计算机科学 2008Vol. 35№. 9

面向方面软件测试的研究进展

顾海波　卢炎生

(华中科技大学计算机科学与技术系　武汉 430074)

摘　要　面向方面编程软件测试技术是保证面向方面软件产品质量的有效手段。首先对 AO 软件的基本特征进行分析，然后从单元测试、集成测试、回归测试以及测试自动生成工具等多个方面介绍了 AO 软件测试的研究现状，最后展望了 AO 软件测试技术的发展前景。

关键词　软件测试，AOP，非对称范型，对称范型

Progress of Research on Testing Techniques of Aspect-oriented Software

GU Hai-bo　LU Yan-sheng

(Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract　Testing techniques of Aspect-Oriented Programming(AOP) software are very import approach to assure the reliability of the productions. First the common features of AO software are presented, followed by a detailed discussion about basic problems for the research on testing AO software. Then the paper surveys the research works of AO testing thoroughly. Finally the future direction of research on testing techniques of AO software is given.

Keywords　Software testing, AOP, Asymmetrical paradigm, Symmetrical paradigm

面向方面编程(AOP: Aspect Oriented Programming)[1]通过引入方面(aspect)实现了对横切关注点的封装，解决了采用面向过程和面向对象实现造成的代码缠结(tangling)和散射(scattering)的问题，提高了软件开发效率。但是，AOP 的广泛应用需要 AO 软件测试(Aspect-Oriented Software Test)技术保证产品质量。另一方面，AOP 实现引入了新的语言成分，可能带来新的缺陷类型，因而也需要新的测试方法和技术。

本文首先对 AO 软件的基本特征进行分析和介绍，以提供理解相关测试研究的必要背景知识；然后按照单元测试、集成测试以及回归测试的次序，详细介绍了现有 AO 软件测试方法和技术的研究现状；最后对 AO 软件测试技术的发展前景做出展望。

1　AO 软件的基本特征

AO 软件在基本概念、语言实现方式、具体语言结构以及程序开发过程等多个方面都具有新的特征，这些特征对 AO 软件测试研究工作的影响重大。

• 多量化(quantification)和不知觉性(Oblivious)的基本特征[1]使得 AOP 提供的封装机制与合成机制密不可分。

多量化是指方面的单个表示会影响到多个程序模块，不知觉性是指受影响的模块并不包含任何对该行为的显示调用或者声明[1]。在 AO 软件中，横切关注点被封装为方面模块，然后在合成机制中，方面模块体现出与被横切的基本功能模块之间的一对多的强耦合特性，从而使得类似与传统的单元隔离测试难以直接应用到 AO 软件之上。

• 非对称 AOP 范型是主流的 AOP 实现机制。

现有的 AOP 范型包括对称 AOP 范型和非对称 AOP 范型两种。非对称范型的代表是 AspectJ™[2]，该实现由 AOP 提出者 Gregor Kiczales 在施乐研究中心领导的研究小组开发而成。对称 AOP 范型由 IBM 公司的 Watson 研究中心提出，称为多维关注点分离(MDSOC: Multi-Dimension Separation of Concerns)[3]，其前身是面向主题的程序开发[4]，相应的支持工具是 Hyper/J™[5]。关于两者的详细比较可参见文献[6]。作为主流的 AOP 实现机制，非对称 AOP 对横切关注点的实现有两种方式[2]：静态横切方式和动态横切方式。两种不同的途径可能产生不同的缺陷类型，需要新的测试方法对其分别进行处理。

• AOP 提供的方面合成机制，相对于面向对象中的对象继承和聚合而言粒度更小。

基于对横切关注点的实现需要，已有的 AOP 实现机制提供的方面合成粒度多数处于方法级[1,7]，但是也提供了对方法内代码块合成的支持，例如异常处理语句块[8]。因此，面向对象软件的集成测试和回归测试方法与技术无法直接应用。

• 引入一些新的语言成分实现对横切关注点的模块化封装。

以非对称范型的代表 AspectJ 为例，引入了多个新的语言成分[2]：连接点(join point)、连接点集合(pointcut)、advice、类型间声明(inter-type)、方面(aspect)。通过织入(weaving)，将方面中的 advice 代码块添加到与 pointcut 声明匹配的所有连接点处，构成最终系统。这些语言成分从语法和语义角度共同实现了 AOP 机制，对 AO 软件开发产生着直接影响，给 AO 软件测试带来了挑战。

