用概念格方法挖掘 Aspect *)

何丽莉[1] 白洪涛[2] 张家晨[1]

(吉林大学计算机科学与技术学院 长春 130012)[1] (深圳电信新技术开发中心 深圳 518048)[2]

摘 要 横切关注引起的代码散布和代码交织现象是面向对象编程技术自身的局限性造成的,面向方面的编程(AOP)技术有望成为解决该问题的有效途径。本文以改善程序代码的横切现象为目标,把面向对象遗产系统中的方法作为实例,方法间的直接调用关系作为实例的特征,使用概念格技术挖掘候选 Aspect 集。候选 Aspect 集较好地反映了遗产系统中横切关注点的行为结构,为 Aspect 重构提供有用的帮助。本文使用一个银行应用系统的实际例子进行了实验,说明了概念格方法的可用性。

关键词 横切关注点,概念格,Aspect

Aspect Mining Using Concept Analysis

HE Li-Li[1] BAI Hong-Tao[2] ZHANG Jia-Chen[1]

(College of Computer Science and Technology, Jilin University, Changchun 130012)[1]

(New Technology Center, Shenzhen Telecom, Shenzhen 518048)[2]

Abstract Crosscutting concern is the inherent limitation for object oriented programming, Aspect oriented programming is hopeful to become an effective way of solving the problem. A novel method using concept analysis technology to identify candidate Aspects set from object oriented legacy system is provided in this paper. Methods in legacy system are regarded as items and direct method invocation relationships are as features. Candidate Aspects are generated automatically after concept clustering. Candidate Aspects reflect the behavior of the crosscutting concerns in the legacy system, which is useful for the following Aspect reflecting and program understanding. A actual example of bank program proved the usability of concept analysis.

Keywords Crosscutting concerns, Concept lattice, Aspect

1 介绍

关注分离是软件开发过程中的一个重要的原则。在现代应用开发过程中,恰当地运用这一原则才能够适应不断出现的新的关注。传统的方法学只提供一维分解,只能为核心关注点提供较好的封装,还有许多其它关注点由于无法被该分解方式封装,被迫散布在整个软件内部,与核心关注点交杂在一起,通常称它们为"横切关注点"(crosscutting concerns)。Tzilla 等人认为横切关注点是大多数复杂软件系统固有的,无论是采用面向对象语言还是过程式语言,都难以用清晰、模块化的代码实现这种横切关注点[1],最后的结果经常是:实现这些设计决策的代码分布横切于整个系统的基本功能代码中,形成了"代码散布"(code scattering)和"代码交织"(code tangling)现象。

面向方面的程序设计(Aspect Oriented Programming: AOP)有望成为解决上述问题的重要方法。AOP 建立在现有技术的基础上,为系统横切关注点的实现提供了一种新的机制。如 AspectJ 语言就是建立在 Java 语言基础上的,并提供了一种新的模块化机制 Aspect。Aspect 能够将同时作用于多个过程、模块或对象的关注封装在一个新的模块中而不破坏原有程序的结构[2]。基于 AOP 的编程语言和开发工具大量地出现,并受到越来越多的专业人士的关注,应用的范围日渐扩大。这样,如何从已有面向对象系统中寻找并封装横切的关注点,即如何将已有的面向对象系统转换成为面向方面系统也就成一个亟待解决的问题。

从面向对象系统到面向方面系统的转换需要如下的两个步骤:(1)横切关注的识别:从源代码中识别出候选的横切关注点;(2)重构成 Aspect:将与横切关注相关的代码拆分出来并构造成 Aspect 的形式。对横切关注的识别是转换进行的关键。

一个横切关注点常常在系统中表现为结构上或者行为上的相似。有的研究从程序的静态分序入手,根据"来自同一关注的行为和属性往往具有相同的命名规则"这一原则提出了基于文本的和基于类型的 Aspect 搜索方法。在遗产系统遵循良好的命名规范的情况下,使用文本匹配的技术能够获得较好的效果[3,4];而当遗产系统未能遵循良好的命名规范,并且系统中存在许多不同类型的对象具有相似的名称时使用基于类型的查询效果更佳[3]。还有的研究从动态分析的角度,根据"来自同一关注的行为具有类似的调用模式"的原则,通过对程序跟踪产生的执行模式进行分析挖掘横切关注。

