基于复杂网络面向对象集成测试的研究*)

李丽萍[1,2] 缪淮扣[1] 钱忠胜[1,3]

(上海大学计算机工程与科学学院 上海 200072)[1] (上海第二工业大学计算机与信息学院 上海 201209)[2]

(江西财经大学信息管理学院 南昌 330013)[3]

摘 要 软件测试是保证软件质量的重要手段。面向对象的方法给软件系统带来好处的同时,也为测试带来了挑战,传统的测试方法无法应用于许多面向对象的特性。研究表明,大型软件系统内部结构具有小世界效应(Small-World,SW)和无标度特性(Scale-Free,SF)。基于软件的复杂网络特性对面向对象的集成测试进行了研究,提出了一种通过分析类之间的交互复杂性和聚集复杂性来确定软件测试顺序的方法。利用该方法进行面向对象集成测试可以减少桩模块的数量,提高测试效率,且不降低原有测试覆盖度。

关键词 软件测试,面向对象集成测试,复杂网络,小世界效应,无标度特性

Object-oriented Integration Testing Based on Complex Networks

LI Li-ping[1,2] MIAO Huai-kou[1] QIAN Zhong-sheng[1,3]

(School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)[1]

(Computer and Information Institute, Shanghai Second Polytechnic University, Shanghai 201209, China)[2]

(School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China)[3]

Abstract Software testing is one of the key techniques to guarantee software quality. Apart from benefiting the software, object-oriented (OO) technology also brings challenges to software testing. Many OO characters can not be applied to conventional testing methods. Research results show that large software systems have small-world (SW) and scale-free (SF) characters. Based on these characters, investigated the OO integration testing. An approach was proposed to acquire the testing order via analyzing the interaction complex and clustering complex among classes. The approach can reduce the number of stubs in object-oriented integrated testing to eventually improve the testing efficiency, and keep the original test coverage.

Keywords Software testing, Object-oriented integration testing, Complex network, Small-world effect, Scale-free character

1 引言

开发出高质量、可靠的软件是所有的软件工程方法想要实现的目标。然而,随着软件复杂度越来越高,软件可靠性和软件质量的控制也相应越来越困难。软件测试是软件质量保证的重要手段,它能够有效地发现软件中的错误。据统计,在软件开发成本中,软件测试的工作量往往占软件开发总工作量的30%~40%以上[10],而对某些安全性要求高的软件,其测试成本更高。因而,提高软件测试的有效性和测试效率,降低软件开发成本,成为软件工程界亟待解决的任务之一。

在20世纪80年代末,面向对象(OO,Object Oriented)思想和方法成为软件设计与开发的主流。面向对象独有的封装、继承和多态等特征不仅改变了程序设计的风格,而且影响了软件开发的全过程。面向对象的方法使软件系统具有更强的可靠性、可维护性和可重用性的同时,也为软件的测试带来了挑战,传统的测试方法无法应用于许多面向对象的特性。

复杂网络是当前研究的热点,自然界中存在的大量复杂系统都可以通过形形色色的网络加以描述。如社会关系网[1]、电力网络、Internet[2]、交通网[2]等等。最近研究表明,大型软件系统内部结构并不是随机的,也具有小世界效应和无标度特性[3,4]。软件测试作为软件质量保证的一个重要手段,在国内外受到广泛重视并已进行了许多研究。进行面向对象测试的一个重要方面就是确定测试顺序,选择有效的测试用例以达到测试充分性准则的要求。文献[5]从对象行为的观点出发,用灰盒分析确定最重要的路径和交互,制定出测试层次;文献[6]通过基于用户用例的结构化模型将软件逐步集成;文献[7]通过分析类与类之间相互作用关系,产生出类间集成测试的顺序,使桩模块的数目最小化;文献[8]提出了基于模型的面向对象形式规格说明推导测试用例的方法和技术。这些研究为面向对象软件测试提供了许多好的方法,但是这些研究很少从面向对象软件的内部拓扑结构特点出发,忽略了大型软件系统的复杂网络特性。目前,基于复杂网络研究软件测试的工作还很少,本文是将两者结合起来,根据目前新的复杂网络研究成果,结合面向对象软件的内部拓扑结构特点,对面向对象程序中的集成测试进行研究。利用该方法进行集成测试可以减少桩模块的数量,提高测试效率,但不

