基于覆盖度的回归测试用例选取方法

贺英杰[1],周仁杰[2]
(1. 91404 部队,河北 秦皇岛 066000;
2. 杭州电子科技大学 计算机学院,浙江 杭州 310018)

摘 要:回归测试是软件测试的一个重要阶段,对软件质量的固化起着关键作用。为降低测试成本,提高测试效率,一般选择部分回归,这就需要对回归测试的用例进行约简。测试用例集约简技术已有很多人进行过研究,最早是直接对测试用例集进行约简,后来提出基于测试需求的约简技术。为了对基于测试需求的约简技术进一步优化,提出基于覆盖度的回归测试用例选取方法,并设计相应的 RCSC 算法进行描述。算法主要思想包括:确定重点测试需求集并分解成最小测试需求;对回归测试用例按照优先级进行排序,构建测试用例与测试需求之间的二元关系矩阵,并用覆盖度表示;采用贪婪策略筛选覆盖最小测试需求最多的用例,并将重复的覆盖度置为 0。该方法不但从实际应用角度将原始测试需求分解成最小测试需求,而且对测试用例与测试需求的二元关系矩阵重新定义,提出覆盖度概念,相比于传统方式更直接有效。

关键词:软件测试;回归测试;测试用例选取;重点测试需求集;覆盖度
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2020)10-0101-05
doi:10.3969/j.issn.1673-629X.2020.10.019

A Regression Test Case Selection Method Based on Coverage

HE Ying-jie[1], ZHOU Ren-jie[2]
(1. Unit 91404, Qinhuangdao 066000, China;
2. School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: Regression testing is an important phase of software testing and plays a key role in software quality solidification. In order to reduce test cost and improve test efficiency, partial regression is generally selected, which requires reduction of regression testing cases. A lot of people have studied the test suite reduction technique, first directly reducing test suite, and then proposing the reduction technique based on test requirements. To further optimize the reduction technique based on test requirements, we present a regression test case selection method based on coverage and design the corresponding RCSC to describe. The main ideas of this algorithm include the following aspects: determining the key test requirement set and decomposing them into minimum test requirements; the regression test cases are sorted according to priority, and the binary relation matrix between test cases and test requirements is constructed, which is represented by coverage; the greedy strategy is used to filter the cases that cover the most minimum test requirements, and the repeated coverage is set to 0. The proposed method not only decomposes the original test requirements into the minimum test requirements from the perspective of practical application, but also redefines the binary relation matrix between test cases and test requirements and puts forward the concept of coverage, which is more direct and effective than the traditional method.

Key words: software testing; regression testing; test case selection; key test requirement set; coverage

0 引 言

软件发生变更后需要进行回归测试,软件回归测试是软件测试的一个重要阶段,对软件质量的固化起着关键作用。回归测试一般分为全面回归和部分回归,全面回归的成本高,代价大,为了节约资源成本,提高测试效率,部分回归的执行频度更高[1]。无论是全面回归还是部分回归,都可以通过测试用例集约简技术对测试用例集进行优化,从而删除冗余测试用例,减少测试用例数量,达到提高测试效率的目的[2]。

测试用例集约简技术已有很多人进行过研究,早期有 G 算法[3]、H 算法[4]、GE 算法[5]和 GRE 算法[6]等,这些算法各有特点,已经证实任何一种算法都不比其他算法优越[7]。上述算法都只是对测试用例集的简化策略,未考虑测试需求在测试用例集优化过程中的

收稿日期:2019-12-05 **修回日期**:2020-04-09
基金项目:浙江省自然科学基金(Y17F020152)
作者简介:贺英杰(1981-),男,硕士,工程师,研究方向为软件测试、网络与信息安全。

