ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.821−831
DOI: 10.1360/jos180821

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于测试需求约简的测试用例集优化方法*

章晓芳[1,2+], 徐宝文[1,2], 聂长海[1,2], 史 亮[1,2]

[1](东南大学 计算机科学与工程学院,江苏 南京 210096)

[2](江苏省软件质量研究所,江苏 南京 210096)

An Approach for Optimizing Test Suite Based on Testing Requirement Reduction

ZHANG Xiao-Fang[1,2+], XU Bao-Wen[1,2], NIE Chang-Hai[1,2], SHI Liang[1,2]

[1](School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](Jiangsu Institute of Software Quality, Nanjing 210096, China)

+ Corresponding author: Phn: +86-25-52090881, E-mail: xfzhang@seu.edu.cn

Zhang XF, Xu BW, Nie CH, Shi L. An approach for optimizing test suite based on testing requirement reduction. *Journal of Software*, 2007,18(4):821−831. http://www.jos.org.cn/1000-9825/18/821.htm

Abstract: Test suite optimization aims at satisfying all testing objectives with the least number of test cases. According to the given testing objectives, the reduced testing requirement set can improve the effectiveness and efficiency of test suite optimization. This paper proposes a testing requirement reduction model that can describe the interrelations among the testing requirements in detail. Based on the model, this paper presents a testing requirement reduction method to generate the reduced testing requirement set, which is the basis of test suite generation, reduction and optimization. The experimental results show that the method is helpful to generate the smaller test suite and it contributes to the systematic, reasonable and effective testing.

Key words: software testing; white-box testing; structural testing; testing requirement; test suite optimization

摘 要: 测试用例集优化的目标是用尽可能少的测试用例充分满足给定的测试目标.针对给定的测试目标,获得精简的测试需求集有助于提高测试用例集优化的效率和效果.从测试需求约简的角度考虑测试用例集优化,首先给出可以精确描述测试需求间相互关系的测试需求约简模型;基于此模型,提出一种测试需求约简方法,可以获得精简测试需求集,作为测试用例集生成和约简的基础,从而实现测试用例集优化.实验结果表明,测试需求约简有助于获得规模较小的测试用例集,实现系统、科学、有效的测试.

关键词: 软件测试;白盒测试;结构测试;测试需求;测试用例集优化

中图法分类号: TP311 文献标识码: A

在进行软件测试时,测试人员首先要确定测试目标,一个测试目标可以表示为一组测试需求.例如:给定语

* Supported by the National Natural Science Foundation of China under Grant Nos.60403016, 60633010 (国家自然科学基金); the National Science Foundation for Distinguished Young Scholars of China under Grant No.60425206 (国家杰出青年科学基金); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2005060 (江苏省自然科学基金); the Excellent Talent Foundation on Teaching and Research of Southeast University of China (东南大学优秀青年教师教学科研资助)

Received 2005-12-14; Accepted 2006-04-05

句覆盖的测试目标,则待测程序中的每一条语句均对应一项测试需求.针对测试需求集,最初设计生成的测试用例集中,往往存在冗余的测试用例,有必要进行测试用例集约简.由于软件开发过程是一个反复迭代的过程,需要不断进行测试,其中,测试用例的设计、执行、管理和维护的开销相当大,因此,测试用例集优化方法成为软件测试领域中的一项关键课题.其目标是使用尽可能少的测试用例,充分满足给定的测试目标,从而提高测试效率,降低测试成本.

测试用例集优化方法主要包括测试用例集的生成和约简方法.许多研究者针对各种测试用例集优化方法进行了深入的研究[1−9].然而,现有的测试用例集优化方法大多不考虑测试需求间存在的复杂的相互关系,而是基于给定测试目标所对应的测试需求集,直接生成、约简测试用例集.在测试实践中,针对原测试需求集,若能获得精简的测试需求集,将有助于提高测试用例集优化的效率和效果.事实上,对于原测试需求集 R,可能存在某个测试需求集 R'($|R'|\leq|R|$),针对 R'设计的测试用例集 T,可以同时满足 R.例如:在结构化测试中,针对语句块覆盖标准所对应的测试需求集 R_c 设计的测试用例集 T,必然可以满足语句覆盖标准所对应的测试需求集 R_s($|R_c|\leq|R_s|$).因此,R_c 可以代替 R_s,此时,仅需考虑满足测试需求集 R_c.再者,例如在强度测试和性能测试中,测试人员可能都需要测试系统在最大负载下的响应时间.此时,这两项具体的测试需求其实是对同一项测试内容的描述,可以认为是等价的,应该予以合并.一般情况下,精简的测试需求集 R'的规模远小于原测试需求集 R 的规模.

基于以上讨论,本文提出一种基于测试需求约简的测试用例集优化方法.首先约简测试需求集,以获得精简测试需求集;然后,针对精简测试需求集,采用已有的测试用例集生成和约简方法,获得优化测试用例集.该方法是现有测试用例集优化方法的有力补充.其中,如何获得精简测试需求集是本文的研究重点.为此,本文首先给出可以精确地描述测试需求间相互关系的测试需求约简模型.基于此模型,提出一种测试需求约简方法,可以获得精简测试需求集.实验分析表明:针对精简测试需求集设计测试用例,不但可以减少生成、约简测试用例的工作量,还有助于获得规模较小的测试用例集.

1 相关工作及存在问题

设原测试需求集 $R=\{r_1,r_2,\ldots,r_m\}$,针对 R 生成测试用例集 $T=\{t_1,t_2,\ldots,t_n\}$.现有测试用例集优化方法将采用测试用例集约简方法,寻找 T 的某个子集,用尽可能少的测试用例满足测试需求集 R.现有测试用例集约简方法主要包括贪心算法、一些启发式算法、整数规划等方法.

贪心算法[1](简称 G 算法)逐次地从 T 中选择一条测试用例,使之能最多地满足 R 中的测试需求,然后从 R 中删除这些测试需求,直到所有测试需求都被满足(R 为空集).该算法的最坏时间复杂度为 $O(mn\cdot\min(m,n))$.在此基础上,Chen 等人提出了 GE 和 GRE 算法[2,3].GE 算法是对贪心算法的改进,先找出必不可少的测试用例,再使用贪心算法,其最坏时间复杂度仍为 $O(mn\cdot\min(m,n))$.GRE 算法反复地交替使用必不可少策略和 1-1 冗余策略,直到这两种策略都无法应用,再使用贪心算法选择测试用例满足剩下的测试需求.GRE 算法的最坏时间复杂度为 $O(\min(m,n)(m+n^2k))$[2],其中,k 表示一个测试用例最多能满足的测试需求的数量.

