers and researchers are engineered some solid methodologies which is based on a new programming element called "aspect". An aspect is a piece of code that describes a recurring property of a program. Application can, of course, have multiple aspects. Aspects provide crosscutting modularity. That is, programmers can develop aspects to create software modules for issues that cut across various parts of an application. Thus programmers could thus think of, write, view, edit and otherwise address these issues as a unit, implementing changes or upgrades across all applicable code sections, rather than by having to find and modify each applicable piece of code. In this research paper we are discussing about Aspect Oriented Approach and Object Oriented Approach then comparison of both the approach and after last conclusion which I found from my research.

## II. Object – Oriented Programming Approach

OOP is a design philosophy. OOP stand for object oriented programming. Object oriented programming uses a different set of programming languages than old procedural programming languages like as C, Pascal etc. Everything in OOP is grouped as self sustainable objects. Therefore, you gain re-usability by means of four main object p oriented programming concepts. The core of the pure object-oriented programming is to develop object, in code, that has certain properties and methods. While designing any Object-Oriented programming modules, we try to see whole world in the form of objects. If we take example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake and soon. So, basically in my point of view Object – Oriented Programming is a style programming that focuses on using objects to design and build applications. First we think object as a model of the concepts, processes, or things in the real world that are meaningful to your applications. Late take example of it, in project management applications; you would have a status object, a cost object, and a client object among others. These object would work together to provide the functionally that you want your project management application to have. Basically Object – oriented programming is used to develop many applications – simple and it may be complex applications, business applications , and games, mobile and desktop applications. So, the developers choose to program in the object – oriented paradigm because the proper use of objects makes it easier to build maintain, and upgrade and applications. Also developing with objects that have been tested and increases the reliability of the particular applications.

### A. Benefits of O-O-P

Object Oriented Programming has the listed benefits over conventional approaches:

1. OOP gives a clear modular structure for programs which makes it good for defining abstract data types where program implementation details are hidden and the unit has a clearly defined interface.
2. OOP makes it easy to maintain and update existing code as a newly created objects can be created with small differences to existing ones.
3. IT provides a better framework for code libraries where supplied software components can be easily adapted and modified by the particular programmer. This is specifically useful for This is specifically useful for graphically user interfaces.

### B. Drawback of O-O-P

Like anything else OOP has also dark side. The biggest drawback of OOP in terms of general simulation programming is probably that the real world refuses to divide up into neat classes and subclasses. At first glance, it's easy to look all around and start dividing things up into classes:

1. With OOP, classes tend to be overly generalized.
2. The relations among classes become artificial at times.
3. The OOP programs' design is tricky.
4. All one needs to do proper planning and design for OOP programming.

In some OOP Language, you may have to perform this extra step. If you force the language into the OOP Concept, you lose some of the features of useful language like the "functional languages". One another drawback is that one programmer's concept of what constitutes an abstract object might not match to the vision of another programmer. The objects frequently extensive documentation.

### C. The Future of OOP

The future scope of OOP probably lies with the languages like ruby and lua, in which the object concept in built into the language and is not always explicitly controlled by the programmer. For example Ruby, it treats everything like an object, including strings, numbers, your program and the contents of the directory you are currently in. You almost never need to declare anything to be an object. You are free to use the any language any way you want: as an OOP, as a functional language or even in ways those modalities.

## III. Aspect Oriented Programming Approach

When Object – Oriented Programming entered the mainstream of software development; it had a dramatic effect on how software was developed. So, the developers could visualize systems as groups of entities and the interaction between those entities, which allowed them to tackle larger, and more complicated systems and develop them in less time than ever before. The only problem with OO Programming is that it is essentially static, and a change in requirements can have a profound impact on development timelines. An aspect – Oriented Programming (AOP) complements OO Programming by allowing the developer to dynamically modify the static OO model to create a system that can grow to meet new requirements. Just as objects in the real world can change their states during their lifecycles, an application can adopt new characteristics as it develops. Let take an example: many of you have developed simple web applications that use servlets as the entry point, where a servlet accepts the values of HTML form, binds them to an object, passes them into the application to be processed and them returns a response to the user. The first cut of the servlet may be very simple, with only the minimum amount of code required to fulfill the use case being modeled. The code however, often inflates to three to four times its original size by the time secondary requirements such as exception handling, security, and logging have been implemented. I use the term "secondary requirements" because a servlet should not need to know about the logging of security mechanism s being used; its fundamental function is accept input and then after process it. AOP allows us to dynamically modify our static model to include the code require to fulfill the secondary requirements without having to modify the original static model (in fact, for that we don't even need to have the original code). Better still, we can often keep this additional code in a single specific location rather than having to scatter it across the existing model, as we would have to if we were using OO on its own.

