计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
47(12): 2192-2200, 2010

通过静态分析逆向恢复面向对象程序中的用况

叶彭飞　彭　鑫　赵文耘
(复旦大学计算机科学技术学院　上海　200433)
(cs. yepengfei@gmail. com)

Recovering the Use Case from Object-Oriented Programs by Static Analysis

Ye Pengfei, Peng Xin, and Zhao Wenyun
(*School of Computer Science and Technology, Fudan University, Shanghai* 200433)

Abstract　In the process of software maintenance, the maintainer can obtain helpful information via reading the use case documents. The problem in real software maintenance is that the maintainer can only obtain out-of-date or incomplete information of the use case. To solve this problem, a novel approach is proposed to identify the use case from object-oriented source code in the function logic level of the software system. The analysis of the behavior protocol of the high level decade classes, which interact with user interface in an object-oriented system, is involved to help discovering the high level system running scenarios which are high level parts of the discovered use cases. Then the conventional branch-reserving call graph is extended to an object-oriented compatible version called object-oriented branch-reserving call graph (OO-BRCG). After appropriate pruning process applied on the OO-BRCG, each running path leftover in the pruned OO-BRCG is regarded as a low level part of the discovered use cases. Combining the high level and low level use parts can get the complete use cases. An experiment for this method on an object-oriented system in real world is performed. The results from the experiment show that this approach can gain a very high recall with acceptable loss on the precision. The experiment has confirmed the overall effectiveness of this approach.

Key words　use case; object behavior protocol; recovery; static analysis; program call graph

摘　要　在软件维护任务中,通过阅读用况能有效地帮助维护人员理解软件系统,然而在现实中用况文档往往是过时或残缺不全的.如何通过代码分析还原用况是一大难题.针对上述问题提出了一种针对面向对象程序源代码通过静态代码分析逆向恢复用况的方法.该方法在高层通过分析系统逻辑层高层门面类的对象行为协议来获取用况的高层划分,在底层通过分析 OO-BRCG(object-oriented branch-reserving call graph)来得到用况的底层划分,然后结合两方面恢复出最终用况.最后通过实验验证了该方法的有效性,恢复用况时该方法能获得极高的用况覆盖度及可观的准确度.

关键词　用况;对象行为协议;逆向恢复;静态分析;程序调用图
中图法分类号　TP311.5

0　引　　言

准确的程序理解是软件维护的重要前提,例如 Corbi 就曾指出[1]:在软件维护任务中,几乎一半的时间被用来理解手头待维护的系统.然而软件维护过程中的一个现实问题是遗产系统往往缺乏详细、规范的系统需求和设计文档,或者相关文档未能随着系统的持续演化过程进行必要的更新.这大大增加了软件维护人员理解遗产系统的难度.

收稿日期:2009-01-05;**修回日期**:2010-01-27
基金项目:国家自然科学基金项目(60703092);国家"八六三"高技术研究发展计划基金项目(2007AA01Z125,2009AA010307)
通信作者:彭　鑫(pengxin@fudan. edu. cn)

随着 UML 的广泛普及,用况已经成为描述软件系统需求的重要手段.用况从外部用户的角度描述了系统的功能性需求,更加接近人的理解;同时用况描述中包含事件的顺序,又比较符合系统的运行时特性.因此,基于源代码分析的用况逆向恢复能够对开发人员的程序理解提供有力的支持.在已有的用况逆向恢复相关研究中,既有使用动态分析方法也有使用静态分析方法用于用况的提取.文献[2]中,Lucca 等人提出了一种基于静态结构分析提取用况的方法.该方法提出,在面向对象系统中用况在代码上可以被映射为方法调用的序列,该方法认为一条从含有输入的方法到含有输出的方法的调用序列是一个潜在的用况.这种方法的不足主要体现在:没有考虑程序内在的用况划分机制;用况往往会包含多次用户交互,而该方法没有考虑任何交互信息.文献[3]提出了一种基于动态分析的用况提取方法,首先在程序运行中记录下的系统和用户的交互情况,然后分析系统和用户的交互来提取用况.该方法需要足够多并且对系统功能覆盖足够全面的测试用例来提供支持,这个要求在现实中往往难以实现.文献[4]则提出了一种基于程序中的分支语句的用况提取方法.该方法中认为代码中隐含着 if 分支语句等划分不同用况实现的逻辑机制,通过分析这些划分信息可以有效地获取用况.该方法以一种带分支信息的程序调用图——带分支信息的调用图(branch-reserving call graph,BRCG)——为基础,通过对分支信息的分析以及剪枝算法的运用得到调用轨迹划分,从而获取候选用况.该方法能有效地克服之前提出的动态方法与静态方法的弊端,但只能针对结构化程序,而且对用户交互的考虑也不多.