Harrold 等人提出了一种根据测试用例的重要性来选择测试用例的启发式算法(简称 H 算法)[1].该算法将测试需求 $r_1,r_2,\ldots,r_m$ 划分到集合 $R_1,R_2,\ldots,R_d$($d\leq n$),其中,R_i(i=1,2,…,d)包含所有正好可以被 T 中 i 条测试用例满足的测试需求.如果 $i<j$,则 H 算法认为满足 R_i 中测试需求的测试用例比满足 R_j 中测试需求的测试用例要“重要”.因此,H 算法首先选出满足 R_1 中测试需求的测试用例.然后考虑 R_2,使用贪心算法选择测试用例,直到 R_2 中的测试需求全部被满足.依次处理 $R_3,R_4,\ldots$,直到 R_d.该算法的最坏时间复杂度为 $O(m(m+n)d)$.

Lee 等人所提出的测试用例选择方法把测试用例选择问题转化为整数规划问题,利用整数规划方法求出最优解[4],在理论上可以获得满足测试需求集 R 的最小测试用例集,但其计算复杂程度较高,运算开销呈指数级增长.

上述测试用例集约简方法均基于已知的测试需求和测试用例的满足关系,直接约简测试用例集.它们忽视了测试需求间的相互关系,没有考虑到可能存在一个精简的测试需求集 R',它的规模小于原测试需求集 R,然而,满足 R'的测试用例可以同时满足 R.因此,本文从测试需求约简出发考虑测试用例集约简:首先约简测试需求集,

获得精简测试需求集 R';然后,基于 R'和已知的满足关系来约简测试用例集.该方法可以有效地降低后续约简计算的工作量,且有可能获得较上述方法更好的测试用例集约简结果.

在测试需求约简方面,Marre 等人已提出扩张集技术用于约简测试需求[5].针对给定的结构化测试覆盖标准,她们利用程序分析的方法计算测试需求间的包含关系,以约简测试需求集.该方法主要适用于单元测试的测试用例集优化.本文给出通用的测试需求约简模型和方法,不仅可以应用于结构化测试,还可以广泛应用于其他各种测试领域.

在前人工作的基础上,我们曾提出一种最小测试用例集生成方法[6].该方法充分考虑了测试需求间的相互关系,对测试需求所对应的测试用例集进行了划分.当选择测试用例时,尽可能地在测试需求对应的可用测试用例集的交集中选择,得到了良好的结果.在此基础上,本文进一步研究如何充分利用测试需求间的相互关系来约简测试需求,获得精简测试需求集,作为测试用例集生成和约简的基础,进行有效的测试用例集优化.

2 测试需求约简模型

测试需求约简作为测试用例集优化的前期处理,其目标是约简测试需求集,减少待处理的测试需求数量,并降低测试需求间关系的复杂度,从而有利于测试用例集的生成与约简.测试需求间的相互关系是测试需求约简的基础,可以通过需求工程、语义分析、程序分析、领域知识、测试历史和测试人员经验等获得.由于每项可测试的测试需求实际上对应着一个非空的可用测试用例集,该测试用例集中的任意一条测试用例都可以满足该测试需求.所以,我们可以通过相应的测试用例集间的相互关系来描述测试需求间的相互关系,而且,描述测试需求间的相互关系并不一定需要生成具体的测试用例,只需要获知相应测试用例的分布情况.下面给出一个可以精确描述测试需求间相互关系的通用的测试需求约简模型.

定义 **1**. 对于待测软件系统 SUT,定义:

(1) 测试需求 r 到测试用例集 T 的映射 $Test(r)$={所有可以满足 r 的测试用例 t}.

(2) 测试用例集 T 到测试需求 r 的映射 $Req(T)=r$,其中:$\forall t\in T,t$ 可以满足 r,且$\forall t\notin T,t$ 不能满足 r.

由 $Test(r)$可以定义测试需求集 R 到测试用例集 T 的映射 $Test(R)=\bigcup_{r\in R}Test(r)$.若测试需求 r 到测试用例集 T 的映射结果为空集,即 $Test(r)=\varnothing$,则认为测试需求 r 为空,即 $r=\varnothing$;反之亦然.

定义 1 描述了测试需求和测试用例集的对应关系,每项测试需求对应着一个非空的测试用例集.因此,对于测试需求 $r_i,r_j\in R$,$T_i=Test(r_i)$,$T_j=Test(r_j)$,我们可以进一步定义测试需求的交、并运算:

$$r_i\cap r_j=Req(T_i\cap T_j),$$

$$r_i\cup r_j=Req(T_i\cup T_j).$$

定义 **2**. 对于测试需求 $r_i,r_j\in R$,定义:

(1) 若 $r_i\cap r_j=r_i$,则称 r_i 包含于 r_j,记为 $r_i\subseteq r_j$.

(2) 若 $r_i\subseteq r_j$ 且 $r_j\subseteq r_i$,则称 r_i,r_j 是等价的测试需求,记为 $r_i\equiv r_j$.

(3) 若 $r_i\cap r_j\neq\varnothing$,则称 r_i,r_j 是部分重合的测试需求,记为 $r_i\oplus r_j$.

(4) 若 $r_i\cap r_j=\varnothing$,则称 r_i,r_j 是相互独立的测试需求,记为 $r_i><r_j$.

定义 **3**. 对于测试需求集 R,R',若$|R'|\leq|R|$,$\forall r\in R,\exists r'\in R',r'\subseteq r$,且$\forall r'\in R',\exists r\in R,r'\subseteq r$,则称测试需求集 R'为 R 的精简测试需求集.若不存在满足$|R''|<|R'|$的精简测试需求集 R'',则称 R'为 R 的最小精简测试需求集,记为 R_m.

由于获取最小精简测试需求集的开销往往较大,我们通常采用启发式方法,根据测试需求间相互关系来获得精简测试需求集 R'.对于测试需求集 $R=\{r_i,r_j\}$,有如下测试需求约简原则:

(1) 当 $r_i\subseteq r_j$ 时,若在 T_i 中选取 t_i,则 t_i 可以同时满足 r_i,r_j,即满足了 r_i 的测试用例必然可以同时满足 r_j.因此,可以约简 r_j,保留 r_i,$R'=\{r_i\}$.

