一种基于AOP的构件合约化测试方法与实现

叶婷婷，王映辉
（西安理工大学 计算机科学与工程学院，陕西 西安 710048）

摘 要：构件合约化测试方法可用于改善软件质量、提高软件可靠性，但在其使用过程中，容易造成合约代码和业务逻辑代码的交叉混乱，影响代码的封装性和复用性。针对该不足，文中提出一种基于AOP的构件合约化测试方法，分别运用横切关注点和纵切关注点实现合约代码和功能代码，达到代码分离的目的。最后通过架构构件测试平台验证该方法的有效性。实践结果表明，该方法能有效地解决代码之间的混乱问题，提高代码的可维护性，对实现高内聚度的软件构件提供一定的理论支持。

关键词：代码混乱；软件构件；合约；测试；AOP
中图分类号：TP31 **文献标识码**：A **文章编号**：1673-629X(2012)06-0071-04

A Contract Testing Method and Implementation Based on AOP for Component Software

YE Ting-ting, WANG Ying-hui
(School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: Contract test of the software-component is used to improve software quality and software reliability. However, it is easy to cause crossover chaos between contract code and business logic code in the process of using, which has affected the encapsulation and reusability of code. According to the deficiency, a method of the contract test of software component based on AOP(Aspect Oriented Programming) was proposed. In this method, contract code and function code was realized by cross-cutting concerns and vertical-cutting concerns respectively to achieve the code separation. Finally, the validity of this method is verified though the building of component test platform. The method proposed in this paper can effectively resolve entanglement problems between the contract code and the function code and improve the maintainability of code, as well as provide some theoretical support to the realization of software component of high concentration.

Key words: code confusion; software component; contract; test; aspect-oriented programming

0 引 言

基于构件的软件工程已经成为软件开发的主流范型[1]，而构件固有的一些特点，诸如内部信息屏蔽、演变速度较快以及构件间的异质、松耦合等，给软件构件系统的测试带来极大不便，寻求良好的软件构件测试技术和开发实用的测试工具是软件业一直追求和要解决的课题之一。

合约设计DBC(Design By Contract)是一种用于提高软件可靠性的系统化设计方法，也是目前一种流行的软件易测试性设计方法[2-5]。该方法通过对合约的监视和检查，可以容易地发现构件之间的交互性错误，从而改善构件质量，提高构件的可维护性。尽管该方法已经成为一种被普遍接受的软件开发方法，但在实现时仍需解决两个关键问题[6]：

一是合约测试代码的编写对于系统功能代码的编写者而言是一种“额外的负担”；

二是合约代码和业务逻辑代码这两类不同的关注点代码纠缠在一起，严重影响了代码的封装性和可复用性，造成了代码混乱和代码分散。

文中给出了一种基于面向方面编程AOP(Aspect Oriented Programming)的软件构件合约化测试方法，其采用AOP中的动态横切机制降低了系统中业务逻辑与横切关注点的耦合性，有效减少了功能代码和合约代码的混合度。在此基础上，实现了一个软件构件测试的原型平台。

1 基于AOP的构件合约化测试方法描述

AOP技术，是Xerox Palo Alto研究中心的研究人

收稿日期：2012-02-16；**修回日期**：2012-05-20
基金项目：陕西省重大科技创新项目(2009ZKC02-08)；陕西省教育厅中试项目(09JC08)
作者简介：叶婷婷(1985-)，女，硕士研究生，研究方向为软件构件；王映辉，博士，教授，博导，研究方向为软件工程和模式识别。

员于 20 世纪 90 年代末提出的一种新的程序设计思想和模型[7~9]。通常开发人员在编写应用程序时，包含两种代码：一种是和业务系统有关的代码，称之为核心关注点；另一种是和业务系统关系不大的代码，例如日志、权限验证、异常处理等，称之为横切关注点。AOP 使用横切（Crosscutting）、通知（Advice）、编织（Weaving）等技术将系统中的这两类关注点分别模块化，通过方面编织器对已模块化的两类关注点进行编译，从而构造出新系统。

AOP 这种封装横切关注点的思想可有效解决合约测试中的代码纠缠问题。同时，其语言结构与合约之间也存在着某种天然的对应关系。如果将前置条件、后置条件和不变式等合约测试代码看成是横切关注点，那么，AOP 就可以将它们从业务逻辑代码中分离出来，单独封装成 AOP 语言中的方面，通过其语言特性（连接点、切入点、通知）指定需要检查合约的具体位置，并运行合约测试代码判断业务逻辑代码是否满足系统需求，从而达到测试软件构件的目的。

对于构件使用方来说，源代码往往不可得。但是从平台适用性、可扩展性角度考虑，在设计时将含有业务核心的源代码和合约代码分别模块化，使之能够进行相互独立的设计、实现和维护。平台内嵌 AspectJ 的编译器[10~13]，能够在编译时提供横切代码的织入，然后根据不同的上下文环境（测试，部署）生成遵循 Java 字节编码规范的.class 文件。方法的整体结构如图 1 所示。

