ISSN No. 0976-5697

Volume 4, No. 8, May-June 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Concepts and Features of Aspect-Oriented Programming using Aspect.Net Framework: A New Approach to Modularization

Sakshi Mundra[*1], Vaibhav Vyas[2] and Versha Bansal[3]
AIM and ACT, Asst.Professor, AIM and ACT,
Banasthali University Jaipur ,India
saakshi.mundra@gmail.com[*1], vaibhavvyas4u@yahoo.com[2], [3]varshabansal80@gmail.com

Abstract: Aspect Oriented Programming methodology is based on the idea that computer systems can be programmed very efficiently by specifying the various concerns of a system and some description of their relationships separately and then relying the mechanisms in the underlying AOP environment to weave them together into a coherent program automatically. The term Aspect Oriented Programming includes Multidimensional Separation of Concerns, along with Subject Oriented Programming, Adaptive Programming and Composition Filters. Multidimensional Separation of Concerns (MDSOC) permits the encapsulation of various kinds of concerns simultaneously and the integration of separate concerns. The Aspect.NET Framework is implemented as an add-in to visual studio.NET. Now, the user can use Aspect.NET in combination to the integration development environment provided by Visual Studio.NET and all its various features for the development of software applications using AOP methodology.

I. INTRODUCTION

Aspect-oriented programming (AOP) is a programming paradigm which aims at increasing modularity by allowing the separation. In other words, we can say that an aspect is a common feature that's typically scattered across methods, classes, object hierarchies, or even entire object models. AOP introduces concern abstraction [1].

AOP provides separation of crosscutting concerns by introducing a new unit of modularization—an aspect, that crosscuts other modules. With AOP, you implement crosscutting concerns in aspects instead of fusing them in the core modules [1]. For example, the logging concern affects every significant module in the system which is one of the examples of cross-cutting concerns that affects every module with access control requirements.

AOP allows the architect to address future potential requirements without breaking the core system architecture, and to spend less time on crosscutting concerns during the initial design phase, since they can be woven into the system as they are required without compromising the original design [1].This flexibility justifies the use of AOP in real life environments like in a banking system, which is a realization of the following concerns: customer and account management, interest computation, ATM transactions, and many more.

II. RELATED WORK

With the arrival of the concept of object orientation, a single concern can be modularized into a single unit which can be a class, a function or a procedure. However, even with the current methodologies, there is a significant gap between knowing the system goals and implementing them completely [2]. The current methodologies, like OOPs, make initial design and implementation very much complex that the actual implementation of the ultimate goal remains as a dream only. OOP implementation creates a coupling between the core and crosscutting concerns that is undesirable. In OOPs, the core module can be loosely coupled, but it cannot make the coupling of crosscutting concerns to be weakened. This is because a concern is implemented in two parts: the server-side piece and the client-side piece. OOP modularizes the server part quite well in classes and interfaces [2].

However, when the concern is to relate the client part, of a cross-cutting nature, consisting of the requests to the server, it is spread all over of the clients. An example to explain this is the logging module which must be independent of other each and we must be able to replace one with the other so as to reduce complexity. This task is only possible with the introduction of the term aspect.

Using AOP, none of the core module will contain calls to logging services—they don't even need to be aware of the presence of logging in the system. The logging logic now resides inside the logging module and clients no longer contain any code for logging, as it is separately handled in an aspect. With such modularization, any changes to the crosscutting logging requirements affect only the logging aspect, isolating the clients completely. So, with AOP, a minimal coupling is required with each concern which further reduces code tangling and code scattering.

III. USED METHODOLOGIES

This paper is focused mainly on Study of Aspect .NET framework and Aspect oriented Programming in a .Net framework and then developing a web application of Aspect Oriented Programming using Aspect .NET tool and Microsoft Visual C#. AOP is supported by java also but the Microsoft .NET platform is based on the principles of peer-to peer multi-language programming [3]. For any of the .NET languages, a very comfortable toolkit for software development and maintenance is provided- Microsoft .NET framework and Visual Studio.NET. Aspect.net supports two types of crosscutting implementation:

a. ***Static:*** Enables the developer to add variables and methods to existing types.
b. ***Dynamic:*** Enables the developer to define additional implementation to run at well defined points in the program.

