ISBN 978-9-8108-3815-7
Proceedings of 2009 International Symposium on Computing, Communication, and Control
Singapore, *9-11 October*, *2009*, *pp. 140-144*

Aop - Introduced Crosscutting Concerns

Goutam Kamble [+]

Maratha Mandal's Engineering College, Belgaum.

Abstract. The "Aspect-Oriented Programming (AOP)" languages such as AspectJ which offer new mechanisms and possibilities for decomposing systems into modules and composing modules into systems. By using AOP and AspectJ, the code size can be reduced and also improve the modularity of crosscutting concerns. AOP introduces aspects, which encapsulate behaviors that affect multiple classes into reusable modules and aspects are concerns that crosscut and a programming construct. A concern is a particular goal, concept, or area of interest. For example, a borrow card processing system's core concern would process book transactions, while its system level concerns would handle logging, transaction integrity, authentication, security, performance etc. Many such concerns are known as crosscutting concerns. These concerns tend to affect multiple implementation modules.

Keywords: crosscut, joinpoint, weaveing, advice, etc.

1. Introduction

"Aspect-oriented programming (AOP) is a new programming technique that allows programmers to modularize crosscutting concerns. This introduces aspects, which encapsulate behaviors that affect multiple classes into reusable modules. The use of aspects for protecting design modularity, transactions, security, management, profiling, tracing, failure handling, optimistic concurrency, etc. Aspect-Oriented Programming (AOP) that allows to express safety concerns separately from the functional part and to enforce them on programs. It enhances our ability to express the separation of concerns necessary for a well-designed, maintainable software system. Some concerns are appropriately expressed as encapsulated objects or components. Others are best expressed as cross-cutting concerns. A cross-cutting concern is a concern that affects several classes or modules. Hence, Aspect oriented programming (AOP) overcomes this problem by introducing a new unit of modularity—an aspect. Aspects allow programmers to avoid of code tangling and scattering, which adversely affect the readability, understandability, maintainability and reusability of the software [2], [3].

2. Features:

- Solve the problems of logging, tracing and security.
- Identify crosscutting concerns & implement to solve the feature [1].

3. Terminologies Introduced by AOP

Major terminologies are given below.

3.1. Aspects

An aspect is a modular unit designed to implement a concern. An aspect definition may contain some code (or advice) and the instructions on where, when, and how to invoke it. Depending on the aspect

[+] Corresponding author.
E-mail address: (goutamkamble@yahoo.co.in).

language, aspects can be constructed hierarchically, and the language may provide separate mechanisms for defining an aspect and specifying its interaction with an underlying system.

3.2. Join points

A join point is a well-defined point in the code at which our concerns crosscut the application. Typically, there are many join points for each concern. If there were just one or two, we could manually change the code with little effort.

3.3. Advice

Advice is the behavior to execute at a join point. For example, this might be the security code to do authentication and access control. Many aspect languages provide mechanisms to run advice before, after, instead of, or around join points of interest.

3.4. Pointcut designator

A pointcut designator describes a set of join points. It provides a quantification mechanism—a way to talk about doing something at many places in a program with a single statement.

3.5. Weaving

Weaving is the process of composing core functionality modules with aspects, thereby yielding a working system [4]. Various AOP languages have defined several mechanisms for weaving, including statically compiling the advice together with base code, dynamically inserting aspects when loading code, & modifying the system interpreter to execute aspects [1].

4. Method

Schematic diagrams of design are given below.