针对这些问题,本文就面向对象的交互式系统,提出了一种新的用况逆行恢复方法.该方法将BRCG扩展为包含继承等面向对象特性的OO-BRCG,并将 OO-BRCG 与对象行为协议分析相结合实现用况逆向恢复.大多数面向对象程序都强调逻辑层与界面层的分离,用户交互由界面层实现,而业务逻辑则由逻辑层实现.本文所提出的方法针对面向对象程序的业务逻辑层进行用况逆行恢复.该方法首先通过分析系统逻辑层顶层门面类的对象行为协议得到用况高层抽象的划分依据,然后通过对系统 OO-BRCG 得到用况底层抽象的划分依据,之后结合以上两步分析的结果获取最终的用况.在处理 OO-BRCG 的过程中,我们在文献[4]的剪枝方法基础上提出了基于用户交互信息的剪枝方法.该方法首先根据与用户输入信息的关系自顶向下地为 OO-BRCG 中涉及到的变量赋予用户交互关联度,之后自底向上根据用户交互关联度进行剪枝.这种剪枝方法将用户输入信息的直接和间接影响纳入到考察范围中,使得所得到的用况中的用户交互性体现得更为明显.

1 相关背景

1.1 对象行为协议

在面向对象的软件系统中,对象作为状态和行为的封装体存在.对于有状态的对象,其对外提供的多个公共方法在使用上并不是完全独立的,而是具有一定的约束关系,这种关系称为对象行为协议,通常可以用有限状态机进行描述[5].对象行为协议中隐含着软件系统同一模块中不同子功能之间的功能性约束.对象行为协议在程序的理解、测试和软件重用等方面都有十分关键的作用.

在一个设计良好的面向对象程序中,每个子模块由一系列类构成,内部类相互以消息传递的方式进行交互以完成内部功能逻辑,对外通过门面类(facade)提供访问接口,模块与模块之间通过消息来激发逻辑功能执行.现在的大量面向对象程序强调逻辑层与界面层的分离.逻辑层中最上层的模块向界面层提供门面类作为接口,界面层向接口发送消息来调用相应的功能.例如,使用 struts 进行开发时,用户界面由 JSP 实现,而界面间流转则由 XML 配置文件实现;使用 MFC 开发时,使用消息与监听器来实现界面;而对于 Web Service,则不存在图形用户界面层,逻辑层接口以服务的方式直接暴露给客户端.由于界面层实现技术的多样性以及与逻辑层实现技术的差异性,同时考虑到用况的具体实现主要在逻辑层中,本文的方法主要针对面向对象的逻辑层代码进行分析,而与界面层直接交互的门面类相关的交互流程信息则借助于对象行为协议逆向恢复工具来进行获取.

我们项目组此前在对象行为协议恢复方面已经作了一些相关工作,并分别提出了基于动态分析[5]和静态分析[6]的行为协议逆向恢复方法.在文献[6]中,我们提出了一种基于依赖性分析的对象行为协议逆向恢复方法,该方法通过分析对象中的公共方法对全局数据(类属性或对象属性)的依赖来得到公共方法之间的前驱和后继关系,能有效地还原出对象行为协议.由于同样属于静态分析方法,我们决定

使用该对象行为协议恢复方法来从源码中获取程序逻辑层顶层门面类的对象行为协议.

1.2 OO-BRCG

BRCG 是一种调用图的扩展形式,在调用图的基础上保留了代码中的分支信息[7]. BRCG最初提出被用来分析结构化程序的程序结构. 在一个BRCG 中有 3 种类型的节点:过程调用、分支语句、分支;节点与节点之间有两种关系:顺序关系和分支关系.

最初的 BRCG 把程序用的 if 语句、switch 语句和循环语句作为获取分支语句节点的依据. 然而在面向对象程序中,程序结构不仅包含在方法调用的轨迹中,还包含在继承结构中. 为了在分析面向对象程序时使用 BRCG,我们需要对原始的 BRCG 进行扩展,扩展后支持面向对象特性的 BRCG 称为 OO-BRCG(object-oriented branch-reserving call graph).

定义 1. OO-BRCG. 一个 OO-BRCG 可以被定义成一个三元组 $OO\text{-}BRCG=\langle N,S,R\rangle$,其中:

1) N 是节点的集合,总共有 4 种类型的节点:方法调用(method)、分支语句(branch statement)、分支(branch)、循环语句(loop);

2) S 是关系集合,总共有 3 种关系:顺序关系(sequential)、分支关系(branch)、循环调用关系(loop_call);

3) R 是 $N\times N\times S$ 的子集,表明了节点与节点之间的相互关系;

4) 对于 $\forall(n_1,n_2,r_1)\in R$,有 $\forall(n_1,n_3,r_2)\notin R$, $n_2\neq n_3$, $r_1\neq r_2$.

