第 34 卷 第 6 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.6 Computer Engineering March 2008
·博士论文· 文章编号：1000—3428(2008)06—0043—02 文献标识码：A 中图分类号：TP311

面向方面的自动化重构方法

曲立平，刘大昕，杨 静
（哈尔滨工程大学计算机科学与技术学院，哈尔滨 150001）

摘 要：提出一种将面向对象程序重构为面向方面范型的自动化方法。该方法分挖掘和重构两个阶段进行。第一阶段挖掘面向对象程序中潜在的横切关注点，第二阶段将横切关注点重构为方面。给出一个约 13 000 行代码的程序自动重构的评价结果。
关键词：横切关注点；方面抽取；重构

Automated Refactoring of Aspect-oriented Software

QU Li-ping, LIU Da-xin, YANG Jing
(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001)

【Abstract】 This paper proposes an automated approach to refactoring object oriented programs to the aspect-oriented paradigm. The approach is based upon the application of two steps: mining and refactoring. After mining potential crosscutting concerns, the refactoring phase transforms crosscutting concerns to a new aspect. This paper presents the results of an evaluation in which about 13000 LoC program is refactored.
【Key words】 crosscutting concerns; aspect extraction; refactoring

面向对象技术实现的系统中，存在着无法模块化的关注点，如日志、错误处理、同步、权限检查等，这些关注点横跨软件系统的基本构件，通常称它们为横切关注点(crosscutting concern)。横切关注点存在的实现代码是重复的或散布在整个软件系统中，形成了代码散布和代码缠结现象。代码散布和代码缠结现象违背了软件设计中关注点分离的基本原则，分散了构件原来假定要做的事情，使得源代码难以理解、维护和进化。为了克服这种普遍存在的现象，面向方面范型(Aspect-Oriented Programming, AOP)提供了一种新的语言结构——方面(aspect)，允许横切关注点封装在aspect中，并提供抽象和组合基本构件和aspect成为全部系统的机制[1]。经过从概念提出至今的近十年时间，AOP开始被工业界广泛采用，同时也提出了将其应用到工业界大量存在的遗产系统的需求。

1 相关工作

将面向对象遗产系统移植到 AOP 的研究是近几年才开始的。文献[2]人工地将 JHotDraw 软件中 Undo 横切关注点重构为 aspect。文献[3]提出了 28 种针对 AspectJ 语言的重构方法，并首次提出了针对 AOP 的代码味道。文献[4]提出了 AOP 重构过程中应遵循的 33 条规则。以上方法都有一个根本的问题：没有研究面向方面的自动化重构方法。文献[5]提出了一种切入点描述语言，对切入点进行较高抽象级别的管理。该方法研究的问题并不是本文主要关心的问题，但是它也是本文方法的有益补充。

2 自动化重构方法

将面向对象系统重构为 AOP 就是在面向对象系统中找到横切关注点，然后在不改变系统外在行为的条件下将其转化为 aspect。因此，自动化重构过程分两个阶段进行：挖掘和重构。本文假设面向对象系统已经进行了 Extract Method 重构。

2.1 挖掘

挖掘阶段的主要任务是在源代码中识别横切关注点。由于面向方面系统已经进行了 Extract Method 重构，因此横切关注点主要表现为方法调用，如日志、权限验证等。本阶段分 2 个步骤进行横切关注点识别。

步骤 1 获取相同调用模式的横切关注点

相同调用模式的横切关注点的症状表现为在不同的上下文中出现。采用文献[6]描述方法可获得这部分横切关注点，记作 CR_{same}。$CR_{same}=\{v|v<rel>u\in R^{<rel>},\ u,v\in N_P\}$，其中 $R^{<rel>}=\{v<rel>u|<rel>u\in U^{<rel>}\wedge v<rel>u$ 满足横切约束，$u,v\in N_P\}$。

步骤 2 获取不同调用模式的横切关注点

不同调用模式的横切关注点的症状表现为被不同的方法调用。形式化概念分析(FCA)可以识别这种症状的横切关注点。

给定上下文 $C(O, A, R)$，其中 O 是对象的有限集合，A 是属性集合，R 是对象集合 O 和属性集合 A 上的二元关系，$R\in O\times A$，概念 C 被定义为集合对(X, Y)且满足：

$$X=\{u|u\in O\wedge\forall v(v\in Y\wedge<u,v>\in R)\} \quad (1)$$

$$Y=\{v|v\in A\wedge\forall u(u\in X\wedge<u,v>\in R)\} \quad (2)$$

令$O=N_P$, $A=N_P$, $R=\{<u,v>|\ u\in O\wedge v\in A\wedge\ v\in lefttree(u)\wedge\forall w((w\in N_P\wedge v\in lefttree(w))\rightarrow(distance(v,u)\leq distance(v,w)))\}$，$lefttree(w)$表示方法调用树中$w$的左子树；$distance(v,u)$表示方法调用树中$v$与$u$之间的距离(路径长度)，使用FCA即可获得不同调用模式的横切关注点，记作CR_{diff}。$CR_{diff}=\{v\mid v\in Y\wedge(X,Y)\in CON_{diff}\}$，$CON_{diff}$为FCA产生的不同调用模式的横切关注点