• 非对称 AO 软件与面向对象软件在开发过程上存在包含关系。

顾海波　博士研究生，主要研究领域为面向方面的软件测试、软件工程；**卢炎生**　教授，博士生导师，主要研究领域为现代数据库技术、软件测试、数据挖掘。

非对称 AOP 机制在程序模块层面引入方面部件，实现对横切关注点的封装；而对非横切关注点的实现仍然采用面向对象开发方法，称为基础部件。通过织入器(weaver)将方面部件织入到基础部件，产生“织入后部件”。这个过程可以看作是面向对象软件的演化：织入后部件具有与织入前的基础部件同样的面向对象模块结构，仅仅是被织入的方法内容发生了变化。从而可以采用面向对象软件的集成测试和回归测试技术来验证该“演化”的正确性。但是，这种与演化“等价”的织入，对原有程序的改变程度远大于一般的演化过程，需要 AO 软件测试找到有效的解决方案。

AO 软件的上述特征，强烈地影响到已有的 AO 软件测试研究。而且，为了使测试研究焦点集中于 AOP 机制的相关错误发现之上，当前 AO 软件测试的研究都假设基础部件通过了测试，从而只考虑方面对最终系统引入的缺陷。尽管存在其它类型的 AOP 实现机制，但是非对称 AOP 实现机制在 AOP 的应用中居于主流，而且几乎所有 AO 软件测试的研究都是基于 AspectJ—非对称 AOP 实现的代表。因此本文如果不是特别指出，AO 软件表示非对称 AOP 范型软件。

2 AO 软件单元测试技术

如前所述，AO 软件中的单元种类很多，但是 AO 软件的单元测试主要关注与 AOP 机制相关的部分，因而测试的内容主要包含：1)方面化合成行为[9]的测试；2)方面化行为[9]的测试。AO 软件的单元测试，按照测试途径可以分为两类：1)非隔离测试；2)隔离测试。本节叙述按照测试途径分类，在每个类别里按照测试内容叙述。

2.1 非隔离测试途径

方面单元的非隔离测试方法基于一个基本前提：方面是核心模块程序的增量开发。核心模块部分被看作被测 AO 软件的主体，方面被看作是对这个主体的修改。从而将方面的单元测试问题转化为主题程序的增量集成测试和回归测试：增量集成测试用于检测方面是否实现了预期的功能；回归测试用于检测方面是否引入了非预期的后果。

2.1.1 方面化合成行为的非隔离测试

非隔离方式进行方面化合成行为的基本方法是：1)为了判断是否选择了不需要的连接点，首先收集当前被测 AO 软件内所有匹配的连接点，然后对每个匹配连接点所在的类进行织入，通过对织入结果的测试判断该连接点是否被错误选择；2)为了判断是否选择的连接点集合不完整，通过对被测 pointcut 实现进行变异，采用变异测试技术生成消除此变异的测试用例，执行测试用例后的结果将使开发人员发现此类缺陷。

已有研究中，Mortensen 和 Alexander[10,11]采用变异测试来检测 pointcut 匹配错误，Lemos 等人[12]通过扩展基于代码覆盖和基于错误的测试技术实现测试。Ceccato 等人[13]提出组合覆盖准则(Composition Coverage)和标志符覆盖准则(Designator Coverage)用来检测织入过程中动态控制流匹配错误。上述研究的共同不足是，仅仅通过例子描述了思路，没有提供具体实现算法和验证。

2.1.2 方面化行为的非隔离测试

非隔离方式进行方面化行为的基本步骤是：1)对方面化行为发生前的被测 AO 程序建模，生成测试对象的初始版本；2)对方面化行为发生后的被测 AO 程序建模，生成被测对象的织入后版本；3)根据两个版本的差异进行分析，生成测试用例以覆盖所有变化情况。可见，这种单元测试方式是对被测方面单元的一种间接测试技术。具体对被测程序单元的方面化行为的描述方式有基于规约和基于实现以及两者结合共 3 种类型。