在 OO-BRCG 中,分支语句节点对应着代码中的 if 语句、switch 语句以及存在动态绑定的方法调用,循环语句节点对应着代码中的 for 语句、while 语句等循环语句. 在 OO-BRCG 中,节点与节点之间的关系是由父节点决定的,父节点与其子节点有且仅有一种关系,分支语句节点(branch statement)的子节点为分支节点,两者是分支关系的;循环语句节点与其子节点的关系为循环调用关系;其他情况是顺序关系的. 图 1 为一个网上购物系统的 OO-BRCG 例子:

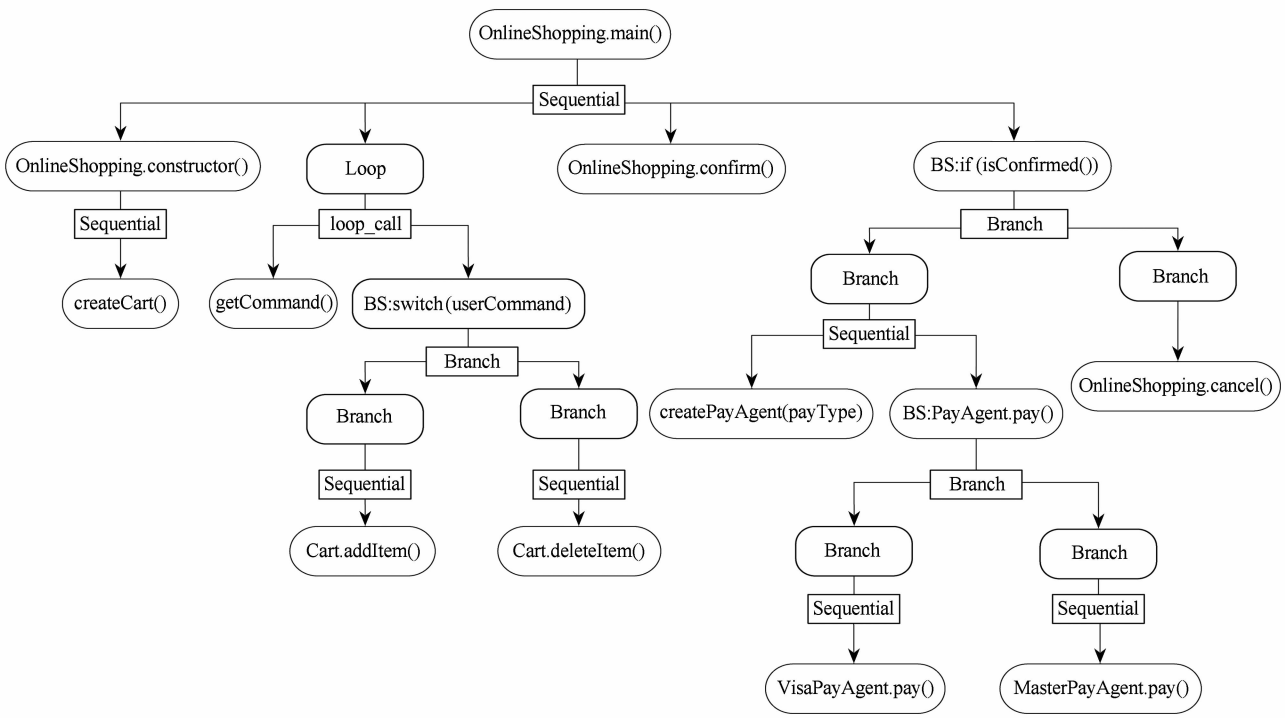

Fig. 1 OO-BRCG of an online shopping system.

图 1 网上购物的 OO-BRCG

2 用况逆向恢复方法

本文的用况提取方法主要流程如图 2 所示,主要包括 5 个步骤. 首先,通过代码分析得到 OO-BRCG 结构(可能有多个);接着根据 OO-BRCG 的根节点识别出逻辑层的顶层门面类;然后利用对象行为协议逆向恢复工具提取出门面类的对象行为协议,在

提取对象行为协议时还要对 OO-BRCG 作一定处理,提取出以门面类中的方法为根节点的OO-BRCG(门面类中有些方法可能不存在以其根节点的 OO-BRCG,但是可以在已有的 OO-BRCG 中以提取子树的方式提取出来);接着对所有的 OO-BRCG 应用剪枝算法进行剪枝;最后,用况恢复模块将剪枝过的 OO-BRCG 和提取出的对象行为协议结合起来,实现最终的用况恢复.下面将对几个关键步骤进行详细介绍.

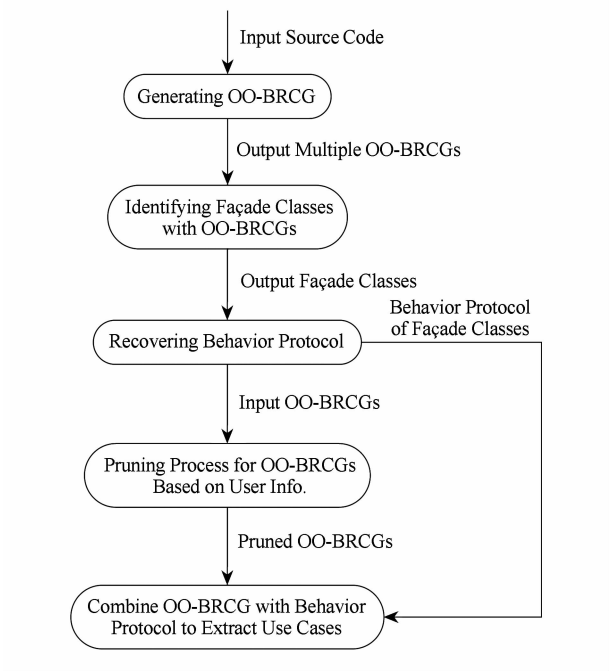

Fig. 2 Flow cart of the use case recovery process.

