ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st&22ndMarch, Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

Enhancing the Performance Metrics of Introducing Aspect Oriented Programming into a Program Course

R.Kabilesh, P.C.Dinesh

PG Scholar, Kalasalingam Institute of Technology, Krishnankoil, India.

Assistant professor, Kalasalingam Institute of Technology, Krishnankoil, India.

Abstract— Service Oriented Architecture (SOA) is a new method for building information systems. It discusses about the effects of using aspect-oriented paradigm in addition to object-oriented paradigm. The results show that the use of aspect-oriented programming as a supplement to object-oriented programming enhances the productivity of program code and reduces the Lines of Code and Cohesion between the modules. Web services are distributed autonomous applications that can be discovered and interactively accessed over the internet. The functional components as well as the non-functional components such as security, reliability and quality of services are given utmost importance for web service languages.

Index Terms — Aspect-oriented programming (AOP), programming, programming languages

I.INTRODUCTION

Aspect Oriented Software Development (AOSD) [10] is a relatively new methodology for neatly encapsulating functionalities and thereby increasing the modularity of the software. The software development process is driven by modelling and implementing the interactions of various software concerns, which can be both core and cross-cutting. Core concerns, are the basic functionality of the system; for example, concerns that specify the behaviour of certain application, such as the function of a class, or the method of a class. In contrast, cross-cutting concerns are concerns such as logging, security, concurrency control, and transactionmanagement that cut across many other modules of the software.

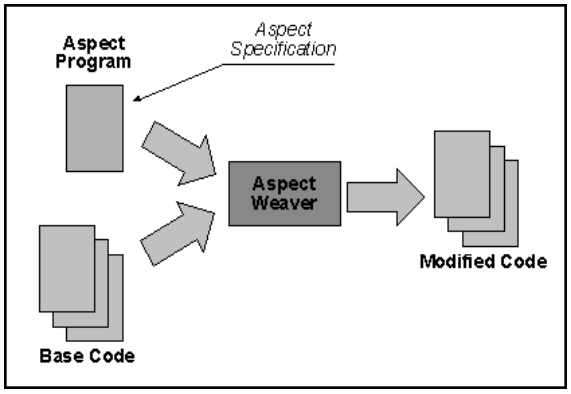
Service Oriented Architecture (SOA) is a software architecture that defines the use of services, to support software user requirements.. Aspect-oriented programming (AOP) has been proposed as a mechanism that enables the modular implementation of crosscutting concerns [1]. It has proven popular [5] because it makes it possible for developers to write modular code for concerns such as synchronization [11, 4], error handling [2], persistence [3,7] and many design patterns Being able to code aspects cleanly is helping developers to think in terms of aspects at earlier stages of the lifecycle [6].

BPEL is a language used for the definition and execution of business processes using Web services. BPEL enables the top-down realization of Service Oriented Architecture (SOA) through composition, of Web services. The interactions are abstract in the sense that the dependence is on port Type definitions, not on port definitions. Define business processes using an XML-based language [8]. Do not define a graphical representation of processes or provide any particular design methodology for processes. Define a set of Web service orchestration concepts that are meant to be used by both the external (abstract) and internal (executable) views of a business process. Such a business process defines the behaviour of a single autonomous entity, typically operating in interaction with other similar peer entities. It is recognized that each usage pattern will require a few specialized extensions, but these extensions are to be kept to a minimum and tested against requirements such as import/export and conformance checking that link the two usage patterns. Provide both

hierarchical and graph-like control regimes, and allow their use to be blended as seamlessly as possible. This should reduce the fragmentation of the process modelling space. Provide data manipulation functions for the simple manipulation of data needed to define process data and control flow.

It deals with the AOP [9] implementation of crosscutting concerns, and modular reasoning in the presence of crosscutting concerns. But it requires an important change in how module interfaces are specified. With AOP interfaces are defined as aspects cut through the primary module structure. So module interface cannot be fully determined without a complete system configuration. But crosscutting concerns inherently global knowledge in order to support modular reasoning.