基金项目：国家自然科学基金资助项目(60673131)
作者简介：曲立平(1973－)，女，博士研究生，主研方向：软件工程，数据挖掘；刘大昕，教授、博士生导师；杨 静，教授、博士
收稿日期：2007-03-25 **E-mail：**quliping@hrbeu.edu.cn

的概念，$CON_{diff}=\{(X,Y)\mid(\forall u\forall v\exists X\exists Y((v<rel>u\in R^{<rel>}\wedge<rel>\in\{\triangledown,\triangle\}\wedge(X,Y)\in CON\wedge u\in X\wedge v\in Y)\rightarrow X=X-\{u\})\vee\forall u\forall v\exists X\exists Y\exists o((v<rel>u\in R^{<rel>}\wedge<rel>\in\{\rightarrow,\leftarrow\}\wedge(X,Y)\in CON\wedge v\in Y\wedge o\in X\wedge u\in lefttree(o)\wedge v\in lefttree(o)\wedge\forall w(w\in N_P\wedge u\in lefttree(w)\wedge v\in lefttree(w)\rightarrow((distance(u,o)\leq distance(u,w))\wedge(distance(v,o)\leq distance(v,w)))))\rightarrow X=X-\{o\}))\wedge|X|\geq 1\}$，$CON=\{(X,Y)\mid(X,Y)$是由 FCA 产生的概念$\wedge|X|\geq 2\}$。

2.2 重构

重构阶段的主要任务是将横切关注点转化为 aspect。该阶段分 3 个步骤进行：

步骤 1 Statement Reordering 重构

为了使横切关注点封装为aspect成为可能，对任意$v\in CR_{same}\cup CR_{diff}$，需要验证$v$与联结点之间是否存在语句。如果存在语句，需要对$v$进行面向对象的Statement Reordering重构。

步骤 2 确定重构方式

针对 AspectJ 语言，本文定义了 6 种自动化重构方式，分别为 bef-exec, aft-exec, bef-call, aft-call, aro-bef-exec 和 aro-aft-exec。

定义 1 如果横切关注点v使得$((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\triangledown u\in R^{\triangledown}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge u\in X\wedge v\triangledown u\in S^{\triangledown})))\wedge\exists s(s\in D_c\wedge s='seq'\wedge s\in lefttree(u)\wedge v\in lefttree(s))$成立，称$v$满足bef-exec重构方式。其中，$D_c$表示方法调用树中控制结点的集合，$seq$表示方法调用树中顺序控制结点。

定义 2 如果横切关注点v使得 $((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\triangle u\in R^{\triangle}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge u\in X\wedge v\triangle u\in S^{\triangle})))\wedge\exists s(s\in D_c\wedge s='seq'\wedge s\in lefttree(u)\wedge v\in lefttree(s))$成立，称 v 满足 aft-exec 重构方式。

定义 3 如果横切关注点v使得$((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\rightarrow u\in R^{\rightarrow}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists o\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge o\in X\wedge u\in N_P\wedge u\in lefttree(o)\wedge v\in lefttree(o)\wedge v\rightarrow u\in S^{\rightarrow})))\wedge\exists s(s\in D_c\wedge s='seq'\wedge v\in lefttree(s)\wedge u\in righttree(s))$成立，称$v$满足bef-call重构方式。其中，$righttree(s)$表示方法调用树中$w$的右子树。

定义 4 如果横切关注点v使得$((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\leftarrow u\in R^{\leftarrow}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists o\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge o\in X\wedge u\in N_P\wedge u\in lefttree(o)\wedge v\in lefttree(o)\wedge v\leftarrow u\in S^{\leftarrow})))\wedge\exists s\exists t(s\in D_c\wedge t\in D_c\wedge t='seq'\wedge u\in lefttree(s)\wedge t\in righttree(s)\wedge v\in lefttree(t))$ 成立，称v满足aft-call重构方式。

定义 5 如果横切关注点v使得$((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\triangledown u\in R^{\triangledown}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge u\in X\wedge v\triangledown u\in S^{\triangledown})))\wedge\exists s(s\in D_c\wedge s='if'\wedge s\in lefttree(u)\wedge v\in lefttree(s))$成立，称$v$满足aro-bef-exec重构方式。其中，$if$表示方法调用树中条件控制结点。