4.1. Use Case-Diagram 1

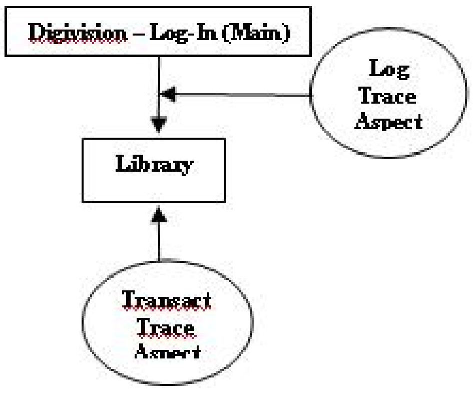

Fig.1: Authentication model

4.2. Use Case Diagram 2

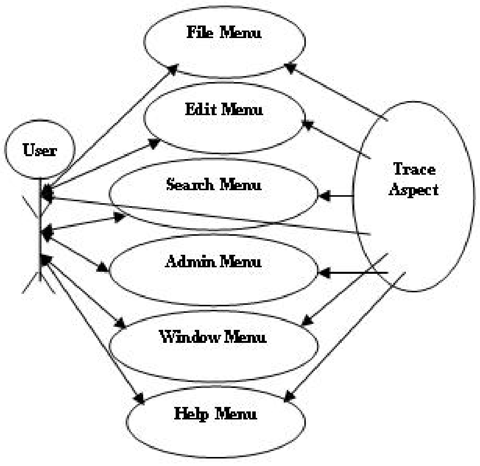

Fig.2: project oriented

Initially, administrator / Staff / Member user Logs-in using his username and password. If both the user name and password are correct, then the user gets into the main application section based on his / her selection. Else he/she will be simply rejected for the connection. In this case only the administrator has full authority to access all the resources related to this software, otherwise a normal users have restricted access. Once entered, the facilities available can only be shown using the screen shots as given below. Each screen shot includes an explanation about that screen where, why and when it is used. As the whole application is being tracked or traced by the administrative aspects, which are predefined for the purpose. Hence no need to say about a specific module that, it is being traced[5].

5. ASPECTS introduced

The Implementation of Digital Library System includes the Aspect program, i.e. TraceAspect.java, which includes following pointcuts:

- Use of **Logging Aspect** to Crosscut Logging to every module.
- Use of **Tracing Aspect** to trace all the methods of the System.
- Use of **Handler Aspect** to know the exception handling by the system.

As explained earlier, the Aspect file 'MasterTrak.java' created with required pointcuts & particular Advises. Following is the list of pointcuts and their functionality with respect to the aspect defined. The File Handling where in the Random Access File is used for the read/write operations Dynamically. Right from the Log-In event to Log-out event of the application the "Trace.txt" file keeps the record along with the Date & Time stamp notifications. Hence, any clash or confusion or related to additions or modifications arises the operator shall be held responsible, as the user of the application is traced by 'Trakuser.java' file also sends a record to this Trace.txt file.

6. POINTCUTS implementations

- **pointcut authent()**:

call (public void Trakuser.userlogin(..));

The first pointcut traces the call to a particular method i.e. Trakuser.userlogin(..). This method provides the login facility to the authenticated user, through this method user used to enter into the application. Suppose the user entered his/her login and password gets matched with system, then the system allow to enter into the application otherwise system is not allowed to enter in the application.

- **pointcut failed():**

call (public void Trakuser.msgFAIL(..));

This pointcut refers to the login failures or false log-in tried by fake users some time. This leads to the message display as well as record writing to file. By this one can monitor that some body is trying to hack the system without proper authentication.

- **Pointcut crosscut():**

(execution(**.*(..))|| execution (*.new(..))) && !within (TraceAspect ||Digivision || FrmSplash||Trakuser);

This pointcut trace execution of any (*.*) class/method and execution of any class constructors (*.new (...)) consisting of any number of parameters. But not to trace within (!within) the listed classes. Accordingly the advices given in the program for some action to do either before() or after() the pointcut occurs.

- **pointcut okayed():**

call (public void Trakuser.msgOK(..));

This pointcut to trace the OK button on any form is pressed to the confirmation of user's action.

- **pointcut cancelled ():**

call (public void Trakuser.msgCANCEL(..));

This pointcut to trace the CANCEL button on any form is pressed to the confirmation of user's action.

- **pointcut trakquit():**

execution (public void LibrarySystem.quitApp());

This is the final aspect where in it is checked that whether the user had normal exit from the application. The last line of Trace file shows this event with the call to quitapp() method of Librarysystem class.

- **pointcut exephandl()**: handler(java.io.FileNotFoundException) || handler(java.io.IOException) || handler(java.sql.SQLException);

This pointcut is used for the exception handling of the system. For any particular exception, using the handler keyword the particular Java exception occurrence and its relative advises is displayed [6], [7].