It deals with the AOP implementation of crosscutting concerns, and modular reasoning in the presence of crosscutting concerns. But it requires an important change in how module interfaces are specified. With AOP interfaces are defined as aspects cut through the primary module structure. So module interface cannot be fully determined without a complete system configuration. But crosscutting concerns inherently global knowledge in order to support modular reasoning. Using AOP, programmers get modular reasoning benefits for crosscutting concerns where without Aspect Oriented Program they do not. Fig 1 describes the structure of AOP and its specification.



. Figure 1 Structure of AOP

Automation in Experimental Test Bed

Aspect oriented programming in the logic of web service, it is possible to develop the quality attributes like maintainability, reusability. The functionalities modelled through the web service fuses the implementation of core and crosscutting concerns. BPEL is used for the composition of web services and models both core business logic and crosscutting functionalities. A typical SOA application development involves designing software components for reuse and converting realized software components as web services and for service consumptions in end user applications.

II.ASPECT WEAVER

Aspect weavers take instructions known as advice specified through the use of point cut and join points, special segments of code that indicate what methods should be handled by aspect code. The implementation of the aspect then specifies whether the related code should be added before, after, or throughout the related methods. By doing this, aspect weavers improve modularity, keeping code in one place that would otherwise have been interspersed throughout various, unrelated classes. It processes the source code and produces the byte code. A weaver produces byte code that conforms to the java byte code specification, allowing java virtual machine to execute those class files. Without an aspect weaver, this feature would necessitate duplication of code in the class for every method. Instead, the entry and exit code is defined solely within the aspect. The aspect weaver analyzes the advice specified by the point cut in the aspect and uses that advice to distribute the implementation code into the defined class. Figure 3 shows the combination of Aspect code and base code the weaver get compiles it and produced the modified code. The Aspect weaver acts here as the compiler to compile the base code and aspect whether correctly working are not. Linking aspects with other applications and to compile the code. This is the main function of Aspect Weaver. By means of Aspect Weaver new services can be activated at any time and older services are deactivated at any time. Service is a piece of functionality delivered to the service consumer.

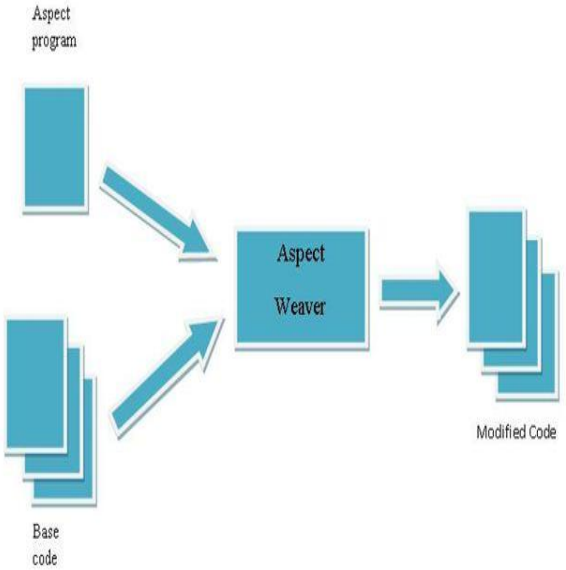


Figure 2 Structure of Aspect Weaver

III.POINT CUT

The Figure 3 shows that point cut is a set of join points. Whenever the program execution reaches one of the join points described or explained in the point cut, a

piece of code associated with the point cut (called advice) is executed. This allows a programmer to describe where and when additional code should be executed in addition to an already defined behavior.