### A. Terminology

Before we delve too deeply into AOP, let us introduce some standard terminology to help us understand the concepts of particular approach.

#### 1. Cross – Cutting Concerns

Even though most classes in an OO model will perform a single, specific function, they often share common, secondary requirements with other classes. For example, we may want to add logging to classes within the data – access layer and also classes in the UI layer whenever a thread enters or exits a method. Even though the basic fundamental functionality of each class is very different, the code needed to perform the secondary functionality is often identical.

#### 2. Advice

This is the additional code that you want to apply to your existing model. In our example, this is the logging code that we want to apply whenever the thread enters or exits a method.

#### 3. Point – Cut

This is the term given to the point of execution in the application at which cross – cutting concern needs to be applied. A point – cut is reached when the thread enters a method, and another point- cut is reached when the thread exits the particular method.

#### 4. Aspect

The combination of the point-cut and the advice is termed an aspect. In the example below, we add a logging aspect to our application by defining a point-cut and giving the correct advice.

### B. Why AOP?

AOP extends the traditional object oriented programming (OOP) model to improve code reuse across different object hierarchies. The basic concept in AOP is an aspect, which is commonly behavior that's typically scattered across methods, classes object hierarchies, or even entire object models. For example, metrics is one more common aspect. To generate useful logs from your application, you have to sprinkle proper informative messages throughout your code. However, metrics is something that your class or object model really shouldn't be relate about. After all metrics is irrelevant to your actual application; it doesn't represent a customer or an account, and it doesn't realize a proper business rule. It's an orthogonal behavior that requires duplicated code in traditional OOP systems. In This Approach AOP , a feature like metrics is called a crosscutting concern, as it's behavior that "cuts" across multiple points in your object models, yet is quite different. So, as a development methodology, AOP recommends that you abstract and encapsulate crosscutting concerns into aspects. In the context of enterprise middleware, container managed services are crosscutting concerns. Once you deployed each J2EE component like EJB or a SERVLET it will automatically gets services, such as logging, security and transaction, from the container. Those services are orthogonal to the core business logic. The applications developers could reuse those services without writing single lines of any code.

### C. Advantages of AOP

- It allows developers to dynamically modify the static OO model to create a system that can grow to meet new requirements
- An applications can adopt new characteristics as it develops just as objects in the real world can change their states during their lifecycle.
- It allowed code layering to made your code easier and reduce complexity
- It also allow cross cutting concern to divide your code in proper module and provide security.
- Code adjusting allows programmer to reuse of code

### D. Drawback of AOP

1. A factory pattern is obviously something that AOP can do better, as can also do it well, but the observer pattern is simpler when using AOP, as is the strategy pattern.
2. IT will be harder to unit test, especially if you do the weaving at runtime.
3. If we having at runtime then you also take a performance hit.
4. Having a good way to model what is going on when mixing AOP with classes is a problem, as UML. I personally don't think is a good model at the point.
5. Unless you are using Eclipse or mobile application development tools do have problems, but Eclipse and AJDT AOP is much easier.
6. We still use JUNIT and NUNIT for example, and so have to modify our code to allow unit tests to run when suing privileged mode AOP could do better unit tests by testing private methods also, and we don't have to change our programs just to make them work with unit – testing. This is one another example of not really understanding how AOP can be helpful, we are still chained in many ways to the past with unit-test frameworks and current design pattern implementations and don't see how AOP could help us do better coding.

### E. Future of AOP

Regarding the specific criticism, it seems uninformed or misunderstood. The entire point of AOP is to reduce maintenance work by reducing code duplication in the area of cross cutting concern. However, it is true that it can make maintenance more hard and difficult by making it harder to understand the code – with AOP, when you look at a particular piece of code, there's always the chance that an aspect defined somewhere else completely change its behavior. Personally, I'm not sure the benefits of AOP outweigh this problem. The only way to I can see to solve it would be through support in IDE's but making you even more dependent on an particular IDE isn't such a great thing either. So, as per my point of view AOP is a relatively a very immerging concept, and large companies are generally way towards new concepts. So, they fear getting stuck with a lot of AOP code and unable to find people who can maintain it. From the EJB perspective, I think the future of declarative technologies will turn in the era of Aspect – Oriented direction. With AOP, you can plug and play any service you wanted to into a container. For example if you don't like your container vendors persistence service, you can get someone else's and use that. Another point for AOP is that the aspect generated code is fabric into the container code. So you or your vendors write the aspects and at runtime you have a single optimized unit of service and "container" . It's not like there's some container module intercepting calls then delegating them off – at runtime this overhead is removed these subsystems can be directly intermingled.

