

# 一种基于构件的回归测试用例选择方法

陈 峰, 李心科

(合肥工业大学 计算机与信息学院, 安徽 合肥 230009)

**摘 要:** 构件的使用给大规模软件系统开发带来了方便, 但构件测试仍然是软件工程中很难解决的问题。由于构件使用者对构件内部结构及变更信息缺乏了解, 因此很难进行有效的回归测试用例选择。针对这一问题提出了一种回归测试用例选择方法, 通过论述及实验分析, 初步证实了所提出的方法在实际中的可行性和有效性。

**关键词:** 回归测试; 构件软件; 用例选择

**中图分类号:** TP311.5

**文献标识码:** A

**文章编号:** 1673-629X(2008)11-0131-03

## A Component - Based Regression Test Case Selection Method

CHEN Feng, LI Xin-ke

(School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

**Abstract:** The use of component has brought convenience for the large - scale software system development, but the component test is still very difficult to solve in the software engineering. Because the component user lacks the information of the component internal structure and the change, it is difficult to carry on effectively regression test cases selection. Presents a component - based method which can select regression test cases. The elaboration and the experiment show that the strategy of regression test case selection is feasible and effective in practice.

**Key words:** regression testing; component - based software; test case selection

## 0 引 言

随着软件规模的不断增大, 构件使用越来越频繁, 虽然构件的使用给软件开发带来很多方便<sup>[1]</sup>, 但由于构件使用者缺乏一些构件内部信息, 因此增加了测试的难度。当构件中的某些部分被修改, 或引入了新的构件, 这时就需要对系统重新测试以保证修改的正确性, 即进行回归测试。根据统计, 回归测试占整个软件系统开销的  $1/3$ <sup>[2]</sup>。

完全的回归测试不仅没有必要, 而且由于测试成本等原因, 也是不现实的<sup>[3]</sup>。内建式设计<sup>[4]</sup>很好地解决了构件的测试问题, 在回归测试用例选择方面还存在许多问题。

文中提出了一种测试用例选择方法, 主要针对构件发生变更情况下, 根据变更及其所能影响的部分从原有用例中选择出部分用例用于测试。

## 1 基础知识

### 1.1 国外研究现状

Sajeev 等人提出了用 UML 及相关的对象约束语言(OCL)描述构件版本的变更信息<sup>[5,6]</sup>, 构件使用者可以根据构件内被修改和被影响的方法等信息来进行测试用例的选择。该方法虽然简单且易于实现, 但过于粗糙, 会选择出一些与变更无关的用例。Orso 等人在基于构件的软件工程中运用了元数据(metadata)的概念<sup>[7]</sup>, 并在此基础上提出用元内容(metacontent)辅助选择回归测试用例<sup>[8]</sup>。所提出的元方法(metamethod)概念初步运用了内建式测试设计的思想, 但也存在不足。由于该方法将所有变更的信息都作为元数据存储, 其信息难以表示且存储开销大; 由于用例执行时对变更方法的调用并不一定能保证执行到变更点, 所以 Orso 等人提出的元方法策略降低了准确度。

### 1.2 相关定义

文中将构件看作是一个近乎独立、可替换、满足一定功能的模块, 并假定构件由面向对象的模式开发。从其组成角度看, 可进行如下定义:

定义1 构件是一个二元组  $C = (D, M)$ , 其中  $D$

收稿日期: 2008-02-20

基金项目: 安徽省科技攻关重点项目 (06012021A)

作者简介: 陈 峰(1982-), 男, 安徽淮南人, 硕士研究生, 主要研究方向为软件工程、软件测试; 李心科, 副教授, 硕士生导师, 主要研究方向为软件工程、神经网络。

$= (D_w, D_r, D_h)$  是一个数据集,  $D_w$  为可写的成员变量子集,  $D_r$  为可读的成员变量子集,  $D_h$  为不能被外部访问的成员变量子集。  $D_w$  和  $D_r$  共同组成发布变量。  $M = (M_p, M_h)$  是一组方法的集合,  $M_p$  是能被外部调用的方法, 也称发布方法集, 反之则属于私有方法集  $M_h$ 。