基于规约的方面化行为测试的例子有，徐殿祥等人[14]使用 UML 序列图对 AO 软件建模，支持对方面和类之间的交互进行测试用例的自动生成。其中的关键是，流图可以实现对被测类的方法和被测方面的 advice 实现可测性表示。该方法对方面中的 advice 和类中的方法都用序列图建模：织入前的基础模块类中的方法对应的 UML 序列图称为织入前序列图；由于 advice 对指定类方法的织入产生织入后类方法，因而也可以用序列图对其建模，称为织入后序列图。然后针对选定的覆盖指标从织入后序列图构造流图，接着将流图转换为流树，则从流树中的路径直接生成测试用例。

基于实现的方面化行为测试，包括使用被测方面单元的控制流和数据流信息等多种技术。例如，赵建军提出一种基于数据流图的单元测试方法[15,16]，可以对被测的方面和类使用 3 个层次的测试：模块内测试、模块间测试、公共模块的测试。不足是没有讨论对被测方面和被横切的类的测试冗余，而且测试方法伸缩性不好，即使是简单的 AO 软件也会产生大量的中间计算，导致测试执行效率不高。另外，Zhou 等人提出的 AO 软件的测试过程中包含了基于被测程序单元结构信息的测试方法[17]，Lemos 等人[18]提出的对 AspectJ 程序定义了面向方面的定义——使用数据流图(aspect-oriented def-use data flow graphs (AODUs))，并使用织入后字节码进行测试的方法。

基于规约的测试方法的缺点是测试用例间可能存在严重冗余，而且可能存在没有检测的软件漏洞；基于实现的测试方法的缺点是对与需求规约描述的行为没有被实现的错误情况无法识别。两者结合的测试方法，可以充分利用两者的优点，避免了各自的不足。Mortensen 和 Alexander[10,11]基于之前提出的 AO 软件的 6 种缺陷类型，提出将结构性测试和变异测试(mutation testing)组合的测试方法。其中结构性测试技术用来提供测试充分性覆盖指标，可以检测出与 advice 实现代码相关的 4 种缺陷。Ceccato 等人[13]提出了增量 AO 软件测试过程，用来检测由于 advice 织入后造成被横切方法的后置条件破坏。该文的不足是仅仅限于分析讨论，没有给出原型和关键算法，可行性需要进一步研究。

2.2 隔离测试途径

方面单元的隔离测试等同于传统的单元测试，将被测单元与程序其它部分进行隔离，使得被测单元在测试过程中不受程序中其它部分的影响。由于单于中可能引用其它部分中的模块功能，因而需要在隔离环境内模拟外部对象，常用的解决方法有 Mock Object[19]。目前 AO 软件的隔离方式单元测试技术研究成果较少，其困难在于：1)pointcut 的可能形式难以穷尽；2)难以有效模拟各种情况下对外部对象的引用。

隔离方式进行方面化合成行为的基本前提是：1)测试用例可以直接执行被测方面的 advice；2)测试用例可以手动生成连接点，用来测试 pointcut 表达式的正确性。Yamazaki 等人[20]提出一个对 AOP 程序中方面的单元测试框架 AJTE，可以实现按照方面的需求规约直接生成被测方面的测试方法，不需要织入就可以对方面内的 advice 和连接点匹配模式进行测试，而且该框架可以集成到 AspectJ 编译器中。

3 AO软件集成测试技术

在完成AO软件的单元测试的基础上，AO软件集成测试主要关注的问题是：1)对AO软件中方面集成过程的可能引入的缺陷分析及其测试；2)对AO软件中方面集成过程的测试表示，指导测试过程中进行集成的步骤；3)对集成后的预期结果行为的表示，作为测试结果判定的依据。按照生成测试用例所使用信息的不同来源，AO软件的集成测试可以分为基于缺陷的集成测试、基于规约的集成测试和基于实现的集成测试。

3.1 基于缺陷的集成测试

在方面的集成过程中可能引入缺陷的部分有：方面内多个pointcut对同一个基础类进行横切织入，多个方面横切同一个基础模块，多个方面之间的交互引用等等。例如，Mortensen和Alexander[10,11]针对多个方面横切同一个基础模块时可能产生的方面之间的优先级缺陷，提出了采用变异测试技术的解决方案。不足是讨论的抽象层次过高，导致忽略部分潜在的缺陷类型，例如被横切类对方面的反作用导致的方面内部状态的错误变化等。