图 2 用况逆向恢复方法流程

2.1 OO-BRCG 的构造

在本文的方法中通过分析面向对象程序的逻辑层源代码来生成 OO-BRCG. OO-BRCG 采用类似 Zhang 等人提出的子图合并的方法[4]来生成,先对每种方法生成 OO-BRCG 子图,再通过合并相同的方法调用节点把 OO-BRCG 子图合并成完整的 OO-BRCG. 面向对象程序中的继承和动态绑定特征会影响 OO-BRCG 中分支语句的选择. 所以,在构造 OO-BRCG 之前我们先要构造一个全局的类继承索引,以便在构造 OO-BRCG 子图时获取正确的继承信息. 类继承索引由一系列索引单元组成,每一个索引单元代表着一棵继承树,类继承索引单元的定义如下:

定义 2. 类继承索引单元. 一个类继承索引单元可以定义成一个四元组 $InheritanceIndexUnit=\langle C,CR,M,MC\rangle$,其中:

1) C 是类的集合,C 中每个类用包名加类名表示;CR 是 $C\times C$ 的子集,$\forall(c_1,c_2)\in CR$ 表示 c_1 是 c_2 的父类,C 和 CR 描述了一棵继承树的类继承结构;

2) M 是方法的集合,M 中每种方法用方法签名(方法名加参数列表)来表示;MC 是 $M\times C$ 的子集,$\forall(m,c)\in MC$ 表示方法 m 在类 c 中存在一个实现.

在代码中遇到某个类的对象上某种方法的调用时,可以通过类继承索引方便地得到该方法调用处的动态绑定信息,查找算法如图 3 所示:

```
Input: class name: c, method signature: m, inheritance index: I;
Output: set of class name: cSet, each class in the set has method implementations which could
        possibly be dynamic binding methods;
Step1. find the index unit: indexUnit, which matchs c∈indexUnit.C;
Step2. generate a set: Candidate⊂indexUnit.C, ∀c1∈Candidate, (m,c1)∈indexUnit.MC;
Step3. find the set: cChildren of all descendants of c which is in Candidate,
       make cSet=cChildren;
       if (candi in Candidate && candi.m is not abstract method)
          cSet=cSet∪{candi};
       else
          find the nearest ancestor candi' of candi in Candidate;
          cSet=cSet∪{candi'};
Step4. return cSet.
```

Fig. 3 Search algorithm of the inheritance index.

图 3 类继承索引查找算法

2.2 OO-BRCG 的剪枝

在 OO-BRCG 中,有一些节点表达的是用户所不关心的接近系统底层实现细节的功能,这些节点不仅不会为用况识别提供帮助,甚至可能造成误导.

因此,为了最后能够得到尽可能准确的用况,需要对原始的 OO-BRCG 进行剪枝,剪枝的目的在于剪去那些接近系统底层实现逻辑的方法调用节点及其子节点,保留实现系统高层功能逻辑的方法调用节点.由于剪枝意义在于考虑用户关注度,我们提出了一种基于用户交互关联度的剪枝方法.我们认为最原始的用户信息为 OO-BRCG 根节点表示的方法的传入参数(可以认为是用户输入),接下来将自顶向下地为 OO-BRCG 中涉及到的变量赋予用户交互关联度(user interaction degree,UID):

1. 根节点的传入参数的 UID 为初始值 w_1;赋值语句左右表达式(变量)UID 相等,方法的形参和实参 UID 相等;一个表达式可以作为一个逻辑变量,根据推理规则计算 UID.

2. 简单变量的计算表达式.表达式的 UID 等于表达式中具有 UID 的变量的 UID 之和加上表达式中不具有 UID 的变量的数量(包括常量).

3. 方法调用表达式.面向对象中一个方法调用由对象、对象的方法和参数列表 3 部分组成,若对象没被赋予 UID 并且参数列表中没有拥有 UID 的变量,则表达式没有用 UID;其余分两种情况:

1) 如果对象不具有 UID,表达式的 UID 等于参数列表中具有 UID 的变量的 UID 平均值加上参数列表中没有 UID 的变量(包括常量)数量.

2) 如果对象具有 UID,表达式的 UID 等于对象的 UID 度加上参数列表中具有 UID 的变量的最小 UID 再加上参数列表中没有 UID 的变量(包括常量)数量.

4. 对于对象的构造,如果没有参数或者参数列表中没有具有 UID 的变量,则该构造表达式没有 UID;否则该构造表达式的 UID 等于参数列表中具有 UID 的变量的 UID 平均值加上参数列表中没有 UID 的变量(包括常量)数量.