测试流程分为以下几个步骤：

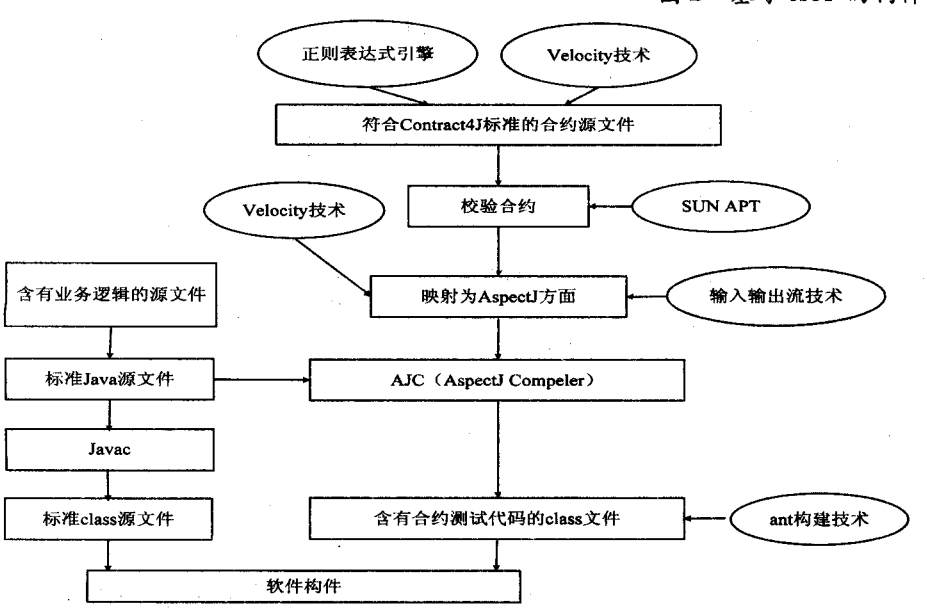

图 1 方法的整体结构图

（1）根据构件的规范和功能需求向构件插入合约，编辑合约的前置条件、后置条件、不变式等。

（2）对合约进行正确性检查。

（3）将合约转换为相应的 Aspect 代码，并将其关联到构件的相关位置。

（4）编译已经插入合约检测代码的软件源代码，得到最终测试结果。

2 构件合约测试平台的构成

按照上述方法，用 Java 语言实现了基于 AOP 的软件构件合约化测试平台。该平台结构主要由文件加载器、合约编辑器、合约检查器、代码转换器、织入器等五大模块组成，如图 2 所示。下面将对这五部分进行详细阐述。

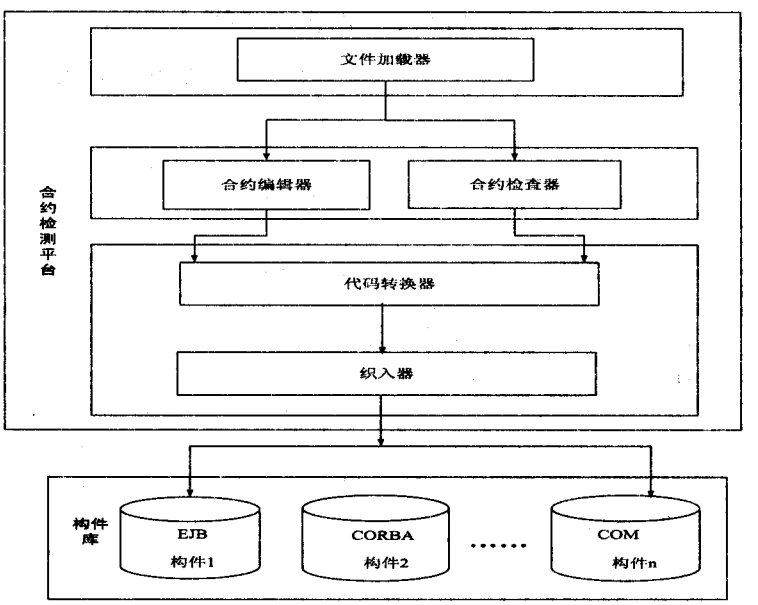

图 2 基于 AOP 的构件合约测试平台结构图

2.1 文件加载器

文件加载器是用户与平台的接口，用来加载需要添加合约的构件。它的实现包括创建 JFileChooser 类、Reflection 类和 JTree 类三部分内容，JFileChooser 类主要实现选择文件，Reflection 类利用反射机制读取文件的包名、类名、方法名等属性，JTree 类使得各层级之间的关系按照树形结构显示。