(2) 当 $r_i\equiv r_j$ 时,无论在 T_i 中选取 t_i 或是在 T_j 中选取 t_j 都可以同时满足 r_i,r_j,此时,可删除测试需求 r_i,r_j 中的任意一项.因此,仅保留 r_i(或 r_j),$R'=\{r_i\}$(或 $R'=\{r_j\}$).

(3) 当 $r_i\oplus r_j$ 时,满足 r_i 的测试用例不一定同时满足 r_j;反之亦然.然而,满足 $r_i\cap r_j$ 的测试用例却可以同时满

足 r_i,r_j.因此,将 r_i,r_j 约简为 $r_i \cap r_j$,即 $R'=\{r_i \cap r_j\}$.

(4) 当 $r_i><r_j$ 时,需要分别从 T_i,T_j 中选取 t_i,t_j 来满足 r_i,r_j.因此,r_i,r_j 无法约简,$R'=\{r_i,r_j\}$.

基于上述测试需求约简原则,对于如图 1 所示的测试需求集 $R=\{r_1,r_2,r_3,r_4,r_5\}$,可进行如下约简:由于 $r_4 \equiv r_5$,则不妨保留 r_4;由于 $r_2 \subseteq r_4$,则保留 r_2;由于 $r_2 \oplus r_3$,则保留 $r_2 \cap r_3$;由于 $r_1><r_i(i=2,3,4,5)$,则保留 r_1.于是可得精简测试需求集 $R'=\{r_1,r_2 \cap r_3\}$,$|R'|=2$.因此,在后续生成或约简测试用例时,仅需要考虑这两项测试需求.在生成测试用例时,至少需要一条测试用例 t_1 来满足 r_1,一条测试用例 t_2 来满足 $r_2 \cap r_3$.由于 t_2 可以同时满足测试需求 r_2,r_3,r_4,r_5,于是满足了精简测试需求集 R' 的测试用例集 $\{t_1,t_2\}$,同时满足了原测试需求集 R.另一方面,若已知测试用例与测试需求的满足关系如表 1 所示,则利用 $R'=\{r_1,r_2 \cap r_3\}$ 可以将满足关系表的尺寸从 5×6 降至 2×2.采用现有的测试用例集约简方法[1−4],可得约简后的测试用例集为 $\{test_1,test_2\}$.该实例表明:测试需求约简可以有效减小待处理的测试需求数量,并降低测试需求间关系的复杂度,有助于减少后续生成或约简测试用例的计算开销.

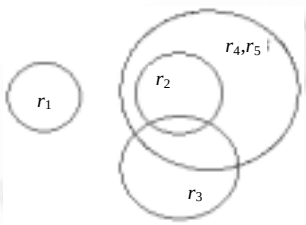


Fig.1 Relationship of testing requirements

图 1 测试需求间关系

Table 1 Satisfiability relation of test cases and testing requirements

表 **1** 测试用例与测试需求的满足关系

r / Test	r_1	r_2	r_3	r_4	r_5	r_1	$r_2 \cap r_3$
$test_1$	×					×	
$test_2$		×	×	×	×		×
$test_3$		×		×	×		
$test_4$			×	×	×		
$test_5$			×				
$test_6$				×	×		

3 测试用例集优化方法

在测试需求约简模型的基础上,我们提出一种基于测试需求约简的测试用例集优化方法 TSO_TRR.该方法首先利用包含和部分重合关系约简测试需求,获得精简测试需求集.然后基于精简测试需求集,采用现有测试用例集生成或约简方法,实现测试用例集优化.本文的研究重点在于如何约简测试需求.

3.1 测试需求约简

本节提出 3 种具体的测试需求约简方法:TRR_Sub,TRR_Greedy 和 TRR_Linear.在测试需求约简过程中,测试需求间的包含关系较易获取,故首先使用 TRR_Sub 利用包含关系约简测试需求.在此基础上,利用部分重合关系进一步约简测试需求,针对不同的测试需求集规模和测试资源约束,我们提出基于贪心算法的测试需求约简方法 TRR_Greedy 和基于线性搜索的测试需求约简方法 TRR_Linear.

TRR_Sub 利用包含关系约简原测试需求集 $R=\{r_1,r_2,\ldots,r_m\}$,获得精简测试需求集 R'.如算法 1 所示,TRR_Sub 首先生成测试需求的两两组合对的集合 $Pair$,$|Pair|=C_m^2$;然后,从集合 $Pair$ 中任意取出一个组合对,判断组合对中两项测试需求的相互关系,根据测试需求约简原则约简测试需求.该算法的步骤 2 最多执行 C_m^2 次,当采用平衡树的数据结构存储测试需求集时,语句(1)、语句(2)的时间复杂度均为 $O(\log(m))$.因此,该算法的最坏时间复杂度为 $O(m^2 \cdot \log(m))$.一般情况下,TRR_Sub 可以有效减少待处理的测试需求数量,降低后续工作的计算开销.

算法 **1**. 基于包含关系的测试需求约简方法(TRR_Sub).

输入:原测试需求集 R.

输出:精简测试需求集 R'.

```
R':=R;                                  //步骤 1:初始化
Pair:={(ri,rj)|ri,rj∈R(i,j=1,2,…,m,i<j)};
while (Pair 不为空)                      //步骤 2:约简等价、包含关系的测试需求
    从 Pair 中取出一个元素(ri,rj),Pair:=Pair−{(ri,rj)};
    if (ri⊆rj) then
        R'=R'−{rj};                     //(1) 合并等价的测试需求或保留被包含的测试需求,保留 ri
    else if (rj⊆ri) then
            R'=R'−{ri};                 //(2) 保留被包含的测试需求,保留 rj
        end if
    end if
end while
```