定义 2 基于构件的软件也可以表示成一个二元组  $S = (P, S_c)$ , 其中  $P$  为系统开发者自行开发用于组装构件为目的的代码, 可以表示成一些方法的集合。即  $P = \{m_1, m_2, \dots, m_n\}$ ;  $S_c$  为该软件所用到的构件的集合, 即  $S_c = \{C_1, C_2, \dots, C_m\}$ 。

定义 3 构件的方法中被修改的语句称作变更点, 简记作  $CP$ 。由变更点所影响到的方法的集合称作变更信息, 简记作  $CI$ 。

定义 4 构件中的方法调用图为  $MCG(C) = (V, E)$ , 其中,  $V$  表示方法的顶点集, 可分为发布方法子集  $V_p$  和私有方法子集  $V_h$ ;  $E = \{(v_i, v_j) | (v_i, v_j \in V)\}$  表示构件内方法的调用关系。

由构件引起的变更通常表现为如下 3 类: 构件的删除和增加; 构件内方法的修改; 构件内私有变量的删除和增加。构件内方法的修改是构件中最为常见的变更, 它导致的缺陷不容易发现, 文中主要探讨这类情形。为了方便阐述测试技术, 以文献[8]中的售货机应用程序 VendingMachine 作为实例, 源代码请查阅该文献。该构件软件由一个构件 Dispenser 和外部程序 VendingMachine 组成, Dispenser 的代码只对开发者可见, 这里假定 Dispenser 存在两个变更版本: ①  $CP_1$ , 对应方法 Dispense 的 54 行做了修改; ②  $CP_2$ , 对应方法 Available 的 62 行做了修改。

## 2 回归测试用例选择方法

对 Orso 等人的方法做以下改进: 构件提供者仅提供与受影响的发布方法相关的信息给构件使用者; 由于笔者只关心变更点相关的回归测试用例, 因此只需在变更点处内建测试脚本, 并为每个受影响的发布方法构建一个测试接口用以判断用例与变更点的关联性。

文中提出的方法可以从构件开发者和构件使用者两方面考虑。对于构件开发者来说, 第 1 步: 确定变更所影响的发布方法集, 首先构造出整个构件的方法调用图(MCG), 再从变更的方法节点出发逆向遍历该图, 计算出所有可能被影响的发布方法集( $M_p$ )。该过程的算法如图 1 所示; 第 2 步: 根据受影响的发布方法集, 结合变更点为每个发布方法  $m_p$  创建一个形如  $test - m_p(var_1, var_2, \dots, var_k)$  的测试接口, 其中接口方法

的参数  $var_i (1 \leq i \leq k)$  是构件的发布变量或方法  $m_p$  的参数。所以测试接口方法都应设置为发布方法以便构件使用者调用。以下为计算构件版本变更信息的算法(见图 1)。

Algorithm BTd-RTforCI

Input:  $MCG(C)$  is the method call graph of component  $C$ ;  $(m_c, p)$  is the information about change point in component,  $m_c$  is the changed method and  $p$  is the change point in this method.

Output:  $CI = \{[L] M\}$ , the change information of component version.

```

1   $CI = \emptyset$ ;
2  if  $m_c$  is a published method of  $C$  then
3  calculate the pre-condition  $l_0$  of  $m_c$  which ensures
the statement  $p$  to be executed;
4   $CI = CI \cup \{[l_0] m_c\}$ ;
5  endif
6   $worklist = \{m_c\}$ ;
7   $walkedlist = \emptyset$ ;
8  while( $worklist$  is not empty)
9  pick out the first one ( $m$ ) of  $worklist$  and remove
it from  $worklist$  head;
10  $walkedlist = walkedlist \cup \{m\}$ ;
11 if in-degree of  $m$  in  $MCG(C)$  is 0 then
12 continue;
13 endif
14 for each incoming edge of  $m$  in  $MCG(C)$ , denoted as  $(v_i, m_i, l_i)$ 
15 if  $v_i \notin walkedlist$  and  $v_i \notin worklist$  then
16  $worklist = worklist \cup \{v_i\}$ ;
17 endif
18 if  $v_i$  is a published method of  $C$  then
19  $CI = CI \cup \{[l_i] v_i\}$ ;
20 endif
21 endfor
22 endwhile
23 return  $CI$ ;
```