*)本文的研究工作受国家高技术研究发展计划(863计划)(2007AA01Z144),国家重大基础研究(973)项目(2007CB310800)和国家自然科学基金(60673115)资助。**李丽萍** 讲师,博士生,CCF会员,研究方向为软件测试、形式化方法;**缪淮扣** 教授,博导,研究方向为软件工程、形式化方法;**钱忠胜**讲师,博士生,研究方向为软件测试、形式化方法。

降低原有覆盖度。本文的其余部分是这样安排的：第 2 节阐述了面向对象测试和复杂网络的基本概念；第 3 节分析了面向对象软件系统的网络观；第 4 节提出了一种基于复杂网络面向对象集成测试的研究方法，并重点研究了该方法在一个实例的应用；最后是结论并给出了进一步的研究方向。

2 面向对象测试与复杂网络

Smith 和 Robson 从面向对象程序的结构出发，认为面向对象的程序测试应当分为 4 个级别[9]：1)方法级，考察一个方法对数据进行的操作；2)类级，考察封装在一个类中的方法和数据之间的相互作用；3)簇级，考察一组协同操作的类之间的相互作用；4)系统级，考察由所有类和主程序构成的整个系统。面向对象方法级和系统级测试与传统测试中的单元测试和系统测试类似；面向对象类测试的实质是测试类的实例的状态和操作；簇级即面向对象集成测试。传统集成测试的主要任务是测试单元之间的接口以保证程序单元能够正确交互。面向对象程序的结构不再是传统的功能模块结构，作为一个整体，原有集成测试所要求的逐步将开发的模块搭建在一起进行测试的方法已成为不可能。面向对象的集成测试主要测试类的对象实例之间的交互作用。由于面向对象的继承、多态和动态绑定等特征使方法之间的交互以及对象与对象之间的交互变得不可预见，非常复杂。因此，面向对象的集成测试是面向对象测试整个过程中最复杂的一步[8]。

面向对象的集成测试可以分成两步进行：先进行静态测试，再进行动态测试。静态测试主要针对程序的结构进行，检测程序结构是否符合设计要求。

动态测试设计可采用基于使用的测试[10]：通过测试那些几乎不使用服务器类的类(称为独立类)而开始构造系统，在独立类测试完成后，下一层使用独立类的类，称为依赖类，被测试。这个依赖类层次的测试序列一直持续到构造完整个系统。

系统的复杂性主要决定于元素之间的交互。网络模型是描述复杂系统的最有效模型。目前大量最新研究发现：网络的拓扑结构决定着网络所拥有的特性。网络的拓扑结构指网络不依赖于结点的具体位置和边的具体形态就能表现出来的性质[12]。

20 世纪 50 年代末，数学家提出了随机网络。在随后 40 多年的时间里，随机网络一直被认为是描述真实系统最适宜的网络。直到最近几年，科学家们才发现真实世界的网络模型既不是规则网络，也不是随机网络，而是具有统计特征的网络，科学家们称其为复杂网络(complex networks)。在网络中，有两个重要的参数：平均路径长度和聚类系数。两点间的距离被定义为连接两点的最短路径所包含边的数目，把所有结点对的距离求平均，就得到了网络的平均路径长度。聚类系数是专门用来衡量网络结点的聚集情况，即网络有多紧密。研究表明，规则网络具有大的聚类系数和大的平均路径长度；随机网络具有小的聚类系数和小的平均路径长度。1998 年 Watts 和 Strogatz 提出大量真实网络都具有小世界效应(Small-World effect，SW)，即具有大的聚类系数和小的平均路径长度[1]。1999 年 Barabasi 和 Albert 指出现实世界中许多复杂网络的结点度分布服从幂律分布[2]，我们把结点度服从幂律分布的网络叫做无标度(Scale-Free，SF)网络。SF 特性刻画了复杂网络的不均匀复杂性，即大部分结点只有少数连接，而少数结点则拥有大量的连接。随后，Barabasi 和 Albert 进一步提出 BA 模型。BA 模型两个最主要的特点是增长性和偏爱性。增长性指网络的规模随着时间在不断地扩大；偏爱性指新增加的结点与网络中结点 i 的连接概率为 $P(k_i)=k_i/\sum k_j$，其中 k_i 为结点 i 的度数。偏爱性显示，BA 模型中，众多结点优先连接到度大的结点上，呈现一种“富者愈富”的现象[11]。