本文将"直接方法调用"关系作为系统行为相同的度量,采用程序静态分析与概念格技术相结合的方法从面向对象的遗产系统识别横切关注点,给出候选 Aspect 集。即如果方法 $d_1,d_2,\cdots,d_i$ 同时被方法 $b_1,b_2,\cdots,b_l$ 中的全部或绝大多数调用,则说明方法 $d_1,d_2,\cdots,d_i$ 散布在了 $b_1,b_2,\cdots,b_l$ 中,从被 $b_1,b_2,\cdots,b_l$ 调用的视角,方法 $d_1,d_2,\cdots,d_i$ 可能成为一个横切关注,即它们组成了一个候选 Aspect。如经过对代码的考察,确认方法 $d_1,d_2,\cdots,d_i$ 是横切关注点,则 $d_1,d_2,\cdots,d_i$ 用一个 Aspect 来封装是非常合适的。

*)国家自然科学基金项目(69903005),吉林大学创新基金。何丽莉 讲师,主要研究领域为软件工程、数据挖掘。

2 挖掘过程

数据库知识发现的过程就是将数据库中蕴含的知识形式化成有用概念的过程。概念格，也称为 Galois 格，提供了一种支持该过程的有效工具。

概念格的每个节点是一个形式概念，由两部分组成：外延，即概念所覆盖的实例，和内涵，概念的描述，即该概念覆盖实例的共同特征。概念格通过 Hasse 图生动和简洁地体现了这些概念之间的泛化和特化关系。因此，概念格被认为是进行数据分析的有力工具。从数据集(概念格中称为形式背景)中生成概念格的过程实质上是一种概念聚类过程；概念格可以用于许多机器学习的任务。目前，已经有了一些建造概念格的算法，并且概念格在信息检索、数字图书馆、软件工程和知识发现等方面得到应用。

假设给定形式背景(context)为三元组 $T=(O,D,R)$，其中 O 是实例集合，D 是描述符(特征)集合，R 是 O 和 D 之间的一个二元关系，则存在唯一的一个偏序集合与之对应，并且这个偏序集合产生一种格结构，这种由背景(O,D,R)所诱导的格 L 就称为一个概念格。格 L 中的每个节点是一个序偶(称为概念)，记为(X,Y)，其中 $X\in P(O)$称为概念的外延；$Y\in P(D)$称为概念的内涵。每一个序偶关于关系 R 是完备的，即有性质：

1)$X=a(Y)=\{x\in O|\forall y\in Y,xRy\}$;2)$Y=\beta(X)=\{y\in D|\forall x\in X,xRy\}$。

对大量的 OOP 代码研究发现，有些横切关注点在程序中表现为重复出现的方法调用。由此猜想，可以从 OOP 程序方法静态调用关系出发找到这一类横切关注。那么，可以将 OOP 程序中的方法作为实例，方法间的直接调用关系作为行为的特征。Aspect 挖掘可分为两个步骤。

步骤 1：定义实例。根据前面的假定：在面向对象的遗产系统中，很多横切关注都表现为方法的散布，那么如果一些方法频繁出现在很多其他实现系统业务功能的方法调用中，就预示可能存在一个横切关注点。所以，我们把系统中每一个的方法作为一个实例，该方法是否是一个横切关注点是由它的静态特征所决定的，即该方法被其他哪些方法所调用了。假设系统中方法的数量为 m，则有 m 个待分析实例；每一个实例都是一个 m 维向量。任意实例 o 是这样构成的，如果实例 $k(k=1,\cdots,m)$调用了实例 o，则实例 o 的第 k 维的特征值为 1，否则为 0，由此定义出所有实例及其特征，即实例的数据矩阵。

步骤 2：运行算法，生成格结构。给定了实例和其特征间的二进制关系，概念分析技术能够把实例及特征自动组织成一个洞察源代码关系的格结构，每个节点代表共享特征的最多实例集合，较高层次的概念代表很多实例共有特定特征，而较低层次的概念代表特定实例共有很多特征。因而，在本文中，较低层次的概念就是可能存在的横切关注点，该实例(方法)集合可以作为一个候选 Aspect。概念分析技术在软件工程领域，如遗传代码重构[7]，程序分片[8]，逆向工程[9]以及特征定位[10]等都有广泛的应用。步骤 2 最终生成反映源代码中方法之间调用关系的概念格结构。