2.2 合约编辑器

由于合约语言规范较

多,为了降低代码编写者的工作难度,减轻工作量,代码编写者不用掌握合约语法规则,只需要在合约编辑器中填写合约表达式,诸如前置条件、后置条件、不变式等,平台按照预先定义好的合约语言模板,将表达式填充到模板的相应位置,完成合约的编辑工作。其中包括合约添加类和合约模板类两个关键内容。

①合约添加类。

合约既可以添加在类上,表明约束条件是针对类中所有方法的,也可以添加在类中的某个具体的方法上,表明只针对此方法可用。

②合约模板类。

本平台采用 Velocity 模板引擎技术,定义合约模板形式,动态取得合约编辑器中的数据,替换模板中的变量,生成合约代码。合约模板以. vm 文件的形式存在,包括合约名称、类名、方法名、前置条件、后置条件和不变式等内容。以下是合约模板类的核心代码:

```
@ contract $ SomeContract;
public class $ ClassName
{
Invariant $ Inv;
public $ MethodName( )
{
Pre $ PreCondition;
Post $ PostCondition;
}
}
```

2.3 合约检查器

合约正确性的检查主要包含以下几个方面:

①合约语法的正确性。合约的主体是合约表达式,保证合约语法的正确性即保证前置条件、后置条件和不变式语法的正确性。

②合约语义的正确性。代码编写者根据需求说明书填写合约内容,即前置条件是对调用构件的约束性检查,后置条件是对构件完成结果的检查。

③合约不能够直接或者间接地修改原有构件的属性值。

2.4 代码转换器

由于合约代码的书写方式并不符合面向方面编程语言规范的要求,平台必须将合约代码换转成相应的 Aspect 代码,才能达到将合约代码和构件代码分离的目的。代码转换器利用输入输出流技术将合约代码中的合约信息传输到 Aspect 模板中,根据添加合约的位置判断切入点,形成带有合约检测条件的 Aspect 代码。

①Aspect 模板类。

本平台仍然采用 Velocity 模板引擎技术,按照 Aspect 语言要求,定义 Aspect 模板形式,动态读取数据添加到模板中,生成 Aspect 代码,和构件代码形成最终系统。由于 Aspect 代码在程序运行前已被内联至核心功能代码中,代码被高度优化,因此执行速度与未使用 AOP 方式编写的代码相差无几,对性能几乎没有影响。Aspect 模板以. vm 文件的形式存在,包括 Aspect 名称、切入点名称、通知、前置条件、后置条件和不变式等。以下是 Aspect 模板类的核心代码:

```
public aspect $ SomeAspect
{
pointcut $ SomePointcut ( ):
<pointcut logic>;
Before( ) : $ SomePointcut( )
{
$ PreCondition;
}
After( ) : $ SomePointcut( )
{
$ PostCondition;
}
Around ( ) : $ SomePointcut( )
{
$ Inv;
}
}
```

②输入输出类。

将合约代码转换成 Aspect 代码,对应关系为:SomeContract 为 SomeAspect,Pre 为 Before,Post 为 After,Invariant 为 Around。读取需要添加合约代码的类和方法的层次关系,判断切入点的位置,加入切入点逻辑。在连接点之前执行通知,使用 before( )类型的通知;在连接点之后执行通知,使用 after( )类型的通知;在连接点周围执行通知,使用 around( )类型的通知。在切入点上使用 around( )通知的时序图如图 3 所示。

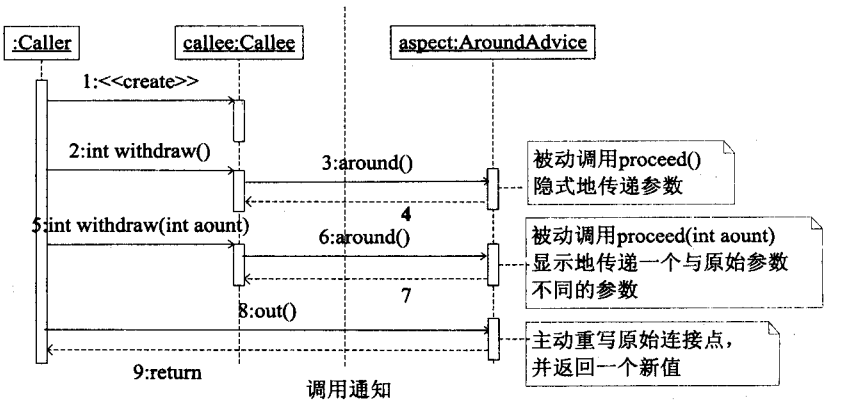

图 3　在 call 切入点上使用 around( )通知的序列图

2.5 织入器

织入的实现机制有多种，从织入的过程看可以分为两类：静态织入与动态织入。本平台采用静态织入方法，即在核心功能代码中的适当位置，比如某段代码执行前，或执行后，将 Aspect 代码织入，从而形成混合的编码。即调用 ajc 编译器，它把 Aspect 特有的关键字编译成字节码织入到主应用程序字节码中，并产生可由 Java 运行环境来执行的、适当的 aj 文件。