IV. PROPOSED WORK

After investigated all the issues affecting design of a website, a new proposal is developed to sort out the tangling of code (Code tangling is caused when such a module is implemented that handles multiple concerns simultaneously). This is done by following AOP concept in Aspect.NET platform which provides an integrated development environment and also provides a variety of code-behind language to choose from [4].

A. Aspect:

An aspect is a modular unit of crosscutting concerns and it contains advice. Advice has the same meaning as in AspectJ, but as there are no language extensions to C#, so the advice must be proper C# code. Aspects are declaratively complete, therefore any variable or method which a piece of advice uses, it must be declared within the advice or as an abstract method or variable.

During the designing of a web application, the aspects for cross-cutting concerns are designed separately. When the application runs, and if there is a call of authentication, then an automatic weaving of these aspects are done with the functional features. A general outlay can be designed as like this:

A logging module is a common module for various kinds of users in a web-application. Every time when a user needs to login, the system must check the various validations and whether the username and passwords entered, are correct or not. So, these non-functional modules span across multiple modules and programmer needs to take care of them every time when he/she is about to perform any changes in the functional modules. So using Aspect, all the non-functional modules can be tested and compiled simultaneously in a single module, called aspect while reducing complexity and redundancy [4].

Using aspect, the logging process can be seen like this: Logging:-Whenever any user login to the system, then a logging aspect log file is created which records users' activities during login to logout session.Null textbox validation-This validation is used by the programmers during login to check whether a specified textbox contains a null value or not. Using aspect, the validation is checked in the aspect and aspect will display an alert message in case, if a null value is found.

(a). Password validation – when any of login function or registration function called, then password validation aspect checks if password field contains appropriate length for password or not. If not, then the alert message will be sent by the password validation aspect.

Whenever any user performs a common function, the same page opens every time. So to reduce the same coding at each page, we can better make a dll that will be called at a point-cut and will be weaved every time at the joint-point.

The whole aspect architecture can be seen like this:

B. Project Architecture:

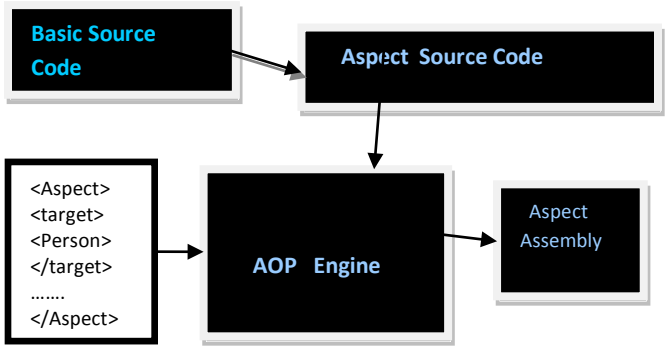

Figure: 1

To understand the actual application and powerful implementations of an aspect, consider this algorithm:

Suppose an enterprise wants to transmit some messages to all the employees working in it. It, by default wants to display 'Hello' in the initial of the message and 'Bye' after the end of the message. So, a part of engaging some employees in this time-taking task, aspect.net can easily submit the task very efficiently like this:

a. Create a class Politeness which will call the methods to display our desired messages.
b. Create two methods to deliver the messages, one which is common for all the employees and the other would be specific for every employee.
c. Create an Aspect at the join point-%modules so that the common terms can be introduced for each employee with the respective messages delivered to them. Use a point-cut at the calling of SayHello() and SayBye() functions. Introduce an advice 'before' with a message 'Hello'.
d. Now introduce an advice 'after' with a message 'Bye' after the deliver function. This advice will be capable of introducing 'Bye' after every displayed message

```
%aspect Politeness
Public class Politeness
{
%modules
Public static SayHello ()
{
System.Console.Writeline("Hello");
}
Public static SayBye ()
{
System.Console.Writeline("Bye");
}
%rules
%before  %call  *
%action public static void sayHelloAction
{
Politeness.sayHello()
}
%after  %call  *
```

```
%action public static void SayByeAction
{
Politeness.SayBye()
}
```