在为 OO-BRCG 中的变量赋予用户交互关联度之后是自底向上的剪枝工作,剪枝时需要手动设置一个阈值 T.剪枝时隐含了一个原则,当一个节点能被继续展开(子节点没被剪去),则不需要再往上考虑他的父节点.在剪枝的过程中不考虑循环语句节点,因为考虑到循环语句节点无论对用况的划分还是对用况的粒度影响都不足够大.定义一个节点的直接方法子节点为该节点的子孙节点中距离该节点最近的那些方法节点.具体的剪枝过程如下:

1) 对于直接方法子节点全是叶子节点的方法调用节点,该方法调用当成一个方法调用表达式,根据 UID 计算规则中的规则 3 来计算该节点的 UID,如果 UID 大于等于阈值 T,则把该方法节点变成一个叶子节点,剪去所有子节点,没有 UID 等价于 UID 无限大;

2) 对于直接方法子节点全是叶子节点的分支语句节点,计算该分支语句节点的 UID,如果 UID 大于等于阈值 T,则把该分支语句节点变成一个叶子节点,剪去所有子节点;分支语句 UID 等于分支语句中拥有 UID 的变量的最小 UID;

3) 自底向上重复 1,2 两步直到不能再进行任何剪枝为止.

2.3 用况提取

获得了经过剪枝的 OO-BRCG 和对象行为协议之后将进行用况提取.与文献[2,4,7]中的想法类似,我们也认为用况在代码上可以被映射为方法调用的序列.

对象行为协议用状态机来表示,状态机上的变迁对应的方法能映射到一个剪枝后的 OO-BRCG,状态机上每条以非错误状态节点为终止节点的变迁轨迹能对应上一条程序逻辑层顶层门面类中的方法调用序列.由于状态机上的变迁轨迹可能有产生无限循环的情况,例如,对于某操作有设置和取消设置两种操作,可能产生“设置、取消、设置、取消、……”的无限循环序列.这时,需要软件维护人员对行为协议状态机上的变迁轨迹进行筛选,选出合理的变迁轨迹.在编写程序时,由于对模块内聚度和耦合度的考虑,模块的门面类中接口方法的数量往往不会过于庞大,而且对接口方法会采用合理的命名.尤其对于逻辑层顶层的门面类,每个接口方法提供的功能已经十分接近用户对系统功能的理解,所以在选择行为协议状态机上的变迁轨迹时引入软件维护人员的协助,无论是维护人员的工作量还是选择的变迁轨迹的合理性方法都是可接受的.经过门面类行为协议的分析可以得到多个用况集合,每个用况集合对应着行为协议状态机上的一条变迁轨迹.每条变迁轨迹对应的方法调用序列是相应用况集合中所有用况公共的高层表示.

经过剪枝的 OO-BRCG 上每条从根节点开始的执行轨迹(方法调用序列)对应着一个子用况,该子用况是一个完整用况的底层表示.结合OO-BRCG 上得到的执行轨迹以及对象行为协议状态机上得到的

执行轨迹能够得到一个用况的完整表示.假设一条对象行为协议状态机上执行轨迹有 n 种方法对应 n 个剪枝后的 OO-BRCG,通过 n 个 OO-BRCG 得到的底层执行轨迹数量分别是 $m_1, m_2, \cdots, m_n$,则该条对象行为协议状态机上的执行轨迹得到的用况有 $m_1 \times m_2 \times \cdots \times m_n$ 个.

2.4 用况逆向恢复实例

本节我们将用一个实现登录功能的简单例子进一步说明本文方法的流程及一些细节.该例子中 3 个类,分别是 Login,User,PersistenceManager,其中 Login 是对外的门面类.该例子应用我们方法的步骤如下:

首先生成初始 OO-BRCG,可以得到 3 个 OO-BRCG,如图 4 所示,其中 init()和 reset()操作十分简单,所以 OO-BRCG 很小,主要的方法是 login().由初始 OO-BRCG 可以确定门面类为 Login.