3.2 基于规约的集成测试

基于规约的集成测试采用规约信息对方面集成的预期结果进行可测性描述，并采用基于规约的测试方法执行测试。例如，Massicotte等人[21]对其之前提出的基于UML协作图的序列生成技术[22]进行了扩展，针对AO软件中方面和类之间的交互提出了基于方面和类之间UML协作图的测试序列生成技术，对已有的基于UML协作图的覆盖准则进行扩展，给出了5个协作图覆盖准则，并对每种测试准则定义了测试自动生成的迭代过程。该方法关注解决方面对基础类的织入过程可能产生的缺陷，并采用实例验证其实践的可行性。

3.3 基于实现的集成测试

基于实现的集成测试，基于AO软件的代码实现对方面集成过程进行可测性描述，并采用基于结构的测试方法执行测试。一般的方法有基于AO软件的控制流、数据流等等。例如，Zhou等人[17]基于被测AOP程序的控制流和数据流信息提出包含4个步骤的集成测试策略。其不足是认为方面织入的次序对测试结果没有影响，实际上可能存在方面之间依赖产生对织入次序的约束，而且其定义的覆盖准则粒度不够精细。Xie等人[23]基于被测AOP程序的结构分析，从3个层次实现集成测试：1)织入后的类方法；2)方面中advice的测试；3)方面内类型间方法测试。Massicotte等人[24]对文献[21]进行了扩展，可以对单个或多个方面与一组相互协作的类之间的细粒度集成进行测试。该文同时给出了详细实例演示该方法的可行性。

4 AO软件回归测试技术

回归测试面对的问题是，当被测程序经过演化(包括添加，修改或者删除功能模块)后，如何验证被影响到的程序部分的功能正确性。最简单的回归测试是直接采用重新运行原有测试用例的方法，但是，由于修改的各种形式可能影响到原有测试用例的可用性，因而常常需要对测试用例集合做出修改。回归测试的目标是，在验证被测模块的正确性的前提下，最大限度地利用已有的测试用例，减少重新创建测试用例的开销。

AO软件回归测试可以分为基于规约的回归测试和基于实现的回归测试。

基于规约的回归测试方法

这种方法从规约角度对程序的演化前后版本进行建模，通常要用到人工辅助或者逆向工程技术。例如，Souter等人[25]提出一种基于被测关注点的测试方法，采用关注图标识出与被测关注点相关的代码，从而大大减少了对被测程序整体捆装造成的测试开销。徐殿祥等人[26]使用状态机对被测AOP程序建模，并提出相关的回归测试方法，使得对方面进行测试时能够更加有效地重用对被横切类使用过的测试用例。

基于实现的回归测试方法

这种方法从被测AO软件的代码实现角度对程序的演化前后版本进行建模，然后通过捕获变化的程序部分确定回归测试的范围。例如，赵建军等人[27]将Harrold等人提出的针对Java程序的安全回归测试选择技术[28]做出扩展，能够对被测AspectJ程序中的基础模块类、方面、方面和类的交互以及程序整体采用扩展后的控制流图进行建模，然后通过比较修改前后的控制流模型生成回归测试用例，从而可以实现对修改过的单个方面、单个类以及受其影响的程序部分的回归测试。不足是仅仅给出了可行方案，没有实验证实和结果分析评价。而且，该方法无法处理动态匹配的SCFG图构造，也没有考虑多个方面织入同一个方法造成的SCFG图构造数量爆炸问题。Xu[29]为AOP程序的回归测试定义了一种新的回归测试用例选择标准，可以进一步提高选择精度，去除那些虽然经过变化路径但不能揭示修改前后程序行为区别的测试用例。该方法可以有效减少传统方法产生的回归测试用例数量。

比较起来，基于规约的回归测试和基于实现的回归测试各有优缺点。基于规约的回归测试方式能够准确地描述程序的变化部分，完整地表达变化的语义，从而可以产生较小的回归测试用例集合，不足是难以自动化规约的表示。而基于实现的回归测试方式刚好相反，可以自动化回归测试的范围分析以及测试用例生成，但是对于复杂的被测程序难以有效地获取准确的演化差异信息，从而不得不扩大回归测试范围，使得测试效率急剧降低。

