Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 1
Fcbruary 2009

Study on the Software Development Based on AOP Technology

Wei Wang & Wenju Liu
College of Computer and Automatization
Tianjin Polytechnic University
Tianjin 300160, China
E-mail: syww126@163.com

Abstract

With more and more extensive application of AOP (Aspect Oriented Programming), almost all OO languages are extended from aspect orientation, and the aspect adjustment mode implements the basic task of weaver through a set of class. In this article, we introduce the basic concept and programming idea of AOP, expatiate on the software development approach based on AOP and one implementation tool, AspectJ, and introduce some application examples of AOP.

Keywords: Aspect orientation, Object orientation, AOP, OOP, AspectJ, Design pattern

1. Introduction

With continual extension of computer application software and continual enhancement of complexity, traditional software development methods such as process oriented programming and object oriented programming (OOP) have not gradually adapted this change. In recent years, a new programming method, AOP has been concerned by domestic and foreign scholars, and it was appraised as one of ten sorts of technology which had most influences to the economy and human living and work mode by MIT Technology Review. At present, there are AspectC, AspectC++, AspectJ and other languages to support AOP.

2. Production background of AOP

2.1 Problems faced by process-oriented programming

The process oriented programming is a sort of programming method from top to bottom, and its essential is to decompose the software from function, and it is fit for small-sized software system. In the large-sized application system, the method from top to bottom has many intrinsic disadvantages whether for the confirmation of system structure, the evolvement and maintenance of system or for reusing of software.

2.2 Problems faced by traditional object-oriented programming

Traditional OOP language acquired large successes because of its good encapsulation, arrangement and inheritance, and the object model can be better mapped to the actual domain. But in the lifecycle of software, it has following disadvantages.

(1) In the design stage, because of taking class as the unit to organize the modeling, so it can not comprehensively reflect the demand of software system.

(2) In the coding stage, the idea which encapsulate data and method into the class increases the security of data and the modularization of software, but it reduce the reusing of code.

(3) In the maintenance stage, because there are various special applied codes in the classes, so it is very difficult to maintain the system.

23 sorts of design module put forward by GoF (Erich Gamma, 1994) relieves thus situation, and he classified common problems according to module, and translated traditional program module division method by function or object into the program module by system character, which is the basic idea of AOP.

2.3 Production of AOP

In the European Conference on Object-Oriented Programming (ECOOP) of 1997, the top scientist of Palo Alto Research Center of Xerox Corporation, the professor of UK Colombia University Gregor Kiczales et al first put forward the concept of AOP (Kiczales, 1997). After that, the professional proseminar about AOP would occur in ECOOP every

year. At 15 March of 2001, Palo Alto Research Center issued the first support language of AOP, AspectJ.

3. AOP language character and its core idea

3.1 Character of AOP

As a sort of program design method, AOP pays attention to enhance the abstract degree and module character of software, which could fully improve the expansibility, reusing, easy comprehension and easy maintenance of software and enhance other factors influencing the quality of software.

(1) Expansibility. AOP offers the expansible mechanism on the system layer, various relative parts of system will be changed through expanding Aspect (AspectJ supports the inheritance mechanism) or increasing Aspect. One advantage is to fully simplify the testing complexity of software and enhance the testing precision through shielding some Aspects in the software test.

(2) Reusing. The system module in AOP includes system subassemblies and characters influencing these subassemblies, and AOP could enhance the reusing character of subassembly (including class or function) through separating the subassemblies realizing basic functions with applied system characters, and make the reusing of system factors (Aspect) which could not be used to encapsulate to class or function possible.

(3) Easy comprehension and easy maintenance. In AOP, we can use a sort of method with clear boundary to modularize the crosscut concern points, and produce the system structure which could be more easily designed, implemented and maintained.