In the connection of user, first screen appears as a login screen to enter his assigned user name & password. The application are concerned I have made three categories viz., Administrator, Staff & Member. Accordingly the data base tables are also created for each of them. The Administrator is a Super user having full rights of Adding, Deleting & Modifying records. The staff has limited permission based on his nature of duties like Issues & Returns of books, Common Search options etc., where as the Member has very restricted options of Search alone. By this Aspect, the tracing activity starts with complete date & time stamp including the user name. These all things are recorded separately as a proof in a text file for later reference as explained above.

7. Results and Discussion: Some sample menu are shown below-

Fig.3: add new member menu Fig. 4: View Fig. 5: Library Statistics. Fig.6: Aspect Trace Screen.

Figure 3, adding a new member is based on his / her category. This includes Teaching, Non-Teaching and Student. Based on category the number of books & number of days to retain the books with them decides. As in case of Books & Members, category exists for Magazine also. Categorization is based on the subject type of the magazine received. The file menu last option provided is 'Exit'. From here the application terminates.

Figure 4, viewing of different section reports are done through this view Menu. In the above figure all the different views supported by the application are all displayed. [e.g.: Member information, Books and Magazines information]

Figure 5, Library statistics screen provides all details about magazines, books, issued books/magazine, available books/magazines and members are in number.

Figure 6, Aspect's screen shown is from the DOS Screen, who has Logged-In(first Line), Current Date & Time - as when the application started (second line) and each of the program modules accessed on a separate line with transaction number till the end of session as when it ended including end time stamping.

This process also includes tracking the OK Button & CANCEL buttons pressed. Any logging errors & exceptions handling errors occurred etc., Finally when a transaction stops the end of transaction along with ending time also recorded. As said earlier, the complete listing of transaction in a systematic format stored on the disk in text file format.

8. Related Work

This paper introduced the concepts of aspect oriented programming and software development. It identified method as the best method to identify relationships between the concerns. It approach provides support for aspect-oriented development at analysis and design levels [7]. At the analysis level, it is carried out by first identifying a set of actions in the requirements list which are, in turn, used to identify crosscutting behaviors. The approach is supported by a tool that creates relationships between concerns and the requirements that mentioned those concerns.

9. Conclusion

Aspect-Oriented Programming introduces a way of handling crosscutting concerns and aspects components. All these components cross-cut each other in a system's implementation. Based on this analysis, we have been able to develop aspect-oriented programming technology that supports clean abstraction and composition of both components and aspects. The key difference between AOP and other approaches is that AOP provides component and aspect languages with different abstraction and composition mechanisms. A special language processor called an aspect weaver is used to coordinate the co-composition of the aspects and components. We have had good success working with AOP in several tested applications. The AOP conceptual framework has helped us to design the systems, and the AOP-based implementations have proven to be easier to develop and maintain, while being comparably efficient to much more complex code written using traditional techniques.

10. Acknowledgements

I express my sincere thanks to Gregor Kiczales who has given the valuable information and idea about AOP. I also expressed my sincere thanks to all authors mentioned below who have given ideas to use AOP in the applications. Last but not the least, I thank all my friends who helped me directly or indirectly to complete the paper.

11. References

[1] Marius Marin A Tool-Set to Query Source Code for Crosscutting Concerns The Netherlands Marius. Marin@accenture.com

[2] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier and John Irwin Aspect - Oriented Programming Xerox Palo Alto Research Center*

[3] Lars Rosenhainer "A Method for Handling Requirements-Level Crosscutting Concerns" Department of Computer Science, Chemnitz University of Technology, 09107 Chemnitz lars.rosenhainer@informatik.tu-chemnitz.de

[4] L. Rosenhainer. "Identifying Crosscutting Concerns in Requirements Specifications". In Proceedings of OOPSLA Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop. October 2004, Vancouver, Canada.

[5] László Lengyel, Tihamér Levendovszky, Hassan Charaf Aspect-Oriented Techniques in Metamodel- Based Model Transformation Budapest University of Technology and Economics Goldmann György tér 3, H-1111 Budapest, Hungary lengyel@aut.bme.hu, tihamer@aut.bme.hu, hassan@aut.bme.hu

[6] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. & Irwin, J., Aspect-oriented programming. European Conference on Object-Oriented Programming, 1997, 220-242.

[7] Niklas Påhlsson Aspect-Oriented Programming Department of Technology University of Kalmar S– 391 82 Kalmar SWEDEN.