作用,后来有研究者在上述思想的基础上,将测试用例集优化与测试需求相结合,进一步拓展测试用例集约简技术。例如,文献[8]考虑测试需求集和测试用例集的约简,提出基于概念格分析的约简算法,将概念格理论的可约简属性与可约简对象等概念[9-10]引入进来,但该方法适用 Web 应用,不具备普遍有效性;文献[11]考虑各测试需求间的相互关系,对满足测试需求的所有测试用例进行了划分,生成测试用例集,再利用现有约简技术进行优化;文献[12]先给出测试需求约简模型,对测试需求进行约简,再对测试用例集进行优化;文献[13]也提出了一种测试需求模型,通过建立模型,整理了测试依据,为测试用例设计搭好框架。但这些方法都不是面向回归测试的,没有针对性和适用性,软件回归测试有其特殊性,主要是从首轮测试用例中筛选出重要的、覆盖需求多的用例[14],从而提高回归测试执行效率和质量。文献[15]针对回归测试,提出基于部分需求进行测试用例集优化的方法,但定义不够全面,认为需求集只和软件改动和修正相关,实践发现还应包括通过影响域分析发现的相关需求、与发现问题用例特征吻合的用例所对应的需求,必要时还应包括新增需求。

该文将理论与实际相结合,从工程实践出发,提出基于覆盖度的回归测试用例选取方法—RCSC(a regression test case selection method based on coverage)。该方法面向重点测试需求集,采用贪婪策略筛选用例,能有效降低回归测试成本,提高回归测试的性价比。

1 相关技术介绍

测试用例集约简技术主要分为两类,包括传统的约简技术和基于需求驱动的约简技术。前者是直接对测试用例集进行约简,也称为非需求驱动技术;后者是先对测试需求集进行约简,再利用传统的约简技术对测试用例集进行约简。目前无论哪种技术,传统的约简技术都是测试用例集约简的基础,研究者对此已经提出多种算法,这些算法大致可归为 3 个类别:启发式贪婪搜索、元启发概率优化以及二进制整数线性规划[15]。

启发式贪婪搜索技术一般一次选择一个或多个能覆盖最多数量测试需求的测试用例,排除已经覆盖的测试需求,循环直到测试需求被完全覆盖为止。G 算法、H 算法、GE 算法和 GRE 算法均属于这类技术。

元启发概率优化技术从一个初始的代表用例集(如备选集 T)出发,应用全局概率优化算法推算最优的代表用例集。模拟退火算法和混合遗传算法属于这类算法。

二进制整数线性规划(binary integer linear programming)技术通过将最优用例集选择问题转化为 0-1 整数规划问题,成本开销最小是优化目标,所有测试需求被覆盖是约束条件,最终获得最优用例集。整数规划技术适用于多种约束条件、适应值函数和测试充分性准则,但是时间复杂度高,实际应用中存在局限性。

2 基本概念

定义 1 原始测试需求:一般包括显性测试需求和隐性测试需求,显性测试需求主要通过需求文档、设计说明、用户手册等软件开发文档直接获取;隐性测试需求主要通过相关测试标准规范来获取。原始测试需求标记为 r_{org},可分解为若干条最小测试需求,其集合可用 R_{org} 表示。

定义 2 最小测试需求:利用需求工程、语义分析、测试经验等知识,通过等价类划分等方式将原始测试需求进行分解,得到子测试需求,如果子测试需求的任意真子集都无法充分覆盖原始测试需求的内容,则称该子测试需求为最小测试需求,标记为 r_{min}。原始测试需求和最小测试需求的关系可描述为 $r_{org}=r^1_{min}\cup r^2_{min}\cup\cdots\cup r^l_{min}$,其中 l 为原始测试需求中包含的最小测试需求的数量,有原始测试需求基数 $|r_{org}|=1$。

定义 3 重点测试需求集:回归测试中有必要进行测试的需求项集合,标记为 R_{key},集合中的元素标记为 r_{key}。重点测试需求集 R_{key} 与原始测试需求集 R_{org} 的关系如图 1 所示。