3 面向对象软件系统的网络观

对于大型复杂的软件系统，软件工程的方法是将复杂的问题分解为多个功能相对独立而又相互关联的简单部分。尽管传统的面向对象设计建议程序应该由许多大小和复杂度类似的小规模的类构造，然而实际的面向对象设计是由那些满足无标度特性的不规则对象(类)构造[12]。当执行时，面向对象程序产生了一个复杂的对象网络，我们可以将该网络看成是一个图，对象(类)看成结点，对象(类)之间的引用看成边，结点之间不计重复连接。每个结点(类)的度就是该类所有边的条数，指向对象的引用(Incoming Reference)称为边的入度，离开对象的引用(Outgoing Reference)称为边的出度。文献[12]对 Java，C++等的对象图进行了研究，指出结点(对象)的入度分布和出度分布都服从幂律分布。结点度服从幂律分布就是说具有某个特定度的结点数目与这个特定的度之间的关系可以用一个幂函数 $p(k)\propto k^{-\gamma}(2\leqslant\gamma\leqslant3)$ 近似地表示。幂函数曲线是一条下降相对缓慢的曲线，如果将其表示在双对数的坐标中就是一种线性关系。

度的频数 $p(k)$ 是系统中具有相同度的结点数，度的概率就是具有相同度的结点数占总结点数的比率。因此，根据度的概率便可以获得度的概率分布图。在几何的随机图中，几乎所有的结点都有数量大致一样的连接。而满足无标度特性的对象网络大多数结点只有较少的连接，只有少数结点有大量的连接。如在 JDK1.4.0 中度大于 25 的结点数只占总数的 10%，而其最大度远大于 25[13]。度较大的少数结点对应于软件系统中那些最常用的类，这些类通常被大量重用。因此，在软件系统中“富者愈富”现象归因于代码重用技术[13]。

4 基于复杂网络的面向对象集成测试

4.1 两种测试策略

由于面向对象程序没有层次的控制结构，相互调用的功能是散布在程序的不同类中，类和类之间有泛化、依赖、关联、聚合和组合等关系，类通过消息相互作用申请和提供服务。而且类的行为与它的状态密切相关，状态不仅仅是体现在类数据成员的值上，也许还包括其他类中的状态信息[9]，因此传统的集成策略在面向对象测试中没有意义。

本文用黑盒测试方法，用 UML 创建系统的结构模型，通过分析类(对象)之间的交互复杂性和聚集复杂性来确定测试的顺序，目标是用较少的测试用例达到较高的覆盖率。

(1)交互复杂性

对象图表示了程序运行时创建的对象实例以及它们之间的关系，它是面向对象程序执行时的“骨架”[12]，因为图中每个结点代表一个对象，当程序运行时对象图会不断变化。程序刚运行时只有几个对象，随着程序的运行，会创建更多的对象，也会销毁不再需要的对象[12]。图的结构(对象之间的连接)也会随之改变。对象是类的实例，类是创建对象的模板，类图在软件的整个生命周期都是存在的，因此我们以类图来分析系统的交互复杂性。类之间的交互复杂性可以通过度分

布刻画，如入度大的结点(类)说明其重用度大，而出度大的结点(类)往往比较复杂[11]。无标度网络一个特别有用的方面是它的抗损坏的鲁棒性和脆弱性。因为大多数结点与其它结点有较少的连接，摧毁它们对其它的结点没有什么影响。另一方面，一小部分 HUB 结点(类)被高度连接，摧毁它们的灾难性是巨大的。基于软件复杂网络的这个特点，本文提出了如下的测试方法：

a)将系统的 UML 类图表示为一个软件网络 $G=(V,E)$，其中 G 是一个连通有向图，V 是结点集，代表类；E 是边集，代表类之间的关系。

b)遍历整个软件网络，统计每个结点(类)的依赖类和它的入度和出度并且记载在一个四元组(*C_Test*，*C_Dep*，*In_Degree*，*Out_Degree*)中，*C_Test* 代表要统计的类，*C_Dep* 代表该统计类所依赖的类集，*In_Degree* 代表指向该类的引用，*Out_Degree* 代表离开该类的引用。