3 挖掘 Aspect 示例

以下是一个用 OOP 技术开发的银行系统的例子，具有存款、取款、查帐户余额、查历史交易明细等功能。由类 Account 来实现，辅助功能有连接数据库、本地日志记录等。每一个方法作为一个实例，类的构造函数被忽略。

```
public class Account {
    private float _balance;
    private int _accountNumber;
    private int _password;
    ……
    public Account(int accountNumber,int password){
        _accountNumber = accountNumber;
        checkpassword(_password);
        …
    }
    public void credit(float amount){
        db. connect();
        Update(amount);
        _logger. logp(Level. INFO,"credit","amount");
        db. close();
    }
    public void debit(float amount)
        db. connect();
        float balance = getBalance();
        if (balance < amount){
            Error. ("Total balance not sufficient");
        } else {
            Update(balance-amount);
            _logger. logp(Level. INFO,"debit","balance-amount
            ");
        }
        db. close();
    }
    public float queryremainings(){
        float Remain;
        db. connect();
        Remain=Selectremain();
        db. close();
        _logger. logp(Level. INFO,"queryremainings","remainings
        ");
        return Remain;
    }
    public string querylogs(){
        string Transtr;
        db. connect();
        Transtr=Searchtran();
        db. close();
        return Transtr;
    }
    ……
};
```

根据上述例子，考虑方法之间的调用与否的关系，12 个方法作为 12 个实例，实例的数据矩阵如表 1 所示，每一行就是一个实例，每一列就是实例的一个特征。

表 1 实例数据矩阵

	credit	debit	Qure	Qulo	Conn	close	Update	logp	getBal	Sere	Sech	Error
Credit(O1)	0	0	0	0	0	0	0	0	0	0	0	0
Debit (O2)	0	0	0	0	0	0	0	0	0	0	0	0
Qure (O3)	0	0	0	0	0	0	0	0	0	0	0	0
Qulo (O4)	0	0	0	0	0	0	0	0	0	0	0	0
Conn (O5)	1	1	1	1	0	0	0	0	0	0	0	0
Close (O6)	1	1	1	1	0	0	0	0	0	0	0	0
Update(O7)	1	1	0	0	0	0	0	0	0	0	0	0
Logp (O8)	1	1	1	0	0	0	0	0	0	0	0	0
GetBal (O9)	0	1	0	0	0	0	0	0	0	0	0	0
Sere (O10)	0	0	1	0	0	0	0	0	0	0	0	0
Sech (O11)	0	0	0	1	0	0	0	0	0	0	0	0
Error (O12)	0	1	0	0	0	0	0	0	0	0	0	0

在这里第 i 行和第 j 列的交点的值是 $d(i,j)$，是实例 O_i 和实例 O_j 的相异性的量化表示，它是一个非负的值，当实例 i 和实例 j 越相似或"接近"，其值越接近 0；两个实例越不同，其值越大。$d(i,j)=d(j,i)$ 且 $d(i,i)=0$。

根据表 1 所给出的实例，可以形成如表 2 所示的概念集。

表 2 概念集

Name	Extent	Intent
TOP	{credit, debit, Qure, Qulo, conn, close, Update, logp, getBal, Sere, Sech, Error}	∅
Concept1	{conn, close, Update, logp, getBal, Error}	{C2}
Concept2	{conn, close, Sere}	{C3}
Concept3	{conn, close, Sech}	{C4}
Concept4	{conn, close, Update, logp}	{C1, C2}
Concept5	{conn, close, logp}	{C1, C2, C3}
Concept6	{conn, close}	{C1, C2, C3, C4}
BOT	∅	{C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12}

由于概念格的完备性，即使对于适当大小的数据也将会产生庞大的格结构，它的构造过程无疑是非常耗时的。人们对此进行了专门的研究，已提出了若干种有效的算法来生成概念格[11]。本文的例子所生成的概念格结构如图 1 所示。

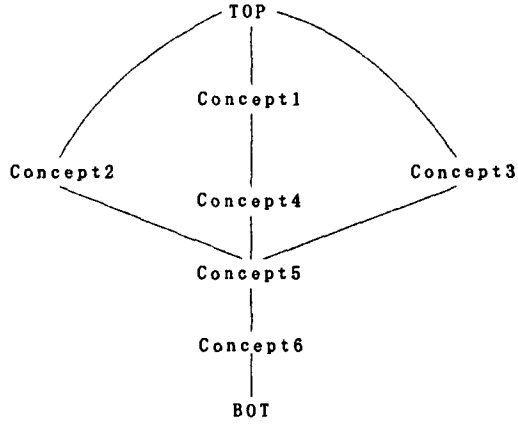

图 1 概念格

最终生成的概念格表达了一系列的概念，比如 Concept6 表示了 conn 和 close 方法同时散布在 credit, debit, Qure, Qulo 四个方法中，这清晰地表达了 conn 和 close 的横切特性，conn 和 close 可以封装在一个 Aspect 中。

在挖掘到的候选 Aspects 中也可能包含无意义的类，如 getBal 和 Error 从程序的角度不能作为一个关注的内容封装。从结果类中得到恰当的 Aspect 还需要根据系统的实际情况，在人的参与下进行，这部分的工作还有待进一步探讨。

