ATLANTIS PRESS
Advances in Computer Science Research (ACRS), volume 54
International Conference on Computer Networks and Communication Technology (CNCT2016)

Trusted Software Monitoring Research Based on Dynamic Proxy

Cai-mao LI [a], Shao-fan CHEN, Han-wei WU and Jing CHEN
College of Information Science and Technology, Hainan University, Haikou, 570228, China
[a] lcaim@126.com

Keywords: Dynamic proxy, Monitor, Trusted software, AOP.

Abstract. This paper studied how to monitor trusted software behavior based on dynamic proxy. First, it analyzed the dynamic proxy mechanism, putted forward the dynamic proxy monitor model, then researched how to weave the interceptor code in software monitor based on dynamic proxy, showed the weaving process of monitoring code. Finally, it shows that dynamic proxy can make trusted software have strong behavior monitoring control ability.

Introduction

In an open and dynamic network environment, with the development of the software requirement, software behavior is more and more complicated for the loosely coupled yet tightly cohesive entities and the higher complexity of the software. The loopholes and defects are difficult to avoid as the scale of the software becomes larger and larger, leading to the breakdown or fault, even failure in the operation of the software, which has brought severe effects on people's work and life. So, the credibility of software has become quite a common problem, especially in the characters of availability, reliability and security. To solve the increasingly prominent problem in the credibility, we need to implement the effective monitoring on software behavior. Traditional software monitoring technology, which leads to scattered code, bad modularity, weak ability in monitoring demand expression, tangled and scattered monitoring code, refusing in adding or deleting monitor by the target system dynamically, is incompatible with action monitoring on the large-scale, loose-cohere and complex software. The credibility of the main performances of Software behavior are these, running behavior can be monitored, the result can assess, abnormal behavior can be controlled. Therefore, monitor software for those running applications in critical environments is crucial.

In this paper, we research how to monitor software behavior efficiently based on dynamic proxy technology, improve and enhance the software reliable properties of availability, reliability and safety.

Software Agent

Proxy Pattern

A proxy pattern provides an agent to control access to an object. Proxy pattern can be an intermediary between the client and the target object. It is suitable for a particular object, which doesn't fit or no direct reference to another object. The proxy pattern has three characters. The one is abstract role, which declares real objects and proxy objects common interface. The second is agent role, which includes the reference of the real

object internally, so can operate on real object. Those proxy objects provide the same interface and real object, so that can instead of real objects at any moment. At the same time, attaching other operation, the proxy objects can be executed in real operation on the object, equivalent to encapsulate the real object. The third is real role, which represents real objects being reference object for us finally.

Through a proxy access to access the real object, the proxy pattern implementation mechanism is to implement the same interface with the proxy objects. So, that externally provides unified interface method is the advantage of the proxy pattern. The additional operations of real class in the interface was realized in proxy class, which can be invoked without affecting the external circumstances for systematic expansion. The proxy pattern is divided into static and dynamic agent. Static agent is the relationship determined between the proxy class and the delegate class. Static agent is a compiled proxy class, which source code created by the programmer or tool, and which proxy class bytecode file is in existed before running of the program. Dynamic proxy is just generated in software running stage, during the implementation phase don't have to care about the proxy class.

Dynamic Proxy Mechanism

In a dynamic proxy, the proxy class is created dynamically at runtime. The dynamic proxy processing class was provided two main methods. The one been created at run time is used to bind an agent's business class and return the dynamic proxy classes. The other method is used to define a content of dynamic proxy, which encapsulates the actual call business method.

As shown in Fig. 1, the dynamic proxy classes was generated at runtime. You can invoke the proxy class business method through that procedure call the bind method of dynamic proxy processing binding by proxy class. According to business rules to invoke the methods in the interceptor, the proxy class calls the method of dynamic proxy classes, before and after [2].

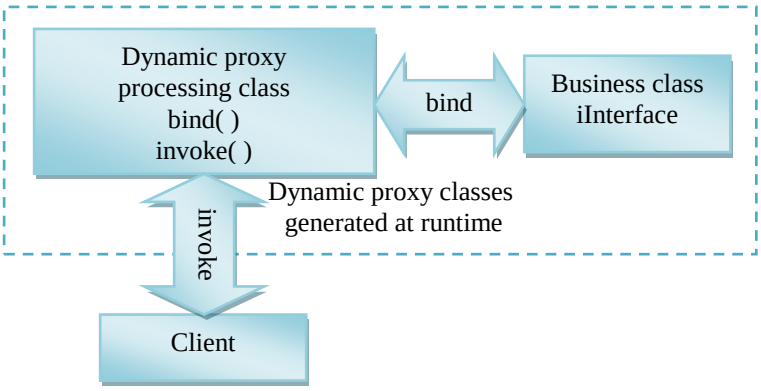

Fig. 1. Dynamic proxy mechanism

Dynamic Proxy Monitor Model

Traditional monitoring model generally adopts the Object-oriented Programming (OOP) technology to achieve monitoring function of software [2,3]. However, OOP monitor solving the problem of crosscutting concerns is slightly insufficient. On the basis of previous studies, this paper used Aspect-oriented Programming (AOP) dynamic agent technology. On the basis of traditional OOP dimension, it introduces the dimension of

AOP, establishes the trusted software monitoring model based on dynamic proxy, as shown in Fig. 2.