c)首先集中测试 *In_Degree* 大而 *Out_Degree* 为 0 的结点(类)，这些类是重用度大而又不依赖于其他类的独立类，先测重用度大的独立类可以在保证测试覆盖率的同时减少桩模块的数量。

d)接着测试使用该类的依赖类，依赖类 *C_Dep* 是一个类集，原则上可以并行测试，但是我们在并行测试依赖类时同样优先选择测试 *In_Degree* 大而 *Out_Degree* 较小的类，并且每一次测试最多增加一个新类，沿着这个依赖类层次的测试序列重复执行步骤(d)，使先测试的类优先于后测试的类，一直持续到构造完整个系统。通过先测试这些 HUB 类，接着再处理依赖它的类。这样我们可以优化测试用例，减少测试工作量。

该方法的困难之处在于实际测试时要了解网络的全局信息，而对于一个大型复杂的面向对象软件系统，由于继承、多态和动态绑定等因素使对象之间的交互异常复杂，因此要全面地分析整个系统类之间的依赖关系比较麻烦。所以我们从大型软件系统具有的聚集复杂性的特征提出了第二种测试策略。

(2)聚集复杂性

面向对象的方法习惯于将语义相近又倾向一起变化的类和接口放在一个单元里，如 Java 的包、C# 的命名空间，这种现象对应于网络的集团结构。网络的集团结构是指节点之间的连接程度不同而形成的“抱团”结构[11]。研究表明，在集团的内部，结点之间的连接程度明显高于不同集团之间的结点之间的连接程度，即表现了“物以类聚，人以群分”的特性。在面向对象软件系统中，包形成了高内聚、低耦合的良好结构。因此在进行面向对象的集成测试时，我们可先以集团(包)为测试单位，在集团的内部随机选出比率为 P 的结点(类)，再对每一个被选出的结点(类)，随机选其中一个邻居结点进行测试。因为度较大的结点有更多的结点与之相连，找到它的可能性也较大。这样一来，我们可以在不了解网络全局信息的同时找到度较大的结点(类)，然后在集团之间再采用相同的测试策略。

同时，我们还可以综合以上两种测试策略，以集团(包)为测试单位，在集团的内部先集中测试入度大而出度为零的独立类，接着测试使用该类的依赖类，沿着这个依赖类层次的测试序列一直持续到集团内所有的类，然后在集团之间再采用相同的测试策略，直到构造完整个系统。

4.2 实例研究

Kung[7] 等人在研究面向对象测试时提出了对象关系图(ORD)。一个 ORD 图显示了类之间的各种关系，包括继承、聚合、关联、多态和动态绑定等，ORD 是一种非常方便的测试工具。我们的方法直接利用软件系统的 UML 类图，类图既具备与 ORD 图类似的功能，又可以直接从系统的规格说明中得到，节省了转换的时间。

假定有如图 1 所示的类图，类 A 与类 C、类 D 是聚合关系，其中类 A 是部分，类 C、类 D 是整体，整体依赖于部分；类 G 与类 D 是泛化关系，D 是基类，G 是子类；类 C 与类 B 是依赖关系，C 依赖于 B；类 B 与类 K、类 L、类 M 是泛化关系，类 K、类 L、类 M 是类 B 的子类；类 B、类 S 与类 T 是关联关系，类 B、类 S 依赖于类 T；类 T 与类 F 也是关联关系，类 T 还依赖于类 F；类 F、类 E 与类 A 是关联关系，类 F、类 E 依赖于类 A，同时类 E 还依赖于类 F。下面根据我们提出的第一种测试策略对该类图进行测试顺序研究。