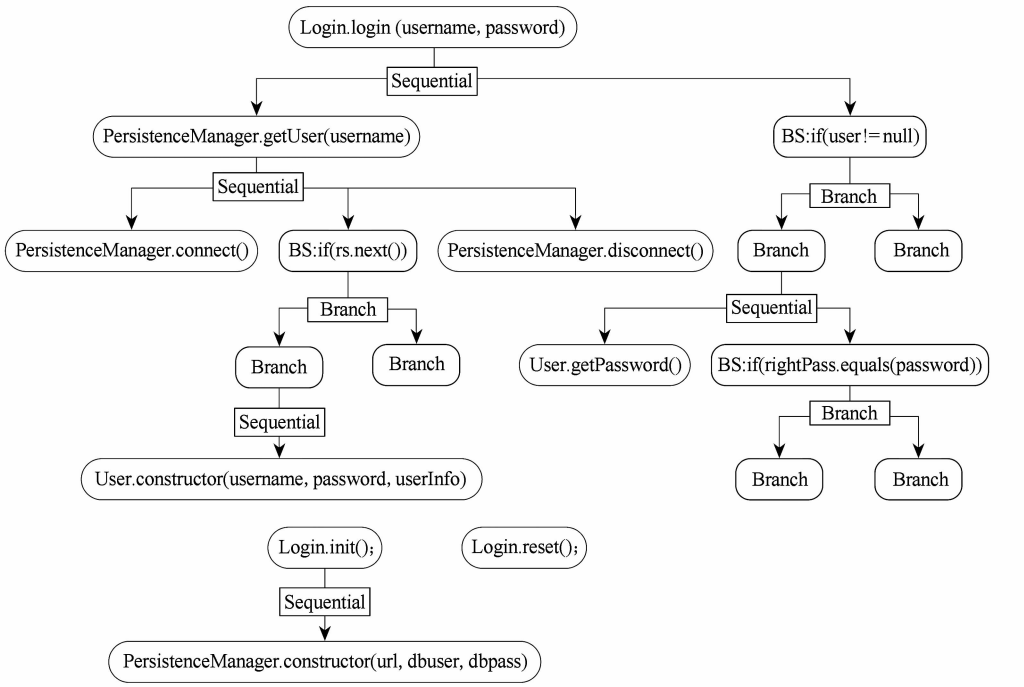

Fig. 4 Initial OO-BRCG.

图 4 初始 OO-BRCG

接下来应用对象行为协议恢复出 Login 的对象行为协议,用状态机来表示,Login 的行为协议状态机如图 5 所示.由于所有门面类的方法都有一个 OO-BRCG 与之对应,不需要特别处理 OO-BRCG,接下来将进行剪枝,剪枝时,我们设定用户交互关联度初始值为 1,这里由于分析的程序十分简单,阈值只要设为 2 就可以,剪枝后的 OO-BRCG 如图 6 表示.

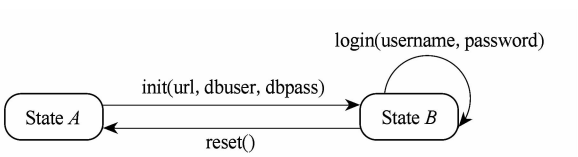

Fig. 5 The behavior protocol state machine of class Login.

图 5 Login 类的行为协议状态机

最后是结合行为协议状态机和 OO-BRCG 进行用况逆向恢复的过程.通过考察行为协议状态机,可以得到用况的高层划分,该例子中仅有一个用况集合:init()→login()→reset().然后把相应的门面类方法对应到 OO-BRCG 上,结合 OO-BRCG 可以得到完整的用况,如表 1 所示,表 1 中省略了公共的 init()和 reset().

Table 1 Identified Use Cases

表 1 最终提取的用况

Use Case	Execution Trace
Login Success	login(){getUser()→if branch{getPassword→if branch}}
User not Exist	login(){getUser()→else branch}
Wrong Password	login(){getUser()→if branch{getPassword→else branch}}

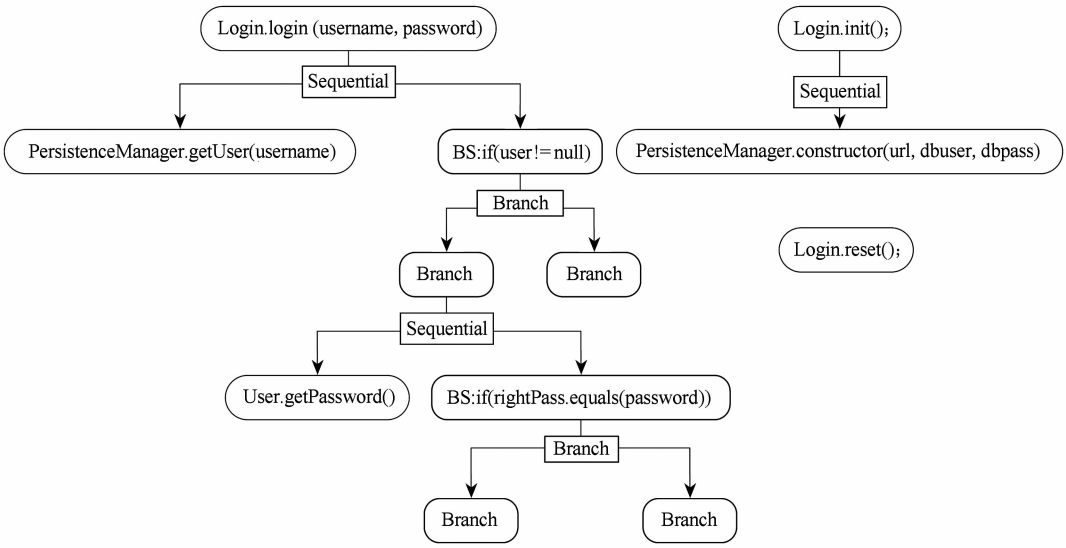

Fig. 6 The pruned OO-BRCG.