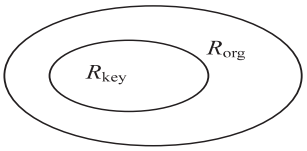

图 1 R_{key} 与 R_{org} 关系图

重点测试需求集包括 4 部分,分别为问题需求集、相关需求集、相似需求集以及新增需求集。问题需求集为包含测试问题的测试需求集合,标记为 R^1_{key};相关需求集为通过影响域分析后修改的测试需求集合,标记为 R^2_{key};相似需求集为与发现问题的测试需求的特征高度吻合的需求集合,标记为 R^3_{key};新增需求集为软件改动后新增的功能、性能、接口、人机等测试需求集,标记为 R^4_{key}。$R_{key}=R^1_{key}\cup R^2_{key}\cup R^3_{key}\cup R^4_{key}=\{r_{key(1)},r_{key(2)},\cdots,r_{key(n)}\}$,且 $|R_{key}|=n$,即重点测试需求集的基数为 n。

定义 4 回归测试用例集:重点测试需求集所对应的测试用例集合,标记为 T,包含问题用例集 T_1、相关用例集 T_2、相似用例集 T_3、新增用例 T_4。$T=T_1\cup T_2\cup T_3\cup T_4=\{t_1,t_2,\cdots,t_m\}$,且 $|T|=m$,即回归测试

用例集的基数为 m 。

定义 5 二元关系矩阵:描述回归测试用例集 T 对重点测试需求集 R_{key} 的覆盖关系,矩阵的行代表重点测试需求,矩阵的列代表回归测试用例。矩阵元素定义为 $a(t_i,r_{key(j)})$,在这里称之为覆盖度,其定义如式(1)所示。

$$a(t_i,r_{key(j)})=\begin{cases}\mathrm{cov}_j(x), t_i \text{ covered } r_{key(j)}\\ 0, t_i \text{ ignored } r_{key(j)}\end{cases} \quad (1)$$

如果 $t_i \in T$ 覆盖 $r_{key(j)} \in R_{key}$,则 $a(t_i,r_{key(j)})=\mathrm{cov}_j(x)$,否则 $a(t_i,r_{key(j)})=0$,其中 x 为被当前用例覆盖的最小测试需求序号,取值范围为 $1,2,\cdots,|r_{key(j)}|$ ($|r_{key(j)}|$ 表示第 j 个重点测试需求所包含的最小测试需求的数量), $i=1,2,\cdots,m$, $j=1,2,\cdots,n$ 。

3　方法描述

3.1　选取策略

进行回归测试时,先确定重点测试需求集 R_{key} ,一般包括 4 部分,分别为问题需求集、相关需求集、相似需求集以及新增需求集,其中新增需求集需根据软件具体整改情况来确定,不是每次软件整改都会新增需求,在这里考虑最大情况,将新增需求集纳入方法描述中。一般根据问题用例集 T_1 引出问题需求集 R^1_{key} ;通过影响域分析,列出改动的需求集 R^2_{key} ,但不包含问题需求,即 R^2_{key} 不包含 R^1_{key} ;通过对问题用例的特征分析,找出与问题用例相似的用例集 T_3 ,列出所对应的需求 R^3_{key} ;考虑到特殊情况下,软件整改后可能会增加新的需求,引入新增需求集 R^4_{key} ,并新设计测试用例集 T_4 。R^1_{key} 、R^2_{key} 、R^3_{key} 、R^4_{key} 组成重点回归测试需求集 R_{key} , T_1 、T_2 、T_3 、T_4 组成原始的回归用例集 T 。这样问题就变成对测试用例集 T 求最优解,以尽可能少的用例覆盖所有关注的需求。

其次,通过语义分析、测试经验等知识将重点测试需求分解成最小测试需求,计算重点测试需求的基数,即包含的最小测试需求的数量;构建二元关系矩阵表示回归测试用例对重点测试需求的覆盖关系,行代表用例,列代表需求,默认用例已根据优先级排序;依次标注出用例对每一重点测试需求的覆盖度,如果用例覆盖需求,则覆盖度标记为 $\mathrm{cov}_j(x)$,即第 j 个用例所对应的第 x 个最小测试需求,否则标记为 0;比较每个用例的覆盖情况,筛选覆盖需求最多的测试用例,放入最优回归测试集中,如果出现多个用例并列的情况,则比较用例的优先级,优先级高的用例置为最优回归测试用例;在关系矩阵中删除最优回归测试用例及其对应的覆盖度,将其他用例所对应的重复的覆盖度置为 0,调整重点测试需求的基数,减去被覆盖的最小测试需求的项数,从比较每个用例的覆盖情况开始重复上述操作,直到所有重点测试需求的基数变为 0,表明所有最小测试需求均已覆盖。