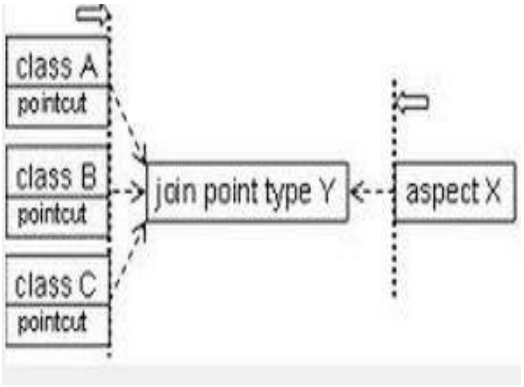


Figure 3 Structure of Point Cut

IV.JOIN POINT

A join point is a specification of when, in the corresponding main program, the aspect code should be executed. The join point is a point of execution in the base code where the advice specified in a corresponding point cut is applied. A join point is an executable point in a system. A call to a method is called field access. AspectJ deliberately exposes only a subset of all possible join points. AspectJ can be implemented in many ways, including source-weaving or byte code-weaving, and directly in the virtual machine (VM). In all cases, the AspectJ program becomes a valid Java program that runs in a Java VM. Classes affected by aspects are binary-compatible with unaffected classes (remain compatible with classes compiled with the unaffected originals). Supporting multiple implementations allows the language.

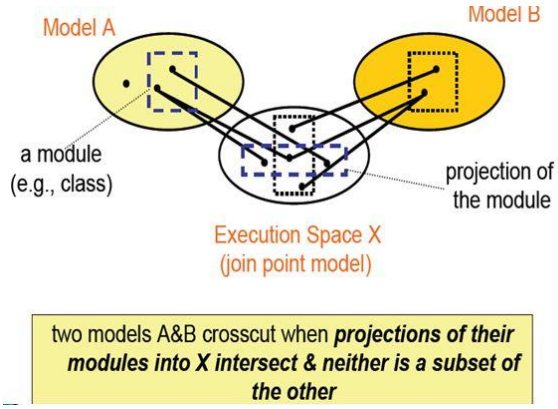


Figure 4 Structure of Join Point

VII.ADVICES

AspectJ supports dynamic crosscutting through advice, a method like construct that defines crosscutting action at the join points selected by a point cut. we need to surround the original code with the caching logic: obtain a value from the cache, execute the original code if the cache doesn't contain a value, and add the value to the cache after executing the original code. AspectJ offers three kinds of advices.

1) Before advice: Executes prior to the point execution.
2) After advice : Executes after to the point execution
3) Around device: This advice has the ability to continue the original execution with the same context or altered context, zero or more times

V.CODE SCATTERING

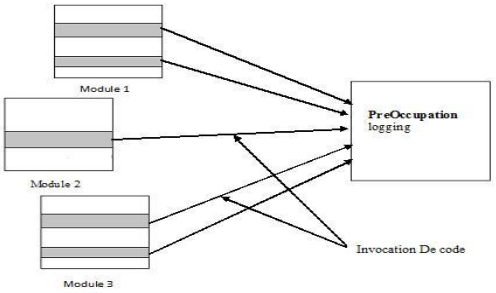


Figure 5 Structure of Code Scattering

It is caused when a single functionality is implemented in multiple modules. Because cross cutting concerns, by definition are spread over many modules, related implementations are also scattered over all modules. The above figure 5 shows its structure.

VI.CODE TANGLING

A concern is tangled if its code is spread out over multiple modules. The concern affects the implementation of multiple modules. Its evaluation is not modular. Developers often consider concerns such as performance, business logic, logging, security and so forth when implementing a module. Figure 6 shows the structure of code tangling.

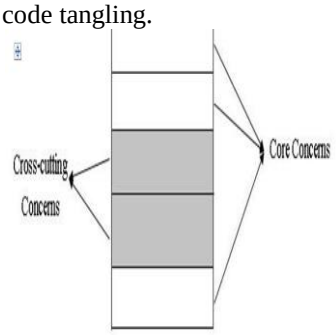


Figure 6 Structure of Code Tangling

VIII.CONS OF SCATTERING AND TANGLING

Scattering and tangling of behaviour are the symptom that of a concern is not well modularized. A concern that is not modularized does not exhibit a well

fledged defined interface. The interactions between the implementation of the concern and the modules of the system are not explicitly declared. They are encoded implicitly through the dependencies and interactions between fragments of code that implement the concern and the implementation of other modules as well. The lack of interfaces between the implementation of crosscutting concerns and the evaluation of the modules of the system impedes the development, the evolution and the maintenance of the system.