图 1 计算构件版本变更信息的算法

对于文中实例, 假设变更  $CP_1$  发生, 则受影响的发布方法为 dispense, 可按图 2 的方式创建测试接口方法  $test - dispense(int t - credit, int t - sel)$ 。其中,  $flag$  为一信号变量用于标识在某一特定输入下所经历的执行路径是否包含变更点。由于构件 Dispenser 不包含发布变量, 因此方法  $test - dispense$  调用方法  $dispense$  之前没有将发布变量赋值到指定场景的动作。构件使

用户根据实际运行方法 dispense 的场景调用测试接口 test = dispense, 如果返回值为真, 则表明在该用例的执行场景与变更点存在关联, 否则不然。

在测试用例选择的早期准备阶段, 同样要对应应用程序代码进行插装, 记录构件的发布方法在被调用之前方法参数及发布变量的取值。用例选择过程中, 构件使用者根据用例的插装记录调用相对应的内建测试接口方法, 用以判断该用例是否与构件中的变更点相关联。如果返回值为真, 则该用例被选作回归测试用例, 反之不然。例如对于变更 CP2 和第 6 号测试用例 insert, insert, vend(3), 其插装记录为 dispense[credit = 50, sel = 3], 由于在该输入下能经历变更点则测试接口函数 test = dispense(50, 3) 的返回值为真, 所以该用例可以被选用于回归测试。

对于构件使用者如何获得发布方法变更的信息有两种解决方法: 第一种, 在对测试接口方法命名时, 构件开发者和使用者形成某种合约, 使用者根据合约和内建测试接口解析受影响的发布方法。例如文中在受影响的方法前加前缀“test -”形成测试接口方法名, 方法的参数则通过前缀“t -”来表示。第二种, 通过 XML 等标准文件格式存储受影响的发布方法信息, 并随新版本构件一起向构件使用者发布。

### 3 方法的过程实现和实例分析

#### 3.1 过程的实现

构件开发者事先在构件中内建用作回归测试用途的测试脚本, 他们通过传送合约或 XML 文件的方式告知构件使用者变更信息, 构件使用者根据所获得的信息, 选择此策略选择出所需要的用作回归测试的用例。图 2 是基于构件回归测试用例选择的过程, 记为 BTDR - RT。

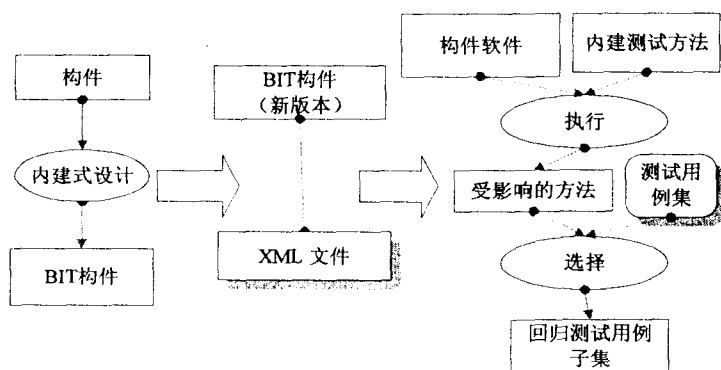


图 2 基于构件回归测试选择的实现过程

#### 3.2 实例分析

为了验证此回归测试用例选择方法的有效性, 仍以文献[8]的例子加以说明, 将 Figure 1 代码转换成 C

# 代码, 在 Visual Studio2005 下调试, 结合 NUnit 单元测试工具, 列出 25 条测试用例来进行回归测试用例选择的测试, 并与 Sajeev 和 Orso 等人的算法相比较得到表 1。

表 1 几种回归测试选择用例结果比较

Subject CBS	Component	Published Methods	Test Cases	Sajeev's method	Orso's method	BTD - RT
VendingMachine(CP1)	1	1	25	20	20	2
VendingMachine(CP1)	1	1	25	20	9	7