定义 6 如果横切关注点v使得$((v\in CR_{same}\wedge\exists u(u\in N_P\wedge v\triangle u\in R^{\triangle}))\vee(v\in CR_{diff}\wedge\exists X\exists Y\exists u((X,Y)\in CON_{diff}\wedge v\in Y\wedge u\in X\wedge v\triangle u\in S^{\triangle})))\wedge\exists s(s\in D_c\wedge s='if'\wedge s\in lefttree(u)\wedge v\in lefttree(s))$成立，称$v$满足aro-aft-exec重构方式。

当一个横切关注点满足多种重构方式时，只需为其选择一种重构方式进行重构。为此，本文根据重构的复杂程度和产生的 aspect 代码的质量，设定了 6 种重构方式的优先级，bef-exec 和 aft-exec 重构方式优先级最高，bef-call 和 aft-call 重构方式优先级其次，aro-bef-exec 和 aro-aft-exec 重构方式优先级最低。

步骤 3 重构

每种重构方式都要完成将横切关注点封装为 aspect 的代码转换，下面用例子来描述每种重构方式下的代码转换结果。

(1)bef-exec 重构方式。图 1 描述了 bef-exec 重构方式重构横切关注点 $v()$为 aspect B 的结果。ds 表示声明语句集合；T 表示方法 u 的返回值类型；ups 和 vps 表示方法 u 和方法 v 的函数参数集合；ss 表示语句集合；$context$ 表示通知的参数列表；$\sigma(u())$表示方法 u 的签名；$bind(context)$表示切入点指示器的表达式。

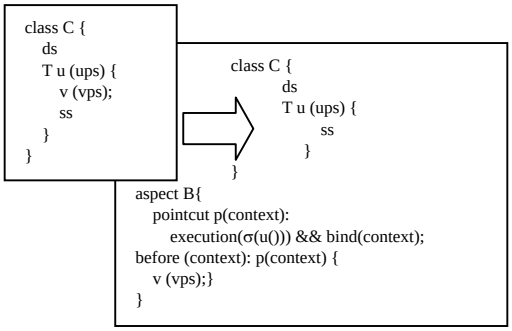

图 1 bef-exec 重构方式重构结果

(2)aft-exec 重构方式。aft-exec 重构方式重构横切关注点 $v()$为 aspect 的结果与 bef-exec 重构方式相似，区别在于重构前 $v(vps)$位于 ss 之后，重构后 asepct B 的通知变为 after。

(3)bef-call 重构方式。图 2 描述了 bef-call 重构方式重构横切关注点 $v()$为 aspect B 的结果。nps 表示方法 n 的函数参数集合；$\sigma(C.n())$表示类 C 的方法 n 的签名。

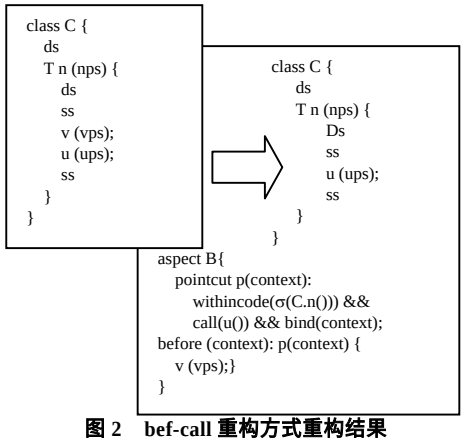

图 2 bef-call 重构方式重构结果

(4)aro-bef-exec 重构方式。图 3 描述了 aro-bef-exec 重构方式重构横切关注点 $v()$为 aspect B 的结果。$cond$ 表示条件。

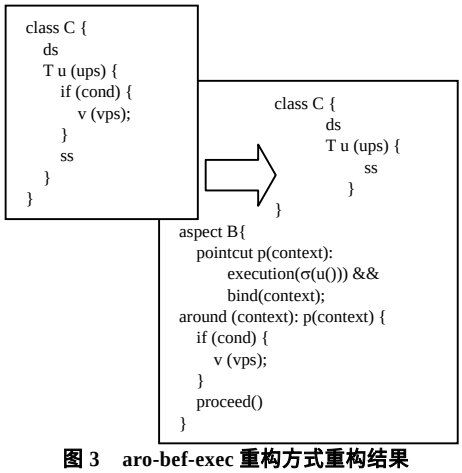

图 3 aro-bef-exec 重构方式重构结果

(5)aft-call 重构方式。aft-call 重构方式重构横切关注点

(下转第 59 页)