3.2 Core idea of AOP

AOP uses a sort of method with clear boundary to modularize the crosscut concern points, and produces the system structure which could be more easily designed, implemented and maintained. The crosscut software development by AOP technology could centralize public codes occurring and needed many times to implement, so it could fully reduce the redundancy and coupling of codes, increase the reading character, weaken the degree that designers get in the dilemma situation of deficient design or over design, and even additional demand occurs in the anaphase, AOP could independently realize the demand through separating it in corresponding object and aspect, and it would reduce the maintenance costs of code.

3.3 What is AspectJ

AspectJ is the popular AOP language at present, and it is the AOP expansion based on Java language developed by Xerox Corporation, and it is not only a sort of AOP language criterion, but a sort of language implementation. The part of language criterion defines multiple language structure and the modes that they support the aspect oriented example, but the compiling, transfer and the tool producing the document from code are defined by the part of language implementation.

The language structure of AspectJ is extended from Java language, so all legal Java programs are legal AspectJ programs. The production of AspectJ compiler is coherent with standard Java bytecode. Class document and JVM with any standards could explain the executive codes. Because of selecting Java as its own language base, so AspectJ possesses all advantages of Java language.

To better support the idea of AOP, AspectJ expands following language structure based on the Java language.

(1) Join point. It means the points which could be marked in legal program execution process. For example, the entrance transferring certain method to certain object is a legal join point which could be marked.

(2) Point cuts. It is used to capture the language structure of special join point. And it could appoint and combine needed join points and search special context information of join pints.

(3) Advice. It defines the executive operation at the join point, and its function is similar with Aspect in the classes.

(4) Aspect. The join point, point cuts and advice could be integrated as an independent unit which is similar with the concept of class in OOP.

3.4 Weaver

The process combining the subassembly (object) codes with the codes of crosscut subassembly in AOP is called as weaving, and the weaving process is completed through the weaver. In the language domain, there is not the concept of weaver with complete meaning, and for different aspects, the weaving is implemented through different mechanisms, and only the aspects with definite meaning which could be discriminated by the compiling system could be the legal composing part of program. In AOP language through the expansion of OO language, as a new legal language structure, aspect is translated into one composing part in the object code through the weaver with complete meaning criterion before compiling, and AOP language doesn't limit the meaning of aspect, but strictly regulate the denotation form of aspect.

4. Approaches of AOP design

4.1 Software development approaches based on AOP

AOP adopts a sort of loose coupling mode to realize independent concern points, and then combines these implementations, and establish final system. The system established by AOP is built by the crosscut concern points implemented by loose coupling and modularization, and these modularized crosscut units in the system are aspects, but in OOP, the establishment of system is realized by modularized common concern point coupling, and this sort of modularized concern points are called as classes. The software development mode based on AOP is to organically combine both sides together (seen in Figure 1).

The software development mode based on AOP includes three clear approaches.

(1) Aspect decomposing. The decomposing demand requires picking up common concern points and crosscut concern points, i.e. separating the core module concern points and the crosscut concern points in one system.

(2) Implementation of concern point. For common concern points, we could adopt OOP technology to implement, but for crosscut concern points, we would adopt AOP technology.

(3) Recombination of Aspect. The aspect complier appoints the rule of recombination through Aspect, and the rule regulate how Aspect combines with the basic codes implemented by OOP to establish the final system, and this recombination process is the weaving.

4.2 Language implementation of AOP

The main function implemented by AOP language is to validate the correctness of codes according to the language standard and translate it into the form which could be executed by the machine. The compiler execution of AOP includes two operations, installing concern points and translating the installation result into executive codes. The implementation of AOP could realize the weaving through many modes including the conversion from sound code to sound code. As seen in Figure 2, the Aspect codes implemented by crosscut concert points organically combine with the codes implemented by common concern points through Aspect weaving, and it forms the codes which could be discriminated by basic language and executed by the compiler of basic language, produces executive codes and completes the weaving process from crosscut concern point codes to common concern point codes.

4.3 Application examples of AOP

At present, many research institutions have begun to combine the idea of AOP with actual application, and there are several classic cases.