不妨设 TRR_Sub 输出的精简测试需求集 $R'=\{r_1,r_2,\ldots,r_k\}(k\le m)$,TRR_Greedy 将利用部分重合关系,使用贪心算法进一步约简测试需求集.如算法 2 所示,TRR_Greedy 逐次生成一项新的测试需求 r,r 可以取代 R'中尽可能多的测试需求.该方法反复迭代,直到 R'为空为止.在 TRR_Greedy 中,语句(1)的时间开销为 $O(1)$(A 中的元素按基数的降序排列);语句(2)、语句(3)的时间开销均为 $O(k)$,故语句(1)~语句(3)一次执行的开销为 $O(2k+1)$.if 语句的 then 分支从 R'中删除已被约简的测试需求,最多执行 k 次,且语句(4)的开销为 $O(2^k)$,故 if 语句的总开销为 $O(k\cdot 2^k)$.由于 while 语句最多执行 2^k-1 次,因此其开销为 $O((2^k-1)(2k+1)+k\cdot 2^k)=O(3k\cdot 2^k)$.因算法中其他语句的开销均不超过 $O(2^k)$,故该算法的最坏时间复杂度为 $O(3k\cdot 2^k)$.在运行过程中,该算法在约简测试需求后更新集合 A,使得集合 A 的元素个数通常远远少于 2^k-1.因此,该算法的实际性能通常非常优于理论上的最坏情况.TRR_Greedy 能够获得规模较小的精简测试需求集,但其时间复杂度较高,通常应用于小规模的测试需求约简.

算法 **2**. 基于贪心算法的测试需求约简方法(TRR_Greedy).

输入:精简测试需求集 R'.

输出:精简测试需求集 R''.

```
R'':=∅;                                          //步骤 1:初始化
A:=R'的非空幂集;          //集合 A 的元素为 R'的非空子集(共有 2^k−1 个),元素按基数的降序排列
while (R'不为空)                                  //步骤 2:约简部分重合关系的测试需求
    从 A 中取出一个当前基数最大的元素 B,A:=A−{B};   //(1) 集合 B 为 A 的元素,即 R'的非空子集
    r:=r1∪r2∪…∪rk;                               //(2) 初始化 r 为 R'中所有测试需求的并集
    foreach ri∈B do                               //(3)
        r:=r∩ri;          //测试需求 r 为集合 B 中测试需求的交集,是约简得到的新的测试需求
    end for
    if (r≠∅) then
        R':=R'−B;                                 //从 R'中删除已被约简的测试需求
        R'':=R''+{r};                             //将新的测试需求 r 加入 R''
        A:=R'的非空幂集;                           //(4) 更新集合 A
    end if
end while
```

由于 TRR_Greedy 的时间复杂度较高,不适用于较大规模的测试需求约简,为此,我们提出基于线性搜索的测试需求约简方法 TRR_Linear.如算法 3 所示,该方法多次线性扫描测试需求集 R',寻找出某些(而非最多的)测

试需求,将这些测试需求的交集所对应的新测试需求 k 加入 R''并更新 R'.在最坏情况下,该算法需遍历 R'中所有 k 个元素以完成约简,而约简一项测试需求需耗费 $O(k)$,所以,其最坏时间复杂度为 $O(k^2)$.虽然 TRR_Linear 获得的精简测试需求集的规模较大,但该算法有效地降低了计算开销,是 TRR_Greedy 的一种简化.对于大规模的测试需求集,TRR_Linear 的效率更高,可行性更强.

算法 **3**. 基于线性搜索的测试需求约简方法(TRR_Linear).

输入:精简测试需求集 R'.

输出:精简测试需求集 R''.

R'':=∅; //步骤 1:初始化

while (R'不为空) //步骤 2:约简部分重合关系的测试需求

从 R'中取出一项测试需求 r,R':=R'−{r};

k:=r;

foreach $r_i \in R'$ ($r_i \neq r$) **do**

if ($k \cap r_i \neq \varnothing$) **then**

k:=$k \cap r_i$; //测试需求 k 为 R'中部分测试需求的交集,是约简得到的新的测试需求

R':=R'−{r_i}; //从 R'中删除已被约简的测试需求

end if

end for

R'':=R''+{k}; //将新的测试需求 k 加入 R''

end while

3.2 基于精简测试需求集的测试用例集优化

采用上述方法约简测试需求,不仅有效地减少了测试需求数量,而且极大地降低了测试需求间关系的复杂度.因此,基于精简测试需求集 R''可以有效地开展测试用例集生成和约简工作,从而实现测试用例集优化.

对于测试用例集生成,最简单的方法是针对 R''中的每项测试需求生成一条测试用例,构成优化的测试用例集 TS,$|TS|=|R''|$.然而,R''中的测试需求 r 可能对应原测试需求集中的多项测试需求,有时难以生成一条测试用例 t 以满足 r.不妨设 $r=r_{i_1} \cap r_{i_2} \cap ... \cap r_{i_s}$,这时,可以把 r 分解为两项或多项测试需求,分别为其生成测试用例.例如,把 r 分解为两项测试需求 r_1 和 r_2,分别生成测试用例 t_1 和 t_2,其中:$t_1 \in Test(r_1)=Test(r_{i_1} \cap r_{i_2} \cap ... \cap r_{i_k})$,$t_2 \in Test(r_2)=Test(r_{i_{k+1}} \cap r_{i_{k+2}} \cap ... \cap r_{i_s})$.在最坏情况下,$r$ 被分解为 $r_{i_1}, r_{i_2}, ..., r_{i_s}$,共需生成 s 条测试用例.

在测试用例集约简方面,可以基于 R''使用已有的测试用例集约简方法.此时,测试人员只需关注已有满足关系中与 R''相关的部分.然而,对于 R''中的测试需求 $r=r_{i_1} \cap r_{i_2} \cap ... \cap r_{i_s}$,若原测试用例集 T_{ori} 中不存在满足 r 的测试用例,同样需要把 r 分解为两项或多项测试需求,使原测试用例集中存在测试用例,可以满足分解后的测试需求.例如:把 r 分解为两项测试需求 r_1 和 r_2,存在 $t_1, t_2 \in T_{ori}$,其中:$t_1 \in Test(r_1)=Test(r_{i_1} \cap r_{i_2} \cap ... \cap r_{i_k})$,$t_2 \in Test(r_2)=Test(r_{i_{k+1}} \cap r_{i_{k+2}} \cap ... \cap r_{i_s})$.在最坏情况下,$r$ 被分解为 $r_{i_1}, r_{i_2}, ..., r_{i_s}$,此时,测试需求约简失效.