Suppose, here the project name is Aspect1, taken by default. You will see the following aspect definition file template generated by Aspect.NET Framework named Aspect. an. ".an" is the extension for the Aspect.NET AOP meta-language (Aspect.NET.ML) files [5].

```
%aspect Aspect1
using System;
class Aspect1
{
%modules
%rules
}
```

Here 'Aspect1' aspect consists of:- the aspect header %aspect Aspect1 . Aspect.NET converter will convert it to the header of the class named Aspect1.

a) The modules part where you should specify the modules of the aspect, as public static methods;
b) The rules part where the aspect's weaving rules are specified. Each weaving rule consists of the weaving condition (%before %call *, %after %call *) and action.

The condition is used to determine the set of join points in the target app subject to the aspect weaving (before call of each method, or after call of each method, accordingly).

If you like, you can start creating aspect from C# source, without using Aspect.NET.ML at all [6]. In this case, you should choose the "Aspect.NET module" kind of project, and will start from the following C# code template generated by

Aspect.NET:

```
using System;
using System.Collections.Generic;
using System.Text;
using AspectDotNet;
namespace Aspect1
{
[
AspectDescription("MyAspect description")
]
public class MyAspect : Aspect
{

}
```

The user can use either Aspect.NET.ML form, or directly C# custom attributes form for defining aspects [6]. In the Aspect.NET Framework, both kinds of aspect definition projects are supported.The diagram presented below shows the flow of the process in both cases.

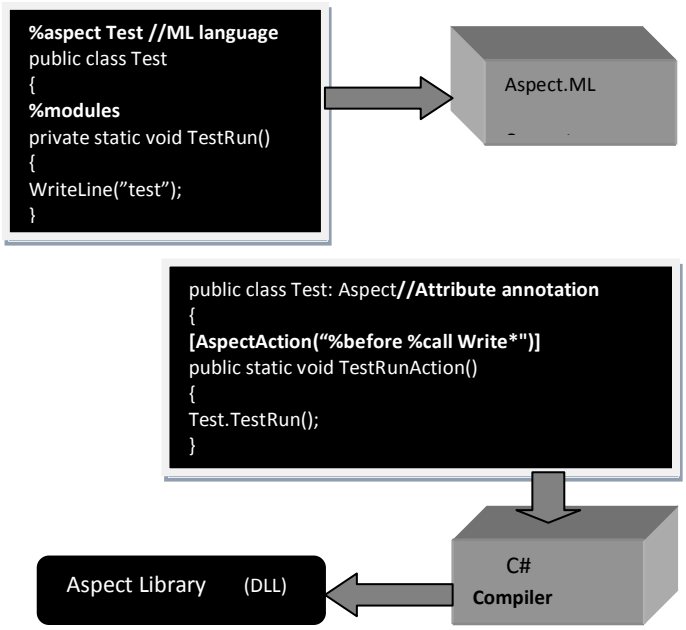

Figure: 2

V. CONCLUSION

It is apparent from this paper that implementation of Designing an application in aspect.net framework is much sorted out in comparison to object-oriented concepts. As it separates out the functional code and non-functional code, so the complexity reduces.

VI. REFERENCES

[1]. Ramnivas Laddad, "AspectJ in action, Practical Aspect-Oriented Programming-2003".

[2]. Aspect-oriented Programming http://www.en.wikipedia.org/wiki/Aspect-oriented programming.

[3]. Howard Kim, AspectC#: An AOSD implementation for C#, Department Of Computer Science Trinity College Dublin

[4]. Vladimir O. Safonov, "Aspect .NET 2.1 User Guide", St.Petersburg University, 2007.

[5]. Deepika et al., Investigating the Web Application of AOP Using Aspect.Net Framework International Journal of Advanced Research in Computer Science and Software Engineering 2 (8), August- 2012, pp. 109-112

[6]. Miguel Katrib Mora, Yamil Hernandez Saa, "Aspect Oriented programming in .Net based on attributes", Computer Science Department. University of Havana, PP-54-70-03/07, 2007.