本文采用的方法能够发现具有不同名称而经常一起出现的方法集合。与单纯的基于文本匹配或者类型匹配的方法相比，该方法突破了重复的行为必须采用相近似的命名规则的限制，从行为角度发掘出了横切关注点。

结束语 目前，大多数的挖掘 Aspect 的方法都是基于程序的静态分析和动态分析上，并且其主要工作是找到横切关注点。本文将概念格技术用于候选横切关注点的挖掘，将源代码中行为相同的方法集生成一个概念层次结构，为面向 Aspect 的程序转换的第二步——Aspect 重构提供了有用的帮助。本文使用一个银行应用系统的实际例子进行了实验，说明了概念格方法的可用性。对于大规模应用程序的 Aspect 自动重构，生成的格结构会非常复杂，因而截取哪个层次的概念作为候选的 Aspect 集是一个难点，解决方法之一是加入一些辅助的约束条件，大规模应用程序的验证是下一步的工作。

参 考 文 献

1 Elrad T, Filman R E, Bader A. Aspect-Oriented Programming. Communication of the ACM, 2001, 44(10)
2 Elrad T, Aksit M M, Kiczales G, et al. Discussing Aspects of AOP. Communication of the ACM, 2001, 44(10)
3 Hannemann J, Kiczales G. Overcoming the Prevalent Decomposition of Legacy Code. In: Workshop on Advanced Separation of Concerns, 2001
4 Griswold W G, Kato Y, Yuan J J. Aspect Browser: Tool Support for Managing Dispersed Aspects: [Technical Report CS99-0640]. Department of Computer Science and Engineering, University of California, San Diego, 1999
5 Breu S, Krinke J. Aspect Mining Using Event Traces. In: Automated Software Engineering Conference, Linz, Austria, 2004. 310～315
6 Ganter B, Wille R. Formal Concept Analysis. Springer Verlag, Berlin, Heidelberg, New York, 1996
7 Tonella P. Concept analysis for module restructuring. IEEE Trans. on Software Engineering, 2001, 27(4): 351～363
8 Tonella P. Using a concept lattice of decomposition slices for program understanding and impact analysis. IEEE Trans. on Software Engineering, 2003, 29(6): 495～509
9 Snelting G, Tip F. Reengineering class hierarchies using concept analysis. ACM Trans. on Programming Languages and Systems, 2000, 22(3): 540～582
10 Eisenbarth T, Koschke R, Simon D. Locating features in source code. IEEE Trans. on Software Engineering, 2003, 29(3): 195～209
11 Tilley T, Cole R, Becker P, et al. A Survey of Formal Concept Analysis Support for Software Engineering Activities. In: Proc. 1st Intl. Conf. on Formal Concept Analysis, 2003

(上接第 144 页)

(Team Work)的所有感知要素，从而有效地支持了用户在移动中完成与他人的协同工作。

模型未考虑安全性以及隐私方面的问题，同时定义不够精细，将在将来的工作中加以完善。

参 考 文 献

1 Dourish P, Portholes S B. Supporting Awareness in a distributed work group. In: ACM CHI'92 Conf. on Human Factors in Computing Systems, Monterey (CA), 1992
2 Ferscha A. Workspace Awareness in mobile virtual teams. In: Proc. of the 9th Intl. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2000), IEEE Computer Society Press, 2000
3 Christein H, Schulthess P. A General Purpose Model for Presence Awareness. Distributed Communities on the Web. In: 4th Intl. Workshop, DCW 2002, Sydney, Australia, Springer-Verlag Heidelberg, Jan. 2002
4 Carroll J M, Neale D C, Isenhour P L. Notification and awareness: synchronizing task-oriented collaborative activity. Intl Journal of Human-Computer Studie, 2003, 58(5): 605～632
5 Kling R. Cooperation, coordination and control in computer-supported work. Communications of the ACM, 1991, 34(12): 83～88
6 Workshop at ECSCW'03. Applying Activity Theory to CSCW research and practice. http://www.uku.fi/tike/actad/ecscw2003-at
7 Steinfield C, Jang C, Pfaff B. Supporting virtual team collaboration: The TeamSCOPE system. In: the Proc. of the Group99 Conf. Phoenix, Nov. 1999
8 Silva F R S. Awareness and Privacy in Mobile Wearable Computers. ICS234A-Virtual Collocation (Fall 2000)
9 RFC 2778. A Model for Presence and Instant Messaging. http://www.ietf.org/rfc/rfc2778.txt