3.2　RCSC 算法

输入: T_1 ,问题用例集; R^2_{key} ,相关需求集; T_3 ,相似用例集; R^4_{key} ,新增需求集; T ,回归测试用例集; R_{key} ,重点测试需求集; $S(T,R_{key})$,从 T 到 R_{key} 的二元关系矩阵;

输出: T' ,最优回归测试用例集;

变量: R^1_{key} ,问题需求集; T_2 ,相关用例集; R^3_{key} ,相似需求集; T_4 ,新增用例集; t_i ,回归测试用例; $r_{key(j)}$,重点测试需求; $\mathrm{cov}_j(x)$,覆盖度;covT,覆盖需求最多的用例集。

```
begin
if( T1≠ø) //存在问题用例集
foreach ti ∈ T1 do
列出 ti 所对应的测试需求 r_key(j) ;
确定问题需求集 R^1_key ;
end for
end if
if ( R^2_key ≠ø)//相关需求集
复用 R^2_key 所包含的用例集 T2;
end if
if( T3≠ø)//相似用例集 T3
列出 T3 所对应的相似需求集 R^3_key ;
end if
if( R^4_key ≠ø)//软件新增功能
设计新增用例集 T4;
end if
//列出重点测试需求集组成
R_key = R^1_key ∪ R^2_key ∪ R^3_key ∪ R^4_key = {r_key(1),r_key(2),…,r_key(n)} ,且| R_key | = n ;
//列出回归测试用例集元素组成
T = T1 ∪ T2 ∪ T3 ∪ T4 = {t1,t2,…,tm} ,且| T | = m ;
//构建从 T 到 R_key 的二元关系矩阵, i = 1,2,…,m , j = 1,2,…,n
S ( T , R_key )= { a(ti,r_key(j)) ∈ T × R_key } ;
foreach ti ∈ T do
foreach r_key(j) ∈ R_key do
if( ti 覆盖 r_key(j) )
( ti , r_key(j) )= cov_j(x) ;// x 为被当前用例覆盖的最小测试需求序号
else
a(ti,r_key(j)) = 0;
end if
end for
end for
while(| r_key(j) |≠0)//重点测试需求基数不全为0, j = 1,2,…,n
```

covT = { t_i };//查找覆盖需求最多的用例(集);
if(|covT| >1)//多个用例并列,查看优先级
t' = min(t_i);//序号最小的优先级高
T' = T' ∪ { t_i }//放入最优回归测试用例集中;
end if
deletet' and $\mathrm{cov}_j(i)$ //删除最优回归测试用例及其对应的覆盖度;
for(i =1; $i \leqslant m$; i ++)
for(j =1; $j \leqslant n$; n ++)
if($\mathrm{cov}_j(i)$ 重复)//其他用例的覆盖度重复,则置为 0
$\mathrm{cov}_j(i)$ = 0;
end if
end for
end for
| $r_{\mathrm{key}(j)}$ |--//相关重点测试需求基数减 1;
end while

4 实例分析

为了说明该方法的有效性,本节通过一个实例对回归测试用例的选取过程进行演示。回归测试用例的选取流程如图 2 所示。

进行如下假定:

(1)问题用例集 $T_1 = \{t_1, t_2, \cdots, t_5\}$,问题需求集 $R^1_{\mathrm{key}} = \{r_{\mathrm{key}(1)}, r_{\mathrm{key}(2)}\}$;相关需求集 $R^2_{\mathrm{key}} = \{r_{\mathrm{key}(3)}\}$,相关用例集 $T_2 = \{t_6, t_7\}$;相似用例集 $T_3 = \{t_8\}$,相似需求集 $R^3_{\mathrm{key}} = \{r_{\mathrm{key}(4)}\}$;新增需求集 $R^4_{\mathrm{key}} = \{r_{\mathrm{key}(5)}\}$,新增用例集 $T_4 = \{t_9\}$;