从以上结果不难看出, 文中所提出的 BTD - RT 方法比 Sajeev 和 Orso 的方法有明显的改进, 原因是前两种方法过于保守。虽然 Sajeev 的方法简单, 易于实现, Orso 的方法在语句级的实现比较准确, 但 BTD - RT 方法无需创建测试脚本, 且提供的增强的变更信息也比较少, 进一步的研究会使得该方法弥补不足, 得到更大改进。

### 4 结束语

构件软件的测试一直是测试的难点, 文中针对构件变更的情况下, 提出了一种用于回归测试用例的选择方法, 和传统方法进行比较并通过实验分析初步证实了该方法的可行性和有效性。

#### 参考文献:

- [1] Beydeda S. Research in testing COTS components - built - in testing approaches[C]//In: Proc of the 3rd ACS/IEEE Int'l Conf on Computer Systems and Applications. Los Alamitos, CA: IEEE Computer Society Press, 2005: 101 - 104.
- [2] Harrold M J. Testing: A roadmap[C]//In: Proc of the Future of Software Engineering. New York: ACM Press, 2000: 63 - 72.
- [3] 方 菲, 孙家骥, 王立福, 等. 面向对象软件回归测试技术研究[J]. 软件学报, 2001, 12(3): 372 - 376.
- [4] Martins E, Toyota C M, Yanagawa R L. Constructing Self - Testable Software Components [C]//In: Proceedings of the International Conference on Dependable Systems and Networks. Goteborg: [s. n.], 2001: 151 - 160.
- [5] Sajeev A S M, Wibowo B. Regression test selection based on version changes of components[C]//In: Proc of the 10th Asia - Pacific Software Engineering Conference(APSEC'03). Los Alamitos, CA: IEEE Computer Society Press, 2003: 78 - 85.
- [6] Sajeev A S M, Wibowo B. UML modeling for regression testing of component based systems[J]. Electronic Notes in Theoretical Computer Science, 2003, 82 (6): 1 - 9.

静态权责分离(SSD)它定义了相互排斥的角色,在同一个静态权责分离集中的角色不可以分配给同一个用户。静态权责分离可以定义为角色的子集,  $SSD \subseteq 2^R \times K$ , 这里  $K$  是一个  $\geq 2$  的整数集。即该子集中的角色同时分配给同一用户时,必须小于指定个数  $k$ 。

定义 8:静态权责分离集(SSD)。

$$SSD = \{(rs, k) \mid rs \subseteq 2^R \wedge k \geq 2\}$$

$$\forall (rs, k) \in SSD (u \in U \Rightarrow |Role(u) \cap rs| < k)$$

其中‘ $|\dots|$ ’表示集合的元素个数。该定义表示在给用户分配角色时,用户获得的角色集与 SSD 中任何一个集合的交集,其元素个数必须小于指定的  $k$ 。

(3) 动态权责分离(DSD)。

动态权责分离是指一个角色集,该集中的角色在分配给用户时不受限制,但用户不能同时激活这些角色,由于用户可以同时创建多个会话。因此这一约束要求跨越同一用户同一时间创建的所有会话。

定义 9:动态权责分离集(DSD)。

$$DSD = \{(rs, k) \mid rs \subseteq 2^R \wedge k \geq 2\}$$

$$\forall (rs, k) \in DSD (u \in U \Rightarrow |permission(s_i) \cap rs| < k)$$

## 4.2 转授权

转授权判定公式是基于以上几方面进行判断外还要判断系统允许的最大转授权步数,某一角色被转授的次数(width)加以限制。

定义 10:转授权的判定如下:

$$Can\_delegate \subseteq R \times CR \times SSD \times DSD \times N \times Y \times TIME;$$

$$\begin{aligned} dlgt = \{r, cr, ssd, dsd, n, y, (P, [t_b, t_e], L, t_d)\} \\ \in can\_delegate \Leftrightarrow u_{ing} \in \{u \mid users(r'), r' \geq r\} \cap \\ roles\_u(u_{ed}, t) \in cr \cap roles\_u(u_{ed}, t) \in ssd \cap roles\_u(u_{ed}, t) \in dsd \cap n(dlgt) \leq n \cap y(dlgt) \leq y \cap [t_b, t_e] \subseteq valid\_d(u_{ing}, r, t_d) \end{aligned}$$