3 结束语

构件测试是近年来受到软件测试界普遍关注的一个话题，而构件内部信息屏蔽、演变速度较快以及构件间的异质、松耦合等特点给软件构件系统的测试带来了极大不便。文中提出的基于 AOP 的构件合约测试方法，将合约代码当作横切关注点，与具体的业务逻辑代码（核心关注点）进行了有效分离。同时，基于此方法实现的构件测试平台通过合约编辑器添加合约内容，使代码编写者不必关注合约代码的语言规范和书写形式，降低了代码编写难度，减轻了开发人员的工作量。

如果程序在运行中违背了合约，就需要根据具体情况产生不同的错误提示信息，进行相应处理。文中实现的测试平台仅仅是个原型，只是简单记录了错误信息的相关内容。下一步的工作重点是将原型平台变为实用程序平台。

参考文献：

[1] Vincenzi A M R, Maldonado J C, Wong W E, et al. Coverage testing of Java programs andcomponents[J]. Science of Computer Programming, 2005, 56(1/2): 211-230.

[2] Mayer B. Applying" Design by Conrtact" [J]. IEEE Computer, 1992, 25(10): 40-51.

[3] Gao J Z, Tsao H S J, Wu Y. Testing and Quality Assurance for Component-based Software[M]. Boston: Artech House, 2003.

[4] Cheng Y C, Chen Chien-Tsun, Hsieh Chin-Yun. ezContract: Using Marker Library and Bytecode Instrumentation to Support Design by Contract in Java[C]//Software Engineering Conference. [s. l.]: [s. n.], 2007: 502-509.

[5] 樊庆林，吴建国. 提高软件测试效率的方法研究[J]. 计算机技术与发展，2006，16(10)：52-54.

[6] Feldman Y A, Barzilay O, Tyszberowicz S. Jose: Aspects for Design by Contract[C]//Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods(SEFM'06). [s. l.]: [s. n.], 2006.

[7] 赵艳妮，王映辉，雷　宇. 一种基于 AOP/IOC 的软件框架研究与实现[J]. 计算机工程与应用，2008，44(29)：92-95.

[8] 陈　成，李　行. 基于 AOP 的 MDA 模型转换[J]. 计算机技术与发展，2008，18(7)：87-89.

[9] 古全友，王恩波，胥昌胜. AOP 技术在 J2EE 系统构建中的应用[J]. 计算机技术与发展，2006，16(4)：150-152.

[10] 李志纯，张南平. 面向 Aspect 编程的应用研究[J]. 计算机技术与发展，2006，16(5)：217-220.

[11] Graddeck J D, Lesiecki N. 精通 AspectJ[M]. 北京：清华大学出版社，2004.

[12] 周庆泉，郝克刚. 基于 AOP 的工厂模式研究[J]. 计算机技术与发展，2008，18(8)：47-49.

[13] 尹　涛，李　翔，林　祥. 基于 AOP 的角色访问控制模型设计与实现[J]. 计算机技术与发展，2008，18(10)：136-142.

（上接第 70 页）

[3] Pawlak Z. Rough sets-theoretical aspects of reasoning about data[M]. [s. l.]: Kluwer Academic Publishers, 1991.

[4] 马　良，朱　刚，宁爱兵. 蚁群优化算法[M]. 北京：科学出版社，2008.

[5] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]//Proceedings of 1st European Conference on Artificial Life. Paris, France: Elsevier Publishing, 1991: 134-142.

[6] Parpinelli R S. Data mining with an ant colony optimization algorithm[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 321-332.

[7] Caro G D, Dorigo M. AntNet: distributed stigmergetic control for communications networks[J]. Journal of Artificial Intelligence Research, 1998(9): 317-365.

[8] Shmygelska A, Hoos H H. An ant colony optimization algorithm for the 2D and 3D hydrophobic polar protein folding problem[J]. BMC Bioinformatics, 2005(6): 30-30.

[9] 于化龙. 基于 DNA 微阵列数据的癌症分类技术研究[D]. 哈尔滨：哈尔滨工程大学，2010.

[10] 蔡立军，蒋林波，易叶青. 基于蚁群优化算法的基因选择[J]. 计算机应用研究，2008，25(9)：2754-2757.

[11] Golub T R, Slonim D K, Tamayo C P, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring[J]. Science, 1999, 286(15): 531-537.

[12] Alon U, Barkai N, Notterman D A, et al. Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues by Oligonucleotide Arrays[J]. Proceedings of the National Academy of Science, 1999, 96(12): 6745-6750.