(2) $r_{\mathrm{key}(1)}$ 、$r_{\mathrm{key}(2)}$ 、$r_{\mathrm{key}(3)}$ 、$r_{\mathrm{key}(4)}$ 、$r_{\mathrm{key}(5)}$ 包含最小测试需求的数量分别为 3、2、2、1、1,即 | $r_{\mathrm{key}(1)}$ | = 3、| $r_{\mathrm{key}(2)}$ | = 2、| $r_{\mathrm{key}(3)}$ | = 2、| $r_{\mathrm{key}(4)}$ | = 1、| $r_{\mathrm{key}(5)}$ | = 1。

选取的具体步骤说明如下:

(1)发现问题的用例必须复用,根据问题用例集 T_1 列出相应的问题需求集 R^1_{key} ;通过影响域分析,找出被影响的相关需求集 R^2_{key} ,复用对应的相关用例集 T_2;对问题用例进行特征分析,找出相似用例集 T_3,继而得到相似需求集 R^3_{key} ;引入新增需求集 R^4_{key} ,设计新增用例集 T_4;

(2)列出重点测试需求集和回归测试用例集元素组成,其中 $R_{\mathrm{key}} = R^1_{\mathrm{key}} \cup R^2_{\mathrm{key}} \cup R^3_{\mathrm{key}} \cup R^4_{\mathrm{key}} = \{r_{\mathrm{key}(1)}, r_{\mathrm{key}(2)}, \cdots, r_{\mathrm{key}(5)}\}$,且 $|R_{\mathrm{key}}| = 5$, $T = T_1 \cup T_2 \cup T_3 \cup T_4 = \{t_1, t_2, \cdots, t_9\}$,且 $|T| = 9$,将各重点测试需求 $r_{\mathrm{key}(1)}$ 、$r_{\mathrm{key}(2)}$ 、$r_{\mathrm{key}(3)}$ 、$r_{\mathrm{key}(4)}$ 、$r_{\mathrm{key}(5)}$ 分解成最小测试需求,计算重点测试需求的基数,分别为 3、2、2、1、1;

(3)构建从 T 到 R_{key} 的二元关系矩阵,矩阵大小为 9×5,默认 t_i 按照优先级排好序,如果 t_i 覆盖 $r_{\mathrm{key}(j)}$,则相应的覆盖度置为 $\mathrm{cov}_j(x)$,即满足第 j 个重点测试需求中第 x 条最小测试需求,否则置为 0。具体覆盖关系如表 1 所示。

表 1 选取前测试用例与测试需求的覆盖关系

	$r_{\mathrm{key}(1)}$	$r_{\mathrm{key}(2)}$	$r_{\mathrm{key}(3)}$	$r_{\mathrm{key}(4)}$	$r_{\mathrm{key}(5)}$
t_1	$\mathrm{cov}_1(1)$	$\mathrm{cov}_2(1)$	0	0	0
t_2	$\mathrm{cov}_1(2)$	0	0	0	0
t_3	$\mathrm{cov}_1(3)$	0	$\mathrm{cov}_3(1)$	$\mathrm{cov}_4(1)$	0
t_4	0	$\mathrm{cov}_2(1)$	0	0	0
t_5	0	$\mathrm{cov}_2(2)$	0	0	0
t_6	$\mathrm{cov}_1(2)$	$\mathrm{cov}_2(2)$	$\mathrm{cov}_3(1)$	0	0
t_7	0	0	$\mathrm{cov}_3(2)$	0	0
t_8	0	0	0	$\mathrm{cov}_4(1)$	0
t_9	0	0	0	0	$\mathrm{cov}_5(1)$