3.3 实例分析

本节通过一个实例演示了基于测试需求约简的测试用例集优化方法的计算过程,并将我们的方法与现有测试用例集约简方法(GE[2],H[1],GRE[2])进行了比较.

针对函数 *pushdown*()的分支覆盖标准所对应的测试需求 b_1~b_{19}(如图 2 所示)[2],本文的方法首先将约简原测试需求集 $R=\{b_1,b_2,\ldots,b_{19}\}$.通过程序分析可知[5]:$b_1 \equiv b_2 \equiv b_4$,$b_5 \equiv b_{10}$,$b_7 \equiv b_9$,$b_{11} \equiv b_{13} \equiv b_{14}$,$b_{15} \equiv b_{17} \equiv b_{18}$,$b_{16} \equiv b_{19}$,$b_3 \subseteq b_1$,$b_3 \subseteq b_2$,$b_5 \subseteq b_3$,$b_6 \subseteq b_3$,$b_7 \subseteq b_5$,$b_8 \subseteq b_5$,$b_{11} \subseteq b_6$,$b_{12} \subseteq b_6$,$b_{15} \subseteq b_{12}$,$b_{16} \subseteq b_{12}$.我们首先利用等价和包含关系,使用 TRR_Sub 获得精简测试需求集 $R'=\{b_7,b_8,b_{11},b_{15},b_{16}\}$;然后,利用部分重合关系继续约简测试需求.由于 R'的规模较小,故使用 TRR_Greedy 来获得精简测试需求集 R''.进一步分析表明:b_7,b_8,b_{16} 是 3 个互斥的循环出口(即 $b_7 \cap b_8 = b_7 \cap b_{16} = b_8 \cap b_{16} = \varnothing$),又有 $b_{11} \cap b_{15} \cap b_7 \neq \varnothing$,$b_{11} \cap b_{15} \cap b_8 \neq \varnothing$,$b_{11} \cap b_{15} \cap b_{16} \neq \varnothing$.不妨假定 TRR_Greedy 首先选择 $b_{11} \cap b_{15} \cap b_7$ 加入

R'',则最终获得的精简测试需求集 $R''=\{b_{11}\cap b_{15}\cap b_7,b_8,b_{16}\}$.通过测试需求约简,测试需求的数量从 19 个减少到了 3 个.此时,只需考虑满足这 3 项测试需求,就可以实现分支覆盖的测试目标,极大地减少了后续测试用例集生成和约简的计算开销.

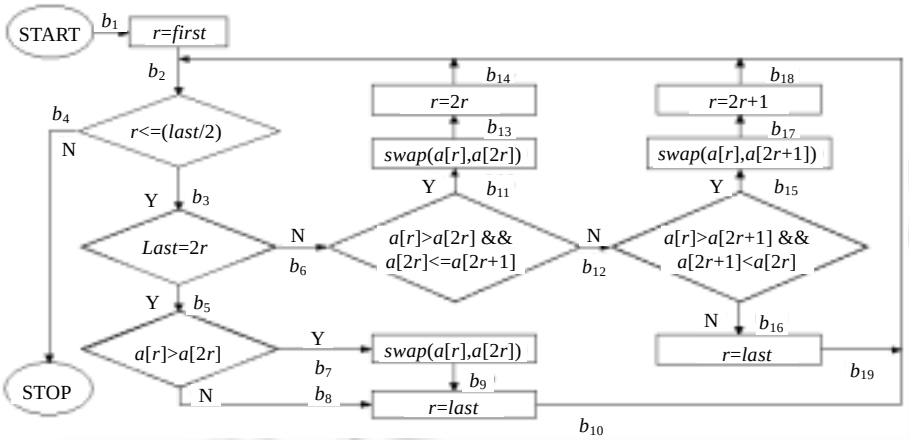


Fig.2 Control flow diagram of *pushdown*()

图 2 *pushdown*()的控制流图

在约简测试需求后,可以基于精简测试需求集进行相应的测试用例集优化工作.在生成测试用例时,测试人员只需针对 $R''=\{b_{11}\cap b_{15}\cap b_7,b_8,b_{16}\}$中的每项测试需求分别生成一条测试用例.表 2 记录了一个满足 R''中所有测试需求的可行的测试用例集 *TS*.同时,*TS* 是满足 *R* 的最小测试用例集,测试用例的个数为 3.

Table 2 Test suite *TS* for R''

表 2 满足 R''的测试用例集 *TS*

R''	*TS*	*a*[*i*]	*first*	*last*
$b_{11}\cap b_{15}\cap b_7$	t_1	7,1,15,2,1,8,9,11,10,5	1	10
b_8	t_2	1,2	1	2
b_{16}	t_3	1,2,3	1	3

除了有效指导测试用例生成以外,精简测试需求集还可以应用于测试用例集约简.基于表 3 所记录的测试需求与测试用例的满足关系[2],表 4 比较了 6 种测试用例集约简方法的约简结果,其中:GE,H 和 GRE 分别表示基于原测试需求集,使用相应方法直接约简测试用例集的实验结果[2];GE′,H′和 GRE′则分别表示基于精简测试需求集,使用相应方法约简测试用例集的实验结果.由于表 3 中不存在测试用例满足 $b_{11}\cap b_{15}\cap b_7$,因此,我们基于精简测试需求集 $R'=\{b_7,b_8,b_{11},b_{15},b_{16}\}$,分别采用 GE,H,GRE 约简测试用例集.

Table 3 Satisfiability relation of test cases t_1~t_{12} and testing requirements b_1~b_{19}

表 3 测试用例 t_1~t_{12} 与测试需求 b_1~b_{19} 的满足关系

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}
t_1	×	×	×	×	×		×		×	×									
t_2	×	×	×	×	×	×		×		×	×		×	×					
t_3	×	×	×	×		×					×	×	×	×	×		×	×	
t_4	×	×	×	×		×						×			×	×	×	×	×
t_5	×	×	×	×	×			×		×									
t_6	×	×	×	×		×						×				×			×
t_7	×	×	×	×	×		×		×	×									
t_8	×	×	×	×		×					×	×	×	×		×			×
t_9	×	×	×	×	×	×		×		×		×			×		×	×	
t_{10}	×	×	×	×	×	×	×		×	×	×		×	×					
t_{11}	×	×	×	×	×			×		×									
t_{12}	×	×	×	×	×	×	×		×	×		×			×		×	×	