其中,  $R, CR, SSD, DSD, N, Y$  和  $TIME$  分别是角色、先决条件、静态权责分离,动态权责分离,最大转授权深度、最大转授权宽度、时间的集合;  $n(dlgt)$  表示转授权操作  $dlgt$  中被转授角色  $r$  的再次转授次数,即转授权深度;  $y(dlgt)$  表示当前操作中被转授角色被同一用户转授次数,即转授权宽度。

## 5 结束语

文中在分析访问控制模型的基础上提出了一个基于时间的角色访问控制模型,并对该模型的授权规则进行系统的论述,这是主要工作。正如文中所述,角色访问控制本身具有很多特性,如何正确地刻画这些性质,以及扩充现有模型或提出新模型以支持多种特性,是今后的一个重要工作。

## 参考文献:

- [1] Sandhu R S, Coyne E J, Feinstein H L, et al. Role - Based Access Control Models[J]. IEEE Computer, 1996, 29(2): 38 - 47.
- [2] Sandhu R S, Bhamidipati V, Munawer Q. Role - Based administration of User - Role Assignment: The URA97 Model and Its Oracle Implementation[J]. The Journal of Computer Security, 1999, 2: 105 - 130.
- [3] Sejong O, Sandhu R. A Model for Role Administration using Organization Structure[C]//Proc. 7th ACM Symposium on Access Control Models and Technologies. Monterey, Calif.: [s. n.], 2002: 155 - 162.
- [4] 乔颖, 须德, 戴国忠. 一种基于角色访问控制(RBAC)的新模型及其实现机制[J]. 计算机研究与发展, 2000, 37(1): 37 - 44.
- [5] Hayton R J, Bacon J M, Moody K. OASIS: Access Control in an Open, Distributed Environment [C]//In Proceedings of IEEE Symposium on Security and Privacy. Oakland, CA: [s. n.], 1998: 3 - 14.
- [6] Hine J, Yao W, Bacon J, et al. Architecture for distributed OASIS services[C]//Proc. Middleware 2000, Lecture Notes in Computer Science. New York: Springer - Verlag, 2000: 107 - 123.
- [7] Elisa B, Andrea B P, Elena F. TRBAC: A Temporal Role - Based Access Control Model[J]. ACM Transactions on Information and Systems Security, 2000, 4(3): 21 - 30.
- [8] 黄建, 卿斯汉, 温红子. 带时间特性的角色访问控制[J]. 软件学报, 2003, 14(11): 1944 - 1954.
- [9] Elisa B, Claudio B, Elena F, et al. Atemporal access control mechanism for database systems[J]. IEEE Trans on Knowledge and Data Engineering, 1996, 8(1): 67 - 80.
- [10] Ferraiolo D F, Kuhn D R, Chandramouli R. Role - Based Access Control[M]. London: Artech House, 2003.

(上接第 133 页)

- [7] Orso A, Harrold M J, Rosenblum D. Component metadata for software engineering tasks[C]//In: Proc of Int'l Workshop on Engineering Distributed Objects (EDO), Lecture Notes in Computer Science 1999. Berlin: Springer - Verlag, 2000: 129 - 144.

- [8] Orso A, Harrold M J, Rosenblum D, et al. Using component metacontent to support the regression testing of component - based software[C]//In: Proc of IEEE Int'l Conf on Software Maintenance (ICSM 2001). Los Alamitos, CA: IEEE Computer Society Press, 2001: 716 - 725.