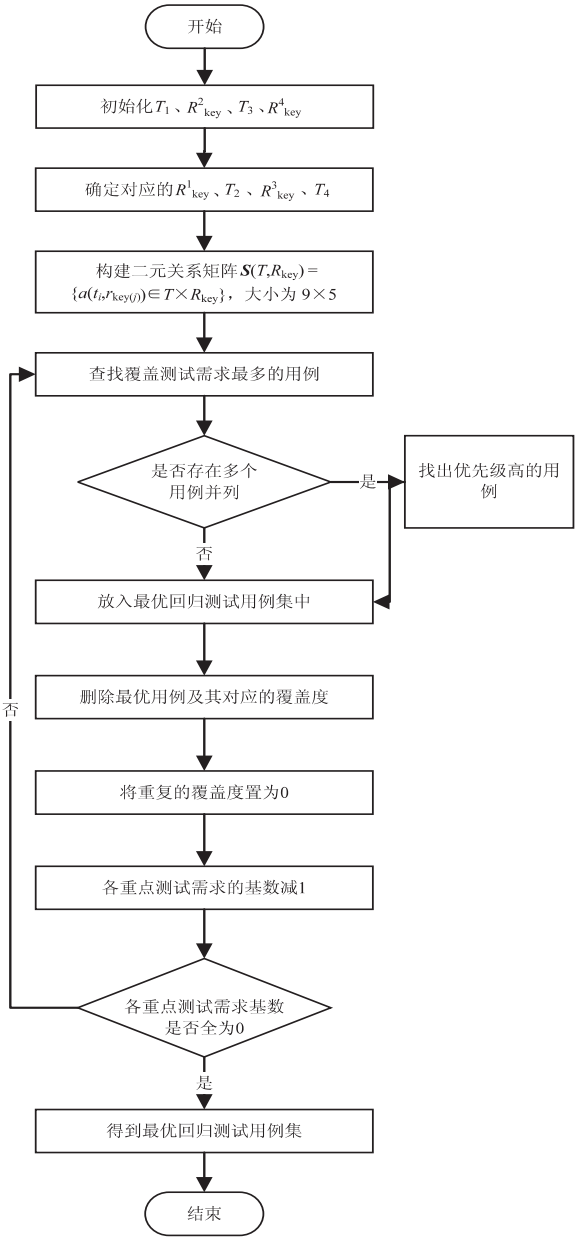

图 2 回归测试用例的选取流程

(4)比较每个测试用例 $t_1 \sim t_9$ 的覆盖情况,筛选覆盖测试需求最多的测试用例,放入最优回归测试集中,

如果出现多个用例并列的情况,则比较用例的优先级,优先级高的用例置为最优回归测试用例。例如,表 1 中 t_3 和 t_6 覆盖的需求数一样多,但是 t_3 优先级高,所以将 t_3 放入最优回归测试用例集中,在关系矩阵中删除 t_3 及其对应的覆盖度,将 t_6 和 t_8 中重复的覆盖度置为 0,如表 2 所示,t_3 及其对应的覆盖度带下划线,表示已被删除,t_6 和 t_8 中数字 0 带下划线,表示覆盖度重复被置 0;同时将 $r_{key(1)}$、$r_{key(3)}$、$r_{key(4)}$ 的基数减 1,各重点测试需求基数分别变更为 2、2、1、0、1。

表 2　首次筛选后的覆盖关系

	$r_{key(1)}$	$r_{key(2)}$	$r_{key(3)}$	$r_{key(4)}$	$r_{key(5)}$
t_1	$cov_1(1)$	$cov_2(1)$	0	0	0
t_2	$cov_1(2)$	0	0	0	0
t_3	$cov_1(3)$	0	$cov_3(1)$	$cov_4(1)$	0
t_4	0	$cov_2(1)$	0	0	0
t_5	0	$cov_2(2)$	0	0	0
t_6	$cov_1(2)$	$cov_2(2)$	0	0	0
t_7	0	0	$cov_3(2)$	0	0
t_8	0	0	0	0	0
t_9	0	0	0		$cov_5(1)$

(5)判断重点测试需求基数是否全为 0,如果否,则重复步骤(4)中的筛选覆盖测试需求最多的测试用例等相关操作,比较剩余用例的覆盖情况,直到所有重点测试需求的基数变为 0,得到最终覆盖关系如表 3 所示,表中用例集合即为最优回归测试用例集。