(1) a-kernel. The a-kernel project group in the UK Colombia University of Canada introduced the idea of AOP into the design of operating system design, tried to enhance the module character of operating system code. They utilized AspectC to modify the kernel of FreeBSD v3.3 operating system, and took the memory page deficiency processing of the operating system as one aspect to deal with, and acquired better effects (Miller, 2001).

(2) Lasagne. Lasagne in Belgium Katholieke University is a sort of middle component based on aspect, and it is mainly used in the service customization of distributing system. Adopting the idea of Aspcet, Lasagne could implement the dynamic program extension without changing codes. Service is taken as one sort of aspect and exists by the form of Wrapper, and it utilizes the weaving idea to implement dynamic selection when running (Truyen, 2001).

(3) FACET. The FACET (Framework for Aspect Composition for an Event Channel) project group of Washington University implements the customized middle component by the method of AOP. They used AspectJ to develop a real time event channel, and the channel possessed better module character, few codes, briefness, easy expansibility and other characters comparing with TAO real time event channel (Hunleth, 2001).

5. Conclusions

As a new programming method, AOP has many works need to be completed in future extensive application. The support languages need to be further enriched and ensured their accuracies, and more tools should be studied to support AOP and fulfill the demands in various stages from software design to maintenance. In the day that the software scale increasingly increases and the software structure is increasingly complex, the software development pattern based on AOP technology would certainly exert more and more important function.

References

Erich Gamma, Richard Helm & Ralph Johnson. (1994). *Design Patterns-Elements of Reusable Object-Oriented Software.* Addison-Wisley.

Hunleth F & Cytron R.Gill C. (2001). *Building Customizable Middleware Using Aspect Oriented Programming.* In: Proc. of Conf. on Object-Oriented Programming, Systems, Languages and Applications 2001 (00PSLA2001).

Kiczales G et al. (1997). *Aspect Oriented Programming.* In: Proc. of the European Conf. on Object-Oriented

Programming (ECOOP). June of 1997.

Miller S K. (2001). Aspect-Oriented Programming Takes Aim at Software Complexity. *Computer 2001 IEEE.* Vol.34. No.4.

Truyen E et al. (2001). *Dynamic and Selective Combination of Extensions in Component Based Applications.* In: Proc. of the 23rd Intl. Conf. on Software Engineering (ICSE' 2001). May of 1997. Toronto. Canada.

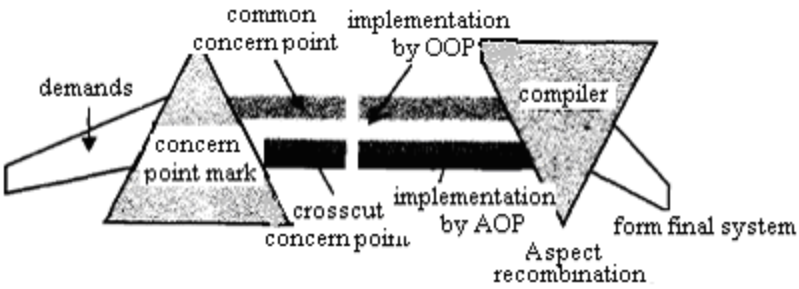

Figure 1. Development Mode by Adopting AOP

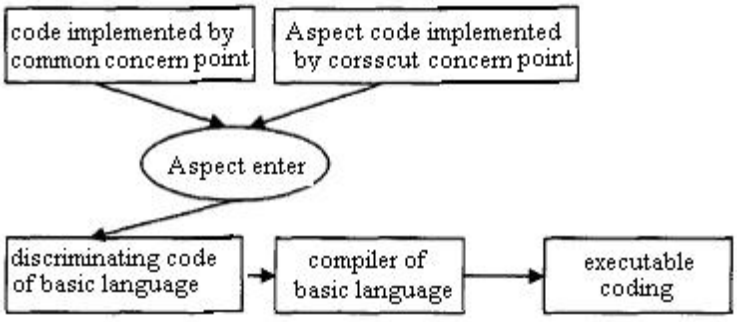

Figure 2. Compiling Process of AOP