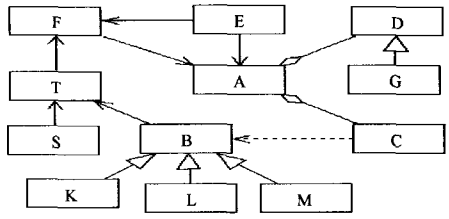

图 1　系统类图

遍历整个类图网络，首先得出每个类的四元组(*C_Test*，*C_Dep*，*In_Degree*，*Out_Degree*)，如表 1 所示。

表 1　各个类的四元组表示

C_Test	C_Dep	In_Degree	Out_Degree
A	Φ	4	0
F	{A}	2	1
E	{A,F}	0	2
C	{A,B}	0	2
D	{A}	1	1
T	{B,S}	2	1
B	{C,K,L,M}	4	1
S	Φ	0	1
G	Φ	0	1
K	Φ	0	1
L	Φ	0	1
M	Φ	0	1

从表 1 可知，类 A 是 *In_Degree* 大而 *Out_Degree* 为 0 的独立类，应该最先测试；类 C、类 D、类 E、类 F 依赖于类 A，但是类 C 还需依赖于类 B，所以类 C 还不能测试；类 D、类 E、类 F 中，类 F 的 *In_Degree* 最大，而且类 E 依赖于类 F，根据策略一，我们先测 F，再测 D，其次测 E；再测 F 的依赖类 T、类 D 的依赖类 G；T 的依赖类是 B 和 S，由于类 B 的 *In_Degree* 大于类 S 的 *In_Degree*，因此先测 B，再测 S；类 C，类 K，类 L，类 M 依赖于 B，且 *In_Degree* 都相同，可以并行测试。于是我们得到的测试顺序是{A->F->D->E->T->G->B->S->{C,K,L,M}}。

该类图的无标度特性不是特别显著，而在实际应用中，大型软件系统的无标度特性非常明显，即只有少数结点(类)拥有大量的连接，大部分结点(类)只有少数连接，并且软件系统具有增长性，新增加的结点(类)具有偏爱性，优先连接到度大的结点上。先测这些 HUB 结点，接着测试依赖于它的类，有利于我们尽早查出错误，同时减少所需设计桩模块的数量，提高测试效率，减少测试工作量。

结束语　软件测试已成为软件开发过程中必不可少的质量保障手段。面向对象的特征使得传统的软件测试方法无法

适用于面向对象软件的测试。最近研究表明，面向对象软件系统内部结构具有小世界(SW)效应和无标度(SF)特性，其中代码重用技术引起了“富者愈富”现象的产生。根据面向对象软件系统内部拓扑结构这一特点，本文从软件系统复杂网络的特性出发，对面向对象的集成测试进行了研究。提出通过分析类之间的交互复杂性和聚集复杂性来确定软件测试顺序的两种不同的测试策略，其中重点分析了基于交互复杂性的第一种测试策略的使用，而对于基于聚集复杂性的第二种测试策略将是我们下一步重点研究的内容。目前，基于复杂网络研究软件测试的工作还很少，以往的对面向对象软件测试的研究大多忽略了面向对象软件复杂网络的特性。本文的创新之处是将两者结合起来，在目前新的复杂网络研究成果的基础上，对面向对象软件系统的集成测试进行研究，目标是在提高软件可靠性的同时减少测试的工作量，提高测试工作效率。

参 考 文 献

[1] Watts J S, Strogatz S H. Collective dynamic of small-world networks. Nature, 1998, 393(6684): 440-442

[2] Batabasi A L, Albert R. Emergence of scaling in random networks [J]. Science, 1999, 286: 509-512

[3] Sole' R V, Ferrer R, Montoya J M, et al. Tinkering and mergence in complex networks [J]. Complexity, 2002, 8 (1): 20-33

[4] Valverde S, SoléR. Hierarchical Small-worlds in Software Architecture [R]. Santa Fe Institute working paper, SFI/ 03-07-044. 2003

[5] Jorgensen P C, Erickson C. Object-oriented Integration Testing. Communications of the ACM, 1994, 37(9): 30-38

[6] Mc Gregor J. Let's Don't and Say We Did. Journal of Object-oriented. Programming, 1998, 11(5): 6-11, 14

[7] Kung D, Gao J, Hsia P, et al. Class Firewall, test order and regression testing of object-oriented programs. Journal of Object-Oriented Programming, 1995, 8(2): 51-65