图 6 剪枝后的 OO-BRCG

3 实验与讨论

为了验证本文方法的有效性,我们使用了一个网上自学考试管理系统来进行实验. 该系统拥有两类用户:学生和管理员. 该系统功能逻辑用 Java 实现,界面层为 JSP 页面,在 JSP 页面中调用逻辑层 JavaBean 接口提供的功能. 逻辑层的门面类主要有两个:Student 和 ExamManager,在 Student 中维护和学生相关的信息,并且提供和学生相关的操作,每个 Student 对象对应一个实际的学生;ExamManager 用来全局管理考试,提供考试相关操作(包括添加考试、激活考试、成绩录入等). 在表 2 中前 3 列给出了文档中的具体用况描述和实际用况数量.

由于在筛选行为协议状态机的变迁轨迹是需要软件维护人员的参与,我们让项目组中的其他同事充当软件维护人员的角色. 在剪枝算法中,我们选择用户交互关联度的初始值为 1,经过实验调整后,我们选择了一个最合理的阈值 $T=4$. 最终的实验结果在表 2 最后两列中给出. 文档中记录的用况总共有 19 个,表中识别出的正确用况数大于记录用况数是由于相同用况的不同表达形式引起的,在下文中会进行分析. 通过我们的方法总共识别出 42 个用况,其中 22 个是正确的,准确度 $precision=52.38\%$;有记录的 19 个用况有 1 个没有识别出,覆盖度 $recall=94.74\%$. 下面对实验结果进行分析.

我们选择用来实验的系统规模相对较小,所以逻辑层的层数嵌套数很少,细分可得到 3～4 层. 该系统中方法的调用层次不多,我们选取初始值四倍的阈值就可以得到比较好的结果. 一般可以采用启发式的方法来获得最合理的阈值:让维护人员设定一个初始阈值,返回分析结果;维护人员根据结果调整阈值(用况粒度太细则减小阈值,用况粒度太粗则增大阈值),直到获得一个比较满意的结果.

在实验结果中我们可以看出,报考课程功能模块中总共有 7 个用况,其中有 2 个和返回相关的,其他 5 个和手续费支付方式相关. 和返回相关的用况有一个没有被识别出来,因为在确认后界面层会调用 Student 中的 initPayAgent(),该方法中调用 selectedCourses 对象的方法来计算应缴费金额,如果没有记录任何选中课程会抛出异常,该异常在 jsp 中被获取将进行返回操作,在对象行为协议状态状态机中该轨迹终止节点是一个出错状态. 另外,可以发现识别出的报考课程的和支付方式相关的用况有 25 个,其中只有 5 个是正确的. 在 initPayAgent() 中生成相应的 PayAgent 对象,这里根据用户的选择不同有 5 个分支对应着 5 个 PayAgent 子类对象的生成. 之后在 pay()方法中调用 PayAgent 对象上的 pay()来进行实际支付,这里涉及到一个动态绑定引入的分支,同样是 5 个分支. 对象行为协议上会发现在 initPayAgent()后调用 pay()的执行轨迹,根据我们之前给出的算法会得到 $5\times5=25$ 个用况,但其中只有 5 个是有意义的. 发生这种情况主要是因为两种方法之间或有隐含的语义依赖关系,上述情况,initPayAgent()中的用户选择决定了 PayAgent 的类型,这个类型同样决定之后动态绑定的分支,然

而我们的方法无法挖掘这种隐含的依赖.

在毕业申报和考试管理模块,均出现了识别出的正确用况比文档中记录的用况数量多的情况.这是由于文档中的用况描述和代码实现的细微差距造成.以考试管理为例,文档中记录了3个用况:添加考试后激活考试;添加考试后在未激活的情况下删除考试;录入考试成绩.然而,在代码实现时实现了4种方法:addExam(),deleteExam(),releaseExam(),recordExamResult(),分析ExamManager门面类发现这4种方法是平等的,没有依赖关系;进一步考察发现是因为在这4种方法中实际执行操作前都会检查对应的考试的状态,状态不正确的情况认为会返回操作失败.我们的方法通过分析源代码得到了4个用况,经过考察我们认为这4个用况应被归为正确的用况,相对文档记录可以认为是同样功能的不同用况描述.

Table 2 Results of the Experiment
表2 实验结果

User	Description of Use Cases	Num. of Use Case Recorded in the Documents	Num. of Use Case Identified	Num. of Use Case Identified Correctly
Student	Maintanence of personal info.: including changing password and changing personal info.	4	4	4
	Application of courses: first choose courses,4 operations: add course,delete course,choose pay mode and confirm,return. After confirmation,enter online payment step. when confirming,the course list can not be empty,if the list is empty,then return automatically. Supported pay modes: alipay,ABC,CCB,CMB,ICBC.	7	26	6
	Application of graduation: after submiting the application,the administrator check all information in the application and return the result: pass or not pass,the administrator could also delay returning the result. The student could cancel the application before the result comes out.	4	5	5
	Score query: 3 kinds of query,query all,query by year and query by subject.	3	3	3
Admin	Exam management: add exam,after the exam is added,it should be activated before it is released to the students,the exam could be deleted before activation.	3	4	4