## IV. AOP Vs. OOP the Comparison

OOP has become the major technology in recent years, which almost takes the place of structure – oriented programming. However, a kind of new technology AOP will rise soon, and its influence will go even farther than OOP. Once I have been developing software system using object – oriented programming techniques for many years. So, when I read that Aspect – Oriented Programming addresses many problems that traditional OOP doesn't solve completely or directly, I wanted to better understand its benefits in real world application development. I thought comparing both the techniques or programming approaches would provide some practical insight. Here are some critical comparison of OOP with AOP:

- AOP doesn't replace OOP. AOP extends OOP. The ideas and practices of OOP stay relevant .Having a good object design will probably make it easier to extend it with aspects.
- The logic for each concern is now in one place, as opposed to being scattered all over the code base.
- Classes are cleaner since they only contain code for their primary concern and secondary concerns have been moved to aspect

### A. AOP

1. Modularization is based o Processes / Functionalities.
2. Reusability of code is at a great extent.
3. Properly separates business logic into reusable code.
4. Addresses cross cutting concerns
5. Code creation is clear.

### B. OOP

1. Code modularization is into real world objects.
2. Implementation is forced by the developer.
3. Functionality is within a class.
4. Modularity is based on real world objects.

Aspect – Oriented Programming (AOP) complements Object – Oriented Programming (OOP) by providing another way of thinking about program structure. In addition to classes, AOP gives you aspects. Aspects used to create modularization of concerns such as transaction management that cut across multiple types and objects.

## V. Conclusion

Analyzing the facts that had been covered in the earlier sections, it can be concluded that AOP does not replace OOP in the maintenance of old legacy system but adds certain decomposition features that address the so – called cross cutting concern. OOP and AOP are working at different levels of abstraction, OOP works at object level while AOP works at code level. One more constraint is that AOP and software architecture have evolved separately as discipline. Therefore integration of aspects into software architecture is a quite complex job. So, the idea and practices of OOP stay relevant. We have discussed object orientation in the light of both component based development and service oriented architecture. So, the conceptually, all the approaches define different characteristics. However, aspect – oriented, on the other hand, can be seen as a complementary paradigm affecting the software system on several levels. Having a good objects. Although this should always be taken into consideration that the

old OOP systems should not necessarily include AOP, as it may result in unnecessary code complexity and the programmers might have to face the anti – pattern problem. Therefore AOP should not be seen as a replacement of OOP, but as an approach that makes your code more clean, loosely – coupled and focused on the business logic.

## References


[1] G. McGraw, G. Morrisett,"Attacking Malicious Code", AReport to the Infosec Research Council. IEEE Software, 17(5), pp. 33–41, 2000.

[2] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C.Lopes, J.M.Loingtier, J. Irwin,"Aspect oriented programming", In Mehmet Aks¸ it and Satoshi Matsuoka, editors, proceedings European Conference on ObjectOriented Programming,volume 1241, pp. 220–242. Springer-Verlag,Berlin, Heidelberg, and NewYork, 1997.

[3] Ramnivas Laddad,"Enterprise AOP with Spring Applications: AspectJ in Action", 2 Edition, Manning Publications, 2010.

[4] Ramnivas Laddad,"AspectJ in Action: Practical Aspect- nd Oriented Programming", by Manning Publications, 2003.

[5] G. Kiczales, E. Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, William Griswold,"An Overview of AspectJ", by Budapest, 2001. Springer Verlag.

[6] Security Functionality Requirements, National Institute of Standards and Technology, 1992.

[7] Technical Security Standard for Information Technology (TSSIT), Government of Canada, 1997.

[8] Information Technology Security Evaluation Criteria (ITSEC), Department of Trade and Industry, London, 1991.

[9] S.A.M Rizvi, Zeba Khanam,"A Comparative Study of using Object Oriented Approach and Aspect Oriented Approach for the Evolution of legacy system", International Journal of Computer Applications, Vol. 1, No. 7

[10] Jaakko Kuusela, Harri Tuominen,"Aspect-oriented approach to operating system development empirical study", Journal of Communication and Computer, USA.

[11] Kotrappa Sirbi, Prakash Jayanth Kulkarni,"Stronger Enforcement of Security Using AOP & Spring AOP", Journal of Computing, Vol. 2, Issue 6, June 2010.

[12] [Online] Available: http://www.devx.com/Java/Article/28422

[13] [Online] Available: http://en.wikipedia.org/wiki/Aspect-oriented_programming

[14] [Online] Available: http://c2.com/cgi/wiki? Aspect Oriented Programming



Dishek Mankad received Master Degree in Computer Application from N.S.V. K.M.S MCA College Visnagar H.N.G.U. Patan University. Gujarat from 2008 to 2011. His area of research is Software Engineering. He has also publish various papers in the area of Software engineering, data warehouse, and data mining. He is working as an assistant professor at B.R.Patel Institute of Computer Application [MCA Program], Anand, Gujarat, India