5 测试用例自动生成和测试工具

测试工具及原型是测试方法和技术的具体实现，能够充分体现相关测试理论的应用实践价值，并有助于评价具体方法的优越性及局限。

开发者可以先将AO软件转化为织入后的面向对象软件，然后直接使用已经产品化的面向对象软件测试工具对其进行系统测试，这些工具有JUnit[30,31]、伪对象(mock object)[19]和JTest[32]等。AO软件测试工具多数都伴随着AO软件的测试方法和技术的提出而出现，就纯粹用于测试AO软件的工具而言，目前尚未见成型的商业产品，主要以原型工具为主。表1给出了已有的AO软件测试工具的简要介绍，突出各自的适用范围、长处和不足。

表1 典型的面向方面软件测试工具

Tools	AOP Imp.	Testing Tech.	Strong Points	Shortcomings
JamlUnit[9]	JAML	Structure-based unit testing on Advice	① support testing aspect isolatedly Extended from JUnit[30,31]; ②Can testing specitial groups of aspects in a huge a artially Without compiled with whole application by using mock object	①cannot testing pointcut
AJTE[20]	AspectJ	Structure-based Unit Testing	①support testing aspect isolatedly; ②automatic testing procedure; ③can integrated into the exist popular testing frameworks;	①bad performance
APTE	AspectJ	Structure-based unit Testing	①support testing aspect isolatedly; ②automatic testing procedure; ③can collect all join points that match exactly even approacitlly with the pointcut expression under testing	①bad performance
aUnit[33]	AspectJ	Unit Testing	①support testing aspect isolatedly; ②can integrated into the exist popular testing frameworks;	①bad performance
Aspectra[34-36]	AspectJ	Structure-based unit Testing	①Generat test case for the woven class automatically by calling exist popular tools; ③ support evaluation of the test cases generated against the percent of the coverage on the code	① have to generat the package class for the unit tested
Wrasp[37]	AspectJ, Hyper/J	Unit Testing, Integration Testing	①Generat test case for the woven class automatically by calling exist popular tools	① have to generat the package class for the unit tested

6 AO软件测试技术的展望

6.1 对称AOP范型程序的测试研究

由于非对称AOP范型在当前的AOP开发领域居于主流，除了Wrasp[38]宣称适用于对称AOP范型Hyper/J™外，已有的研究工作几乎都是针对非对称AOP范型开展的，缺乏对对称AOP范型程序的测试方法和技术的研究。

在对称AOP范型中，强调非横切关注点与横切关注点的平等对待，都称为方面，并且提供显示定义方面关系的部件，编译器根据该定义进行方面合成。由于对称AOP范型与非对称AOP范型存在逻辑组成上的不同，而且具体的开发过程也有很大差异[44]，显然需要不同的测试方法和技术。

通过对比两种范型，我们发现：在对称AOP范型程序中，方面之间的合成关系是显示指定的；而非对称AOP范型程序中，方面与类之间的合成采用字符串模式匹配进行，特别是通配符的使用使得方面可以合成的模块规模不确定，而这一点常常是那些很难发现的程序缺陷的根源。因此，我们认为，对称AOP范型程序比非对称AOP范型程序的可测性更好。这些都需要针对对称AOP范型程序测试进行深入研究。

6.2 AO软件的缺陷模型的深入研究

对AO软件的缺陷模型的研究尽管已经有了初步成果，但是还处于初级阶段。这主要体现在3个方面：1)已有的缺陷类型讨论限于非对称AO软件；2)对于非对称AOP机制，仅仅考虑了动态横切的实现，没有涉及静态横切造成的缺陷类型讨论，例如由于方面的静态横切造成的数据耦合产生的缺陷；3)动态横切相关的缺陷类型不完整。例如，关于方面与被横切类之间行为的相互作用。AO软件的分析研究成果指出，被横切类可能改变方面的内部状态[38]。显然，这种类对方面的反作用可能导致程序缺陷产生，我们未见文献从测试角度讨论过此类程序缺陷。另外，AOP机制提供的方面继承[8]，类似于面向对象软件中类的继承产生新的缺陷类型，其产生的缺陷类型同样值得研究。