4 总结和未来工作

用况的提取是一种非常有实用价值同时又极具挑战性的工作,本文提出了一种对象行为协议和OO-BRCG相结合的在面向对象程序源代码中提取用况的方法.在高层,我们通过分析门面类的对象行为协议获取用况的划分依据;在底层,我们通过一种BRCG的面向对象扩展版OO-BRCG来辅助挖掘用况的底层划分依据;结合OO-BRCG和顶层门面类的对象行为协议,最终可以提取出正确的用况.通过分析一个网上自学考试管理系统,我们验证的本文方法的有效性.我们的方法在提取用况时能到达极高的覆盖度,然而在准确度方面我们的方法容易受到一些不好的设计以及实现与文档的差别所影响,识别出一些没有意义的用况.在实验中我们发现了本文方法有待改进的地方,未来的进一步研究集中在以下两个方面:1)本文方法中的用户交互关联度推理规则还比较粗糙,需要进一步优化推理规则,同时可以通过实验挖掘阈值设定和系统规模之间的定量关系;2)在结合行为协议状态机和OO-BRCG时,会因为一些隐含依赖信息的丢失而识别出一些没有意义的用况,影响最终结果的准确度,需要寻找新的途径挖掘出这些隐含的依赖信息,以提高方法的准确度.

参 考 文 献

[1] Corbi T A. Program understanding: Challenge for the 1990s [J]. IBM Systems Journal, 1989, 28(2): 294-306

[2] Lucca G A Di, Fasolino A R, Carlini U De. Recovering use case models from object-oriented code: A thread-based approach [C] //Proc of the 7th Working Conf on Reverse Engineering (WCRE'00). Los Alamitos, CA: IEEE Computer Society, 2000: 108-177

[3] El-Ramly M, Stroulia E, Sorenson P. Mining system-user interaction traces for use case models [C] //Proc of the 10th Int Workshop on Program Comprehension (IWPC'02). Los Alamitos, CA: IEEE Computer Society, 2002: 21-29

[4] Zhang Lu, Qin Tao, Zhou Zhiying, et al. Identifying use cases in source code [J]. Journal of Systems and Software, 2006, 79(11): 1588-1598

[5] Wu Hao, Peng Xin, Zhao Wenyun. An automatic extraction method of object state machine based on condition expressions [J]. Journal of Chinese Computer Systems, 2008, 29(9): 1584-1590 (in Chinese)
(吴浩, 彭鑫, 赵文耘. 一种基于分支条件的对象状态机自动提取方法[J]. 小型微型计算机系统, 2008, 29(9): 1584-1590)

[6] Huang Zhou, Peng Xin, Zhao Wenyun. Behavior protocol recovery based on dependency analysis [J]. Journal of Computer Science, 2008, 35(8): 265-268 (in Chinese)
(黄洲, 彭鑫, 赵文耘. 基于依赖性分析的对象行为协议逆向恢复[J]. 计算机科学, 2008, 35(8): 265-268)

[7] Qin T, Zhang L, Zhou Z, et al. Discovering use cases from source code using the branch-reserving call graph [C] //Proc of the 10th Asia-Pacific Software Engineering Conference. Los Alamitos, CA: IEEE Computer Society, 2003: 60-67

Ye Pengfei, born in 1987. Master candidate at the School of Computer Science, Fudan University. His current research interests include software reengineering and software maintenance.

叶彭飞, 1987 年生, 硕士研究生, 主要研究方向为软件再工程和软件维护.

Peng Xin, born in 1979. PhD and associate professor at the School of Computer Science, Fudan University. Member of China Computer Federation. His main research interests include software product line, software component and architecture, software maintenance and software reengineering.

彭　鑫, 1979 年生, 博士, 副教授, 中国计算机学会会员, 2006 年在复旦大学获得博士学位, 主要研究方向为软件产品线、软件构件与体系结构、软件维护与再工程.

Zhao Wenyun, born in 1964. Professor and PhD supervisor at the School of Computer Science, Fudan University. Senior member of China Computer Federation. His main research interests include software engineering. software reuse, software component and architecture, software maintenance and reengineering.

赵文耘, 1964 年生, 教授, 博士生导师, 中国计算机学会高级会员, 主要研究方向为软件工程、电子商务.

Research Background

In software maintenance, reading the use case documents can help the maintainers obtain helpful information about the software system under maintenance. The problem is that information of use case is usually out-of-date or incomplete. To solve this problem, we propose a novel approach to identify the use case in object-oriented source code in this paper. We introduce the analysis of the behavior protocol of the high level decade classes in an object-oriented system to help discover the high level use case separation. Then with the help of the OO-BRCG, an extension of the conventional BRCG, the low level use case separation is gained. Combining the high level and low level use case separation can get the whole use cases. We finally carry out an experiment on an object-oriented system in the real world. The results from the experiment have confirmed the effectiveness of our approach. Our work is supported by the National Natural Science Foundation of China (No. 60703092) and the National 863 High Technology Development Program of China (Nos. 2007AA01Z125 and 2006AA01Z189).