表 3　最终覆盖关系

	$r_{key(1)}$	$r_{key(2)}$	$r_{key(3)}$	$r_{key(4)}$	$r_{key(5)}$
t_1	$cov_1(1)$	$cov_2(1)$	0	0	0
t_3	$cov_1(3)$	0	$cov_3(1)$	$cov_4(1)$	0
t_6	$cov_1(2)$	$cov_2(2)$	0	0	0
t_7	0	0	$cov_3(2)$	0	0
t_9	0	0	0	0	$cov_5(1)$

从最后的筛选结果可以看出,用例集 $\{t_1, t_3, t_6, t_7, t_9\}$ 可以满足所有测试需求,从而达到了用最少回归测试用例覆盖重点测试需求的目的。

5　结束语

测试用例优化的目的是用最小的用例集达到最大的覆盖率,对于回归测试来说,从时间、人力等成本的角度考虑,不需要达到测试需求的全覆盖,满足最优覆盖即可,即覆盖重点测试需求,实现用最小的代价达到最优的目标;对于回归测试用例约简方面,该文不但从实际应用角度将原始测试需求分解成最小测试需求,而且对测试用例与测试需求的二元关系矩阵重新定义,提出覆盖度概念,摒弃简单的 1-0 关系表示,便于剔除重复的覆盖关系,相比于传统方式更直接有效。

该方法适用于规模大、进度要求高的软件系统测试,既可减少回归测试的工作量,又不降低回归测试的质量,能够在用例数量与软件质量之间达到一个平衡。下一步需要对原始测试需求分解标准、相似用例选取方法等方面进行深入研究,提出详细的标准规范,进一步完善方法的体系结构。

参考文献:

[1] 程晓菊. 测试用例集约简技术研究[D]. 长沙:湖南大学,2011.

[2] 章晓芳,陈　林,徐宝文,等. 测试用例集约简问题研究及其进展[J]. 计算机科学与探索,2008,2(3):235-247.

[3] JOHNSON D S. Approximation algorithms for combinatorial problems[J]. Journal of Computer and System Sciences, 1974,9(3):256-278.

[4] HARROLD M J, GUPTA R, SOFFA M L. A methodology for controlling the size of a test suite[J]. ACM Transactions on Software Engineering and Methodology, 1993,2(3):270-285.

[5] CHEN T Y, LAW M H. Dividing strategies for the optimization of a test suite[J]. Information Processing Letters, 1996, 60(3):135-141.

[6] CHEN T Y, LAU M F. A new heuristic for test suite reduction[J]. Information and Software Technology, 1998, 40(5/6):347-354.

[7] CHEN T Y, LAU M F. A simulation study on some heuristics for test suite reduction[J]. Information and Software Technology, 1998, 40(13):777-787.

[8] TALLAM S, GUPTA N. A concept analysis inspired greedy algorithm for test suite minimization[C]//Proceedings of the 6th ACM SIGPLAN-SIGSOFT workshop on program analysis for software tools and engineering. New York, NY, United States: Association for Computing Machinery, 2005:35-42.

[9] ZHANG Wenxiu, WEI Ling, QI Jianjun. Attribute reduction theory and approach to concept lattice[J]. Science in China Series F: Information Sciences, 2005, 35(6):628-639.

[10] GANTER B, WILLE R. Formal concept analysis mathematical foundations[M]. Berlin: Springer, 1999.

[11] 聂长海,徐宝文. 一种最小测试用例集生成方法[J]. 计算机学报,2003,26(12):1690-1695.

[12] 章晓芳,徐宝文,聂长海,等. 一种基于测试需求约简的测试用例集优化方法[J]. 软件学报,2007,18(4):821-831.

[13] 哈清华,刘大有,沈湘衡,等. 基于需求模型的航天软件测试用例生成方法[J]. 光学精密工程,2016,24(5):1185-1196.

[14] 张智轶,陈振宇,徐宝文,等. 测试用例演化研究进展[J]. 软件学报,2013,24(4):663-674.

[15] 顾　庆,唐　宝,陈道蓄. 一种面向测试需求部分覆盖的测试用例集约简技术[J]. 计算机学报,2011,34(5):879-888.