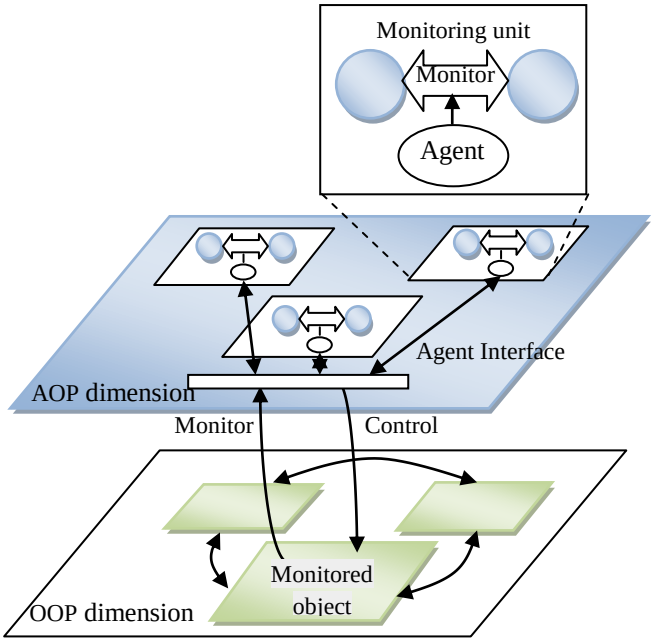

Fig. 2. Dynamic proxy monitor model

Software monitoring mainly includes the monitoring objects, monitoring unit and monitoring framework elements, etc. Monitoring object is a software in a variety of entities. Monitoring unit is the core of the software to monitor management, controlled by monitoring components, monitoring engine and parts [5,6]. First of all, monitoring unit gets the software running state by the monitoring components, and provides the perceptive ability of the software runs for monitoring unit. Then, according to the monitoring information, the monitoring engine produces the corresponding control decision, and the control unit completes the specific implementation. Software monitoring framework provides monitoring services. Monitoring service manages the monitoring information, analyzes the monitoring information processing, provides the basis for the generation of control decision making. In the agent of dynamic monitoring model, monitoring units and monitored objects are separated. Each monitoring unit can independently complete the monitoring of behavior and state. Monitoring framework manages all of the monitoring unit, in order to effectively monitor information, processing and analysis.

As a supplement of Object-oriented Programming(OOP), AOP widely used in processing some system-level service with characteristics of crosscutting, such as transaction management, security, caching, object pool management, etc. The key of AOP implementation lies in that AOP framework creates AOP agent automatically. This study uses java dynamic proxy to implement AOP interception function, which increases section of logic on the basis of the target class, monitors the code into the target system, generates enhanced target class, effectively monitors the trusted system behavior and condition.

Reliable Monitoring based on Dynamic Proxy

Dynamic Proxy Intercept Weave

Software behavior monitoring is to monitor the interaction behavior between software entities, collect information.So,it can provide basic data for diagnosis by forecasting, assessing and controlling behavior [1]. In this paper, based on dynamic agent, we study on crosscutting concerns processing logic to weave into the software behavior monitoring methods of the target class by using aspect-oriented programming (AOP). In AOP, the woven way of AOP can be divided into static weave and dynamic weave. Relative to the static weave, it didn't to have to learn and modify the source code of the target system, the dynamic weaving monitor code didn't weave to the target system of the business code.So this paper studies dynamic weave.

AOP can use JDK or CGLIB (Code Generation Library) dynamic proxy to implement dynamic monitoring code, which been woven into interceptive function. The JDK dynamic proxy is implemented by java internal reflection mechanism, using a very low-level bytecode technology. CGLIB principle is to create subclasses for a class through the bytecode, use the intercept method in the subclass to intercept all calls of the superclass method to weave the crosscutting logic. From the intercept point of setting, a method been called before, during and after to intercept weave, the dynamically monitoring code executes in the process of software entity interface method call.

Weaving Code of Monitoring based on Dynamic Proxy

Software reliable monitoring problem belongs to the crosscutting concerns in the system. It used agent to implement the functions of interception by aspect-oriented programming. Based on dynamic proxy, reliable monitoring is the target system run-time dynamic woven into the code, which will not affect the function of monitored entity.It implements the monitoring dynamics.Making software entities can dynamically join or leave the body. AOP agent is an object generated by the AOP framework dynamically, which used as the target object.AOP agent contains all of the target object method. AOP method in a specific point adds enhancement processing, and calls back methods of the target object. Through the breakthrough point of the target object dynamically, AOP proxy class method weaves enhancement processing to complete the enhancement of the target method [4].