6.3 适合于AO软件开发过程的可测表示研究

AO软件测试技术的有效应用前提是必须适合可行的AO软件开发周期模型。很多研究人员提出了不同的开发方法，一些研究将AOP应用到已有的开发周期模型上进行了尝试[44]。

然而，已有的研究对这个问题处理的基本思想都是假设AO软件开发按照面向对象软件的增量开发模型进行，将被测AO软件通过织入转换为面向对象软件之后进行测试用例生成，其缺陷是方面单元在测试用例生成之前就"消失"了。这里采用的假设是建立在非对称AOP模型之上，造成的直接困难是无法进行方面的单元测试，或者说这种情况下的方面不存在单元测试只有集成测试。另一方面，在AO软件的开发过程中，单独开发的方面、AO软件的子系统、修改后的方面等等都需要有效的测试方法和技术来判断这些不同开发阶段的产品质量是否符合预期。解决这些问题，需要进一步开展适合于AO软件开发过程的测试研究。

致谢 华中科技大学计算机科学技术系丁志俊副教授阅读了本文并提出了宝贵意见，在此表示感谢。

参考文献

[1] Filman R E, et al. 面向方面的软件开发. 莫倩，王恺，等译. 北京：机械工业出版社，2006

[2] Kiczales G, et al. An Overview of AspectJ // ECOOP'2001. Budapest, Heidelberg: Springer-Verlag, 2001

[3] Ossher H, Tarr P. Multi-Dimensional Separation of Concerns and The Hyperspace Approach // Symposium on Software Architectures and Component Technology. Heidelberg: Springer-Verlag, 2000

[4] Harrison W H, Ossher H L. Subject-oriented programming: a critique of pure objects. ACM SIGPLAN Notices, 1993, 28(10)

[5] Ossher H, Tarr P. Hyper/J™: Multi-Dimensional Separation of Concerns for Java™ // 22nd International Conference on Software Engineering (ICSE2000). Boston: IEEE Computer Society Press, 2000

[6] Harrison W H, Ossher H L, Tarr P L. Asymmetrically vs. Symmetrically Organized Paradigms for Software Composition. New York: IBM Watson Research Center, 2002

[7] Ossher H, Tarr P. Operation - Level Composition: A Case in (Join) Point [position paper] // Aspect-Oriented Programming Workshop at ECOOP'98. Heidelberg: Springer-Verlag, 1998

[8] Laddad R. AspectJ in Action: Practical Aspect-Oriented Programming. Greenwich: Manning Publications Co, 2003

(下转第277页)

件对动态环境的感知，搜集数据复制所需要的各种参数，从而作出正确的复制决策。由于反射式中间件能够适应动态变化的特征，其在数据网格中的其他领域必然也有用武之地，未来的工作中，我们将会将反射技术应用到元数据管理、数据传输等方面。

参 考 文 献

[1] Chervenak A, Kesselman C, Schuler R, et al. Design and Implementation of a Data Replication Service

[2] Teng M, Junzhou L. A prediction-based and cost-based replica replacement algorithm research and simulation. IEEE, 2005

[3] Stockinger K, Stokinger H, Duka L, et al. Access Cost Estimation for unified Grid Storage Systems. IEEE, 2005

[4] Coulson G, Blair G S, Davies N, et al. Supporting Mobile Multimedia Applications Through Adaptive Middleware. IEEE, 1999

[5] Fassino J P, Stefani J B, Lawall j, et al. THINK: A Software Framework for Component-based Operating System Kernels // Usenix Aunual Technical Conference. Monterey (USA), June 2002

（上接第 269 页）

[9] Lopes C V, Ngo T C. Unit-Testing Aspectual Behavior[Position Paper]// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[10] Mortensen M, Alexander R T. Adequate Testing of Aspect-Oriented Programs. Colorado State University: Fort Collins, 2004

[11] Mortensen M, Alexander R T. An Approach for Adequate Testing of AspectJ Programs// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[12] Lemos O A L, et al. Testing Aspect-Oriented Programming Pointcut Descriptors// 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press, 2006