[8] 刘玲. 基于面向对象形式规格说明的测试用例生成技术. 博士学位论文. 上海大学, 2004

[9] 金凌紫. 面向对象软件测试技术进展[J]. 计算机研究与发展, 1998, 35(1): 6-13

[10] Pressman R S. 软件工程—实践者的研究方法. 第5版. 梅宏, 译. 机械工业出版社, 2002

[11] 李兵, 王浩, 李增扬, 等. 基于复杂网络的软件复杂性度量研究. 电子学报, 2006: 2371-2375

[12] Potanin A, Noble J, Frean M, et al. Scale-free geometry in object-oriented programs [J]. Communications of the ACM, 2005, 48(5): 99-103

[13] 闫栋, 祁国宁. 大规模软件系统的无标度特性与演化模型. 物理学报, 2006, 55(8): 2800-2804

(上接第 249 页)

统则很好地解决了这一问题。

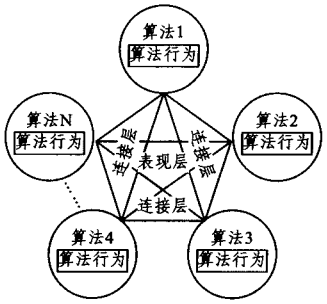

图 4 事件驱动的算法演示系统

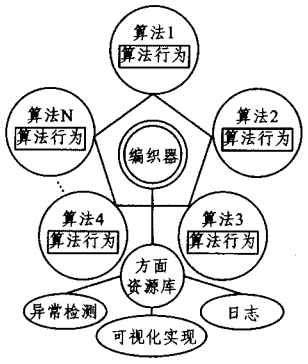

图 5 面向方面的算法演示系统

(2)可扩展性强，便于定义感兴趣事件

如果需要新增算法，只需将直接算法代码添加到系统中即可。对于感兴趣事件，即待可视化的事件，可通过在关注点处使用预先约定好的函数(Operation 类成员函数)，而这些函数的使用完全不影响算法本身的结构和可独立运行的能力，这种替换是非侵入性的。如果要修改感兴趣事件，也只需对 Operation 类进行改变即可。

(3)提高代码可重用性

通过 AOP 对关注点的拦截，系统可以有效地实现代码重用。在排序类算法的演示中通常需要对数组元素的交换操作作可视化，通过在方面中写入可视化代码，就可以轻易实现所有对应关注点上的可视化。

结束语 本文介绍了一种使用面向方面技术来实现算法演示系统的方法。使用面向方面技术，克服了传统事件驱动的算法演示系统存在的代码混乱、可维护性和可扩展性低的问题。相比较而言，本文介绍的 AOP 方案更为清晰灵活，各模块间耦合度大大降低，提高了算法演示系统的可维护性、可扩展性以及可重用性。

我们下一步将针对面向方面技术的特点，将上述算法演示系统的分层模型进一步完善，构建一个面向方面的算法演示系统通用框架，并开发一些中间件，支持各种算法的演示。

参 考 文 献

[1] Kerren A, Stasko J T. Algorithm Animation - Introduction[C] //Diehl S, ed. Software Visualization, LNCS. Springer-Verlag, 2002, 2269: 1-15

[2] 黄永红, 黄卫红, 符开耀, 等. 开放式算法动态演示平台模型[J]. 长沙电力学院学报, 2004, 19(1): 23-25

[3] Demetrescu C, Finocchi I, Stasko J T. Specifying Algorithm Visualizations: Interesting Events or State Mapping[C]// Diehl S, ed. Software Visualization, LNCS. Springer-Verlag, 2001, 2269: 16-30

[4] Kiczales G, et al. Aspect-Oriented Programming[J]// Aksit M, Matsuoka S, eds. ECOOP'97, LNCS. 1997, 1241: 220-242

[5] Gradecki J D, Lesiecki N. 精通 AspectJ[M]. 王欣轩, 吴东升, 等译. 清华大学出版社, 2005

[6] Colyer A, Clement A. Eclipse AspectJ 中文版[M]. 钱竹青, 邹正武, 等译. 清华大学出版社, 2006

[7] Masuhara H, Kiczales G. Modeling Crosscutting in Aspect-Oriented Mechanisms[J]// Cardelli L, ed. ECOO P2003, LNCS. 2003, 2743: 2-28

[8] The AspectJ Team. The AspectJ Programming Guide. Palo Alto Research Center, 2003. http://www.eclipse.org/aspectj/doc/released/progguide/index.html