如表 4 所示,基于原测试需求集 R,当 $T^{(1)}=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7\}$时,GRE 获得了最小测试用例集$\{t_2,t_4,t_1(t_7)\}$(其中:测试用例按选择的先后顺序排列,括号中的测试用例可以替换相应的测试用例来构成新的测试用例集);当 $T^{(2)}=\{t_1,t_2,t_3,t_4,t_8,t_9\}$时,H 获得了最小测试用例集$\{t_1,t_4(t_8),t_2(t_9)\}$;当 $T^{(3)}=\{t_1,t_3,t_4,t_5,t_6,t_8,t_{10},t_{11},t_{12}\}$时,GE 获得了最小测试用例集$\{t_{12},t_8,t_5(t_{11})\}$.因此,GE,H,GRE 中的任意一种方法都不比其他方法更优越,不总是获得最小测试用例集.然而,基于精简测试需求集 R',GE′,H′,GRE′均获得了相同的最小测试用例集(合并显示为表 4 的第 5 列),即获得了由测试需求间相互关系所确定的优化测试用例集,测试用例的个数为 3.而且,由于 R'的规模较小,从而有效地降低了测试用例集约简算法的计算量.

Table 4 Test suite reduction comparison of GE, H, GRE, GE′, H′ and GRE′

表 **4** GE,H,GRE,GE′,H′,GRE′的测试用例集约简结果比较

Initial test suite	GE	H	GRE	GE′/H′/GRE′
$T^{(1)}=\{t_1,t_2,t_3,t_4,t_5,t_6,t_7\}$	$\{t_3,t_1(t_7),t_4(t_6),t_2(t_5)\}$	$\{t_3,t_1(t_7),t_4(t_6),t_2(t_5)\}$	$\{t_2,t_4,t_1(t_7)\}$	$\{t_2,t_4,t_1(t_7)\}$
$T^{(2)}=\{t_1,t_2,t_3,t_4,t_8,t_9\}$	$\{t_1,t_3,t_2(t_9),t_4(t_8)\}$	$\{t_1,t_4(t_8),t_2(t_9)\}$	$\{t_1,t_3,t_2(t_9),t_4(t_8)\}$	$\{t_1,t_4(t_8),t_2(t_9)\}$
$T^{(3)}=\{t_1,t_3,t_4,t_5,t_6,t_8,t_{10},t_{11},t_{12}\}$	$\{t_{12},t_8,t_5(t_{11})\}$	$\{t_5(t_{11}),t_3,t_1(t_{10},t_{12}),t_4(t_6,t_8)\}$	$\{t_5(t_{11}),t_3,t_{10}(t_{12}),t_4(t_8)\}$	$\{t_8,t_{12},t_5(t_{11})\}$

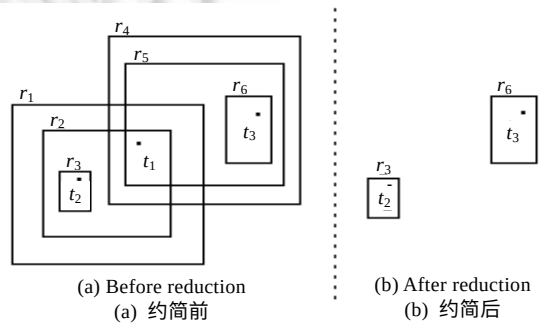
该实例分析表明,在测试需求数量较大、相互间关系复杂、测试资源又较为紧张的情况下,首先约简测试需求,然后基于精简测试需求集来实施测试用例集的生成和约简,不仅可以有效减小计算开销,而且有助于获得最小测试用例集.

4 关于测试用例集约简的仿真实验

为了进一步比较现有测试用例集约简方法(记为 TR)和基于精简测试需求集的测试用例集约简方法(记为 RR_TR),我们进行了一系列关于测试用例集约简的仿真实验.在仿真实验中,通过更改实验配置参数,验证本文提出方法的有效性,并研究该方法的适用范围.实验分析表明,测试需求约简是现有测试用例集约简方法的有力补充,作为现有约简方法的前期处理,有助于获得规模较小的测试用例集.

4.1 实验设计

在仿真实验中,我们建立如下实验模型:测试需求和测试用例均分布在一个二维的平面空间,该空间表示为一个矩形区域.在该矩形区域中,一项测试需求表示为一个矩形,一条测试用例表示为一个点.若一个点落在一个矩形中,则该点所代表的测试用例能够满足该矩形所代表的测试需求.测试需求 r_i,r_j 的相互关系可以通过相应矩形 A_i,A_j 的相对位置来描述:当 A_i,A_j 完全重合时,则 $r_i\equiv r_j$;当 A_i 包含在 A_j 中时,则 $r_i\subseteq r_j$;当 A_i,A_j 存在相交部分时,则 $r_i\oplus r_j$;当 A_i,A_j 不存在相交部分时,则 $r_i><r_j$.例如,在图 3(a)中,6 个矩形分别对应测试需求 $r_1,r_2,\ldots,r_6$,其中:$r_3\subseteq r_2;r_2\subseteq r_1;r_6\subseteq r_5;r_5\subseteq r_4;r_1\oplus r_4;r_1\oplus r_5;r_2\oplus r_4;r_2\oplus r_5;r_3><r_4;r_3><r_5;r_3><r_6;r_1><r_6;r_2><r_6$.3 个点分别对应测试用例 t_1,t_2,t_3,其中:$t_1\in Test(r_2\cap r_5);t_2\in Test(r_3);t_3\in Test(r_6)$.该模型采用矩形和点的表示方法,简化地描述了测试需求和测试用例之间的满足关系、测试需求间的相互关系.在此基础上,可以高效地进行关于测试用例集约简的仿真实验.



(a) Before reduction (a) 约简前 (b) After reduction (b) 约简后

Fig.3 Reduction of testing requirements

图 3 测试需求约简结果

该仿真实验包括下面 5 个步骤:(1) 随机生成一批矩形构成测试需求集 R;(2) 在每个矩形中生成一个点(测试用例),以确保每项测试需求至少能被一条测试用例满足;(3) 在测试用例空间中,随机生成一批测试用例,即随机地建立测试需求和测试用例的满足关系;(4) 利用本文提出的测试需求约简方法,获得精简测试需求集 R'';(5) 基于 R 和 R'',分别利用 G[1],H[1]和 GRE[2]来约简测试用例集,并输出约简的测试用例集.