[13] Ceccato M, Tonella P, Ricca F. Is AOP code easier or harder to test than OO Pcode // Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[14] Xu W, Xu D. A Model-Based Approach to Test Generation for Aspect-Oriented Programs // 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[15] Zhao J. Tool Support for Unit Testing of Aspect-Oriented Software// OOPSLA'2002 Workshop on Tools for Aspect-Oriented Software Development. New York: ACM Press, 2002

[16] Zhao J. Data-Flow-Based Unit Testing of Aspect-Oriented Programs// 27th Annual International Computer Software and Applications Conference (COMPSAC'03). Boston: IEEE Computer Society, 2003

[17] Zhou Y, Richardson D, Ziv H. Towards a Practical Approach to Test Aspect-Oriented Software // 2004 Workshop on Testing Component-Based Systems (TECOS2004). Germany: GI, 2004

[18] Lemos O A L, Maldonado J C, Masiero P C. Structural Unit Testing of AspectJ Programs // 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[19] Mackinnon T, Freeman S, Craig P. Endo-Testing: Unit Testing with Mock Objects // eXtreme Programming and Flexible Processes in Software Engineering - XP2000. Boston: Addison-Wesley, 2000

[20] Yamazaki Y, et al. A Unit Testing Framework for Aspects without Weaving// 1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[21] Massicotte P, Badri M, Badri L. Generating Aspects-Classes Integration Testing Sequences a Collaboration Diagram Based Strategy// 2005 Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05). Boston: IEEE Computer Society, 2005

[22] Badri M L, Badri, Naha M. A Use Case Driven Testing Process: Towards a Formal Approach Based on UML Collaboration Diagrams// FATES 2003. Heidelberg: Springer-Verlag, 2003

[23] Xie T, et al. Automated Test Generation for AspectJ Programs //Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. 2005

[24] Massicotte P, Badri L, Badri M. Aspects-Classes Integration Testing Strategy: An Incremental Approach. Lecture Notes in Computer Science, 2006, 3843: 158-173

[25] Souter A L, Pollock D S a L L. Testing with Respect to Concerns// International Conference on Software Maintenance(ICSM 2003). IEEE Computer Society, 2003

[26] Xu D, Xu W. State-Based Incremental Testing of Aspect-Oriented Programs // AOSD 2006. Bonn, Germany: New York: ACM Press, 2006

[27] Zhao J, Xie T, Li N. Towards Regression Test Selection for AspectJ Programs // 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press, 2006

[28] Harrold M J, et al. Regression Test Selection for Java Software // ACM Conference on Object-Oriented Programming, Systems, Languages, Applications (OOPSLA2001). New York: ACM Press, 2001

[29] Xu G. A Regression Tests Selection Technique for Aspect-Oriented Programs // 2nd Workshop on Testing Aspect-Oriented Programs (WTAOP 2006). New York: ACM Press

[30] JUnit. JUnit. 2004 [cited 2007 March]. Available at: http://www.junit.org

[31] Massol V. Junit in Action 中文版. 鲍志云，译. 北京：电子工业出版社，2005

[32] Parasoft. Jtest manuals version 4.5. 2007 [cited April]. Available at: http://www.parasoft.com/

[33] Miles R. aUnit: Unit Testing for Cross-Cutting Concerns. 2004 [cited March]. Available at: http://aunit.sourceforge.net/

[34] Xie T, et al. Detecting Redundant Unit Tests for AspectJ Programs. University of Washington Department of Computer Science and Engineering: Seattle, WA, 2004: 23

[35] Xie T, Zhao J. A Framework and Tool Supports for Generating Test Inputs of AspectJ Programs // AOSD 2006. New York: ACM Press, 2006

[36] Xie T, et al. Detecting Redundant Unit Tests for AspectJ Programs// 17th IEEE International Conference on Software Reliability Engineering (ISSRE'06). Boston: IEEE Computer Society, 2006

[37] Xie T, et al. Automated Test Generation for AspectJ Programs //1st Workshop on Testing Aspect-Oriented Programs, 4th International Conference on Aspect-Oriented Software Development. New York: ACM Press, 2005

[38] Katz S. A Survey of Verification and Static Analysis for Aspects. AOSD-Europe eu network of excellence: Israel, 2005