XI.PERFORMANCE METRICS

Greater the number of operations inherit makes it more complex(Inheritance) gets reduced, Larger the number of operations makes the code length more(Weighted operations in a module) gets reduced, If the Lines of Code gets increased more operations need to inherit are reduced when using aspects in service oriented architecture.

X.RESULT

In this different web services were mentioned for the Service Oriented Courseware Information System were created. These web services were made to run on different machines with different operating systems to achieve platform independence. Composition of web service is done using Glassfish. The future of this project work involves, in making BPEL engine aware of aspect and then introducing aspect into the BPEL process and to creating an aspect weaver that in turn weaves the advice defined in the point cut into the join point. By using of Aspect we can add service even at run time.

XI.CONCLUSION

AOP enables modular implementation of crosscutting concerns, and modular reasoning in the presence of crosscutting concerns. But it requires an important change in how module interfaces are specified. With AOP interfaces are defined as aspects cut through the primary module structure. So a module's interface cannot be fully determined without a complete system configuration. But crosscutting concerns inherently require global knowledge to support reasoning. Using AOP, programmers get modular reasoning benefits for crosscutting concerns whereas without AOP they do not. In the future work, by means of AOP to support the compatibility futures.

REFERENCES

[1] Anderson.N, Mache.J, and Watson.W "Learning CUDA: Lab Exercises and Experiences" in Proc.ACM Int.Conf.Companion Object Oriented Program 2011 pp 183-188.
[2] Bockisch, Haupt.C, Mezini.M "Virtual Machine Support for Dynamic Join Points, Int.Conf on Aspect-Oriented Software Development, 2004, ACM Press 83-92.
[3] Chiba.S, Nakagawa.K, Josh:"An Open AspectJ Like language", International Conference on aspect-Oriented Software Development, 2004, ACM Press 102-111.
[4] DeLine.R and Fahndrich.M "Typestates for Objects" European Conference on Object-Oriented Programming, 2004
[5] Eichberg.M, Mezini.M and Osterman.K "First-Class Pointcuts as Queries. Asian Symposium on Programming Languages and Systems, 2004, Springer Lecture Notes.
[6] Fowler.M and Beck.K "Refractoring: improve the design of existing code" Addison-Wesley, Reading, MA 1999.
[7] Grundy.J, and Lesiecki.N "Mastering AspectJ: Aspect Oriented programming in Java" International Symposium on Requirements Engineering, 1999, IEEE Computer Society Press, 84-91.
[8] Hannemann.J and Kichzales.G" Using Design Pattern Implementation in Java and AspectJ" Symposium on Object Oriented Programming: Systems, Languages and Applications, 2002, 161-173.
[9] Hilslade.E and Hugunin.J "Advice Weaving in AspectJ" International Conference on Aspect-Oriented Software Development, 2004, ACM Press, 26-35.
[10] Kiselev.I "An Aspect-Oriented Programming using AspectJ" Sams, Indianapolis, Ind.2003.
[11] Laddad.R "AspectJ in Action: practical aspect-oriented programming" Greenwich, CT, 2003.

BIOGRAPHY

Mr. R. Kabilesh

The author is currently an ME Student in Computer Science and Engineering Department at Kalasalingam Institute of Technology. He had completed B.E from Government College of Engineering Tirunelveli.

Mr. P.C.Dinesh

The author is an Assistant Professor in Department of Computer Science and Engineering Department at Kalasalingam Institute of Technology. He received his BE Degree from National Engineering College and M.E. Degree from SSN Engineering College.