该仿真实验的配置参数包括:初始测试需求个数(RN)、测试用例个数(TN)和嵌套阈值 $STRP$.其中:1-$STRP$ 为包含关系的测试需求的生成概率.即 $STRP$ 越小,在当前矩形中生成一个被包含的矩形的概率就越大.在实验过程中,随机生成一个实数 $a\in[0,1]$,若 $a\geq STRP$,则在当前矩形中生成一个被包含的矩形;否则,停止在当前矩形中继续生成被包含的矩形.在本次实验中,令 RN=20,TN=1000,$STRP$ 的取值范围为[0.025,0.900].对于每组参数配置(RN,TN,$STRP$)进行 20 次实验,实验结果的平均值记录在表 5 中.其中:G,H 和 GRE 分别表示基于原测试需求集 R,直接约简测试用例集的实验结果(即 TR);G′,H′和 GRE′分别表示基于精简测试需求集 R''约简测试用例集的实验结果(即 RR_TR).

Table 5 Test case number comparison between TR and RR-TR

表 5 TR 与 RR-TR 获得的测试用例个数比较

STRP	G	G′	H	H′	GRE	GRE′
0.025	19.2	3.5	19.2	3.5	19.2	3.5
0.050	18.4	6.9	18.4	6.9	18.4	6.9
0.075	16.8	8.7	16.8	8.7	16.8	8.7
0.100	15.4	10.6	15.4	10.6	15.4	10.6
0.125	15.1	11.0	15.1	11.0	15.1	11.0
0.150	14.7	11.2	14.7	11.2	14.7	11.2
0.175	14.7	12.3	14.7	12.3	14.7	12.3
0.200	14.5	12.6	14.5	12.6	14.5	12.6
0.250	13.0	12.1	12.2	12.1	12.2	12.1
0.300	11.9	11.8	11.7	11.7	11.7	11.7
0.400	11.1	10.5	10.6	10.4	10.6	10.4
0.500	8.5	8.1	8.0	8.0	8.0	8.0
0.600	7.9	7.5	7.3	7.3	7.3	7.3
0.700	6.2	6.0	5.9	5.9	5.9	5.9
0.800	5.1	5.0	4.8	4.8	4.8	4.8
0.900	4.8	4.8	4.5	4.5	4.5	4.5

通过对表 5 的分析可知:(1) 当 $STRP$ 的取值范围为[0.025,0.250]时,无论采用何种测试用例集约简方法,RR_TR 的约简结果均优于 TR.图 4 显示了这两种方法的约简结果的变化趋势,随着 $STRP$ 取值的逐渐增大,这两种方法获得的测试用例的个数差距将逐渐减小;(2) 当 $STRP$ 的取值范围为[0.250,0.900]时,TR 和 RR_TR 的约简结果基本相同.然而在这种情况下,方法 G 的约简结果总是最差,即约简后测试用例的个数最多.方法 G′的约简结果略优于方法 G,但通常仍差于其他 4 种方法.

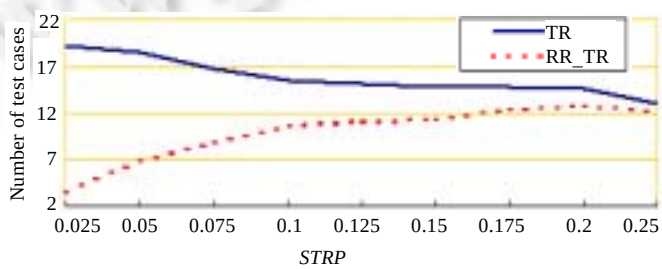


Fig.4 Test case number comparison between TR and RR-TR when $STRP\in[0.025,0.250]$

图 4 $STRP\in[0.025,0.250]$,TR 与 RR-TR 获得的测试用例个数比较

4.2 实验分析

以上实验结果表明:方法 G 总是生成规模较大的测试用例集;实验参数 *STRP* 将直接影响实验中测试需求间包含关系的复杂程度,从而影响输出的约简测试用例集的规模.下面分析产生上述实验结果的可能原因.

方法 G(贪心算法)逐次挑选出满足最多数量的测试需求的测试用例,当一批测试用例满足的测试需求数量相同时,G 将随机选择其中一条测试用例.该方法存在下列问题:首先,随机选择的不确定性无法保证获得全局最优解;其次,由于测试需求间存在着复杂的相互关系,贪心策略所选择的局部最优并不能保证全局最优.例如,基于图 3(a)的测试需求与测试用例的满足关系,G 将首先选择 t_1 来满足 4 项测试需求$\{r_1,r_2,r_4,r_5\}$,然后继续选择 t_2 和 t_3 来分别满足 r_3 和 r_6,共需要 3 条测试用例.事实上,仅需要 2 条测试用例$\{t_2,t_3\}$就可以满足所有 6 项测试需求.可见,G 很有可能产生规模较大的测试用例集.然而,若首先约简测试需求,获得精简测试需求集 $R'=\{r_3,r_6\}$,其中,$r_3><r_6$,如图 3(b)所示.此时,G′仅需选择 t_2 和 t_3 分别满足 r_3 和 r_6,同时,$\{t_2,t_3\}$满足了原测试需求集.因此,基于精简测试需求集,G′很有可能产生规模较小的测试用例集,其约简结果略优于方法 G.此外,在方法 H 和 GRE 的计算过程中,可能利用 G 作为一种测试用例选择策略.于是,H 和 GRE 可能存在和 G 同样的问题.然而,由于 H 和 GRE 不总是使用 G,当它们的计算过程中不使用 G 时,其约简结果通常将优于 G.因此,G 总是生成规模较大的测试用例集.

实验表明:当 *STRP* 较小时,方法 TR 将生成规模较大的测试用例集.一种可能的解释是:当 *STRP* 的取值范围为[0.025,0.250]时,大量的测试需求间存在着复杂的包含关系.TR 不考虑测试需求间的相互关系,基于原测试需求集直接约简测试用例集.此时,由于随机选择的不确定性,G,H 和 GRE 将生成规模较大的测试用例集.相对地,方法 RR_TR 先约简测试需求,有效地减少了测试需求的数量,并降低了测试需求间关系的复杂程度,将有助于获得规模较小的测试用例集.另一方面,当 *STRP* 取值大于 0.250 时,TR 和 RR_TR 的约简结果基本相同.可能的原因是:此时,测试需求间的关系较为简单,在测试需求约简阶段只能约简掉少量的测试需求,RR_TR 的优势不明显.尽管如此,测试需求约简仍是现有测试用例集约简方法的有力补充,有助于生成规模较小的测试用例集.