Spring AOP is an AOP framework based on java language. Spring AOP can dynamically choose JDK or CGLIB to generate dynamic proxy AOP. Spring defines aspect, pointcut and enhanced processing with AspectJ annotation. AOP agent was generated according to the annotation. Spring AOP generates AOP agent (target class) dynamically, temporarily at runtime, to enhance the proxy class without using any special commands to compile the Java source code.

In this paper, we study using Spring AOP dynamic proxy to weave monitoring code into enhancement processing, which implements trusted software monitoring in the interception of crosscutting concerns. Software behavior monitoring, evaluation, and carry on corresponding processing can be done . Trusted software behavior monitoring system requirements is to monitor the business logic to handle the method call.

First of all, this requires aspects of decomposition, divides the business logic processing and methods for tracking. The former into general concern, the latter for crosscutting concerns. Then uses Spring AOP implementation of common concerns of crosscutting concerns enhancement processing, thus achieves the goal of credible

behavior surveillance. Spring AOP support AfterReturning, Around, Before and After, AfterThrowing enhancement processing. Here are two enhancement processing application briefly.

1) Define general concerns (business logic)

```
@Component
public class task {
    // implement the method task ()
     public String task() {
            // complete business process operation
            ......
     }
}
```

2) Define the crosscutting concerns (aspect)

For the above task class method, in view of the transaction control and logging aspects, AOP enhances processing, such as Around, AfterReturning.

(1) Around enhancement processing

Decorated with @Aspect, the Spring will be a class as a Bean processing. Things control the code of Around enhancement processing is as follows:

```
@Aspect
public class AroundAdviceTask {
    // under com.yft.service.impl package,
    // the execution of all methods of all classes as a starting point
    @Around("execution(* com.yft.service.impl.*.*(..))")
    public Object processTx(ProceedingJoinPoint jp) throws java.lang.Throwable
    {
            // before executing the target method, start business operation
            ......
            // the target method,
            // save the target method after the execution of the return value
            Object rvt = jp. Proceed (new String [] {} "variable value");
            // after executing the target method, end the transaction
            ......
            Return rvt;
    }
}
```

Under com.yft.service.impl package, in the calling any method (Around) before and after, the class will weave processTx(ProceedingJoinPoint jp) method.

(2) AfterReturning enhancement processing

Decorated with @Aspect, the Spring will the class as a Bean processing. AfterReturning enhancement processing logging code is as follows:

```
@Aspect
public class AfterReturningTask {
    // under com.yft.service.impl package,
    // all methods of all classes of the execute as a starting point
    @AfterReturning(returning="rvt",    pointcut="execution(*
       com.yft.service.impl.*.*(..))")
     public void log(Object rvt) {
            // logging operations
            ......
```

```
        }
    }
```

Under com.yft.service.impl package, all methods of all classes in the call after woven into the log(Object rvt) method.

(3) Spring configuration

In the code, the general concerns of *Task* classes to use @Component, crosscutting concerns in the *Bean* class is to use @Aspect. In the aspect of *Bean* to enhance processing (Advice), you need to use @AfterReturning, @Around respectively. Spring opens Spring AOP with zero configuration, The bean configuration file bean.xml code is as follows:

```
<?xml version="1.0" encoding="GBK"?>
<beans ..... >
    <!-- Specify the automatic search Bean component, automatic search class -->
    <context:component-scan base-package="com.yft.service,com.yft.advice">
        <context:include-filter type="annotation"
        expression="org.aspectj.lang.annotation.Aspect"/>
    </context:component-scan>
    <!-- Start the @Aspectj support -->
    <aop:aspectj-autoproxy/>
</beans>
```

From the above enhancement processing, we can see the developer is in the *Bean* using the @Aspect. For the Spring AOP, the program does not need to use special compiler, weave machine for processing.

Conclusions

This paper studies the dynamic proxy AOP implementation using java technology. The emergence of AOP is a good complement to OOP. AOP is widely used in processing some system-level services. With characteristics of crosscutting, it allows developers can use a more elegant way to deal with a cross-cutting nature of service. This is to provide the technical support for reliable software to monitor. Spring AOP for general concerns (business logic) class at runtime to generate the AOP proxy class,which call is AOP agent enhancement processing method.So,it is good to achieve dynamic proxy mechanism of trusted software monitoring.

Acknowledgment

This work was financially supported by Natural Science Foundation of Hainan Province (613158).

References

[1] Canjun Wan,Changyun Li, Application Research of Computers,2009:1201-1204(In Chinese).

[2] Xi Yang, Tong Li, Computer Engineering and Applications, 2013, 49(23):39-44 (In Chinese).

[3] D.Hachani, D.Bardou,Proc, Workshop Reuse in Object-Oriented Information Systems Design, 2002:345-354.

[4] Grundy J, 4th IEEE International Symposium on RE,IEEE Computer Society Press, 1999.84-91.

[5] Jianming Chen, Song Liu,Zhishu Li, Gejian Ding, Computer Engineering, 2011(37):66-68 (In Chinese)

[6] Canjun Wan, Changyun Li, Journal of Computer Applications,2011(31):572-576 (In Chinese).