关于测试用例集约简的仿真实验表明:测试需求约简作为测试用例集优化方法的第一阶段工作,效果良好.当测试需求间关系复杂,存在较多包含关系时,我们的方法呈现出更为明显的优势.

5 结论和未来工作

在已有工作的基础上[1−9],本文从测试需求约简的角度考虑测试用例集优化问题,提出基于测试需求约简的测试用例集优化方法.该方法包括测试需求约简和测试用例集优化两个阶段.本文重点研究第一阶段的工作,在测试需求约简模型的基础上,提出一种测试需求约简方法来获得精简测试需求集.实验分析表明:测试需求约简可以有效地减少后续测试用例集生成和约简的计算开销,且有助于获得由测试需求所确定的最小测试用例集,是现有测试用例集优化方法的有力补充.

本文提出的模型与方法具有以下特点:

(1) 本文提出通用的测试需求约简模型和方法,充分利用测试需求间的关系,获取精简测试需求集作为测试用例集生成和约简的基础.本文提出的测试需求约简方法可以适用于不同的测试需求集规模和测试资源约束,具有良好的灵活性.当测试需求数量较大、相互间关系复杂、存在较多包含关系时,该方法效果明显.

(2) 已有的启发式测试用例集约简方法(如 G[1],H[1],GRE[2]等方法)仅考虑测试需求与测试用例间的满足关系,忽视了测试需求间的相互关系.我们的方法针对测试需求间关系获取精简测试需求集,减少了待处理测试需求的数量,降低了测试需求间关系的复杂度.基于精简测试需求集再进行测试用例集的生成和约简,可以有效减少计算开销,且有助于获得规模较小的测试用例集.

(3) 在结构化单元测试中,扩张集技术[5]针对结构化测试覆盖标准,利用程序分析技术来约简测试需求.本文给出的通用的测试需求约简模型,可以抽象地描述测试需求间的相互关系,所提出的测试需求约简方法不仅可以应用于结构化测试,还可以广泛应用于其他各种测试领域.

基于现有的工作,我们将对下列问题展开进一步研究:(1) 获取测试需求间关系是测试需求约简的基础.在

结构化单元测试中,可以使用扩张集技术获取测试需求间关系.在其他类型的测试中,如何获得完整、准确的测试需求间关系还有待进一步研究[10];(2) 当测试需求发生变更时,研究如何利用测试需求间关系、测试历史等信息,生成和约简测试用例,对修改后的软件进行有效的回归测试;(3) 针对具体的测试目标,研究特定的测试需求约简、测试用例集优化方法[11,12].

致谢 我们曾就测试需求约简的有关问题与澳大利亚 Swinburne 工业大学的 Chen 教授进行了讨论;此外,审稿人也对我们的研究提出了一些宝贵意见,在此一并表示感谢.

References:

[1] Harrold MJ, Gupta R, Soffa ML. A methodology for controlling the size of a test suite. ACM Trans. on Software Engineering and Methodology, 1993,2(3):270−285.

[2] Chen TY, Lau MF. A new heuristic for test suite reduction. Information and Software Technology, 1998,40(5/6):347−354.

[3] Chen TY, Lau MF. On the divide-and-conquer approach towards test suite reduction. Information Sciences, 2003,152(1):89−119.

[4] Lee JG, Chung CG. An optimal representative set selection method. Information and Software Technology, 2000,42(1):17−25.

[5] Marre M, Bertolino A. Using spanning sets for coverage testing. IEEE Trans. on Software Engineering, 2003,29(11):974−984.

[6] Nie CH, Xu BW. A minimal test suite generation method. Chinese Journal of Computers, 2003,26(12):1690−1696 (in Chinese with English abstract).

[7] Nie CH, Xu BW. An algorithm for automatically generating black-box test cases based interface parameters. Chinese Journal of Computers, 2004,27(3):382−388 (in Chinese with English abstract).

[8] Xu BW, Nie CH, Shi L, Chen HW. A software failure debugging method based on combinatorial design approach for testing. Chinese Journal of Computers, 2006,29(1):124−131 (in Chinese with English abstract).

[9] Zhang XF, Xu BW, Nie CH, Shi L, He YX. A requirements-driven test suite generation strategy. In: Tsai WT, ed. Proc. of the 9th IASTED Int'l Conf. on Software Engineering and Applications. Phoenix: ACTA Press, 2005. 224−227.

[10] Jin Z. Ontology-Based requirements elicitation. Chinese Journal of Computers, 2000,23(5):486−492 (in Chinese with English abstract).

[11] James AJ, Harrold MJ. Test-Suite reduction and prioritization for modified condition/decision coverage. IEEE Trans. on Software Engineering, 2003,3(29):195−209.

[12] Harman M, Hu L, Hierons R, Wegener J, Sthamer H, Baresel A, Roper M. Testability transformation. IEEE Trans. on Software Engineering, 2004,30(1):3−16.

附中文参考文献:

[6] 聂长海,徐宝文.一种最小测试用例集生成方法.计算机学报,2003,26(12):1690−1696.

[7] 聂长海,徐宝文.基于接口参数的黑箱测试用例自动生成算法.计算机学报,2004,27(3):382−388.

[8] 徐宝文,聂长海,史亮,陈火旺.一种基于组合测试的软件故障调试方法.计算机学报,2006,29(1):124−131.

[10] 金芝.基于本体的需求自动获取.计算机学报,2000,23(5):486−492.

章晓芳(1980－),女,福建泉州人,博士生,主要研究领域为软件分析与测试.

聂长海(1971－),男,博士,副教授,主要研究领域为软件工程,软件测试技术,模糊信息处理,神经网络.

徐宝文(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为程序设计语言及其理论与实现技术,软件开发方法与技术,软件分析测试与质量保证,软件逆向工程与软件再工程,知识与信息获取技术,基于 Web 系统及其分析测试技术.

史亮(1979－),男,博士生,主要研究领域为软件分析与测试.