文章编号 1004-924X(2016)05-1185-12

# 基于需求模型的航天软件测试用例生成方法

哈清华[1,2],刘大有[1,3*],沈湘衡[2],刘 逻[2]
(1. 吉林大学 计算机科学与技术学院,吉林 长春 130012;
2. 中国科学院 长春光学精密机械与物理研究所,吉林 长春 130033;
3. 吉林大学 符号计算与知识工程教育部重点实验室,吉林 长春 130012)

**摘要**:为了提高航天软件测试的效率和覆盖率,增加航天软件测试的有效性,提出了一种基于需求模型的软件测试用例设计方法。针对航天软件测试的特点,该方法采用从用例层深入至步骤层的测试优化策略。首先,基于元建模方法定义了一种测试需求模型;通过建立模型,将测试需求加以拆分,获取了测试需求之间的先后依赖关系以及测试需求与测试步骤之间的对应关系。然后,基于测试需求模型,构建了测试需求的路径图,进而通过对图的遍历获得了测试用例。最后,将该方法用于工程实践进行了实验验证。验证结果表明,该方法有效保证了测试活动的充分性和有效性,降低了测试用例约简的风险。与不约简的测试方法相比,该方法减少测试工作量达 18%,减少测试用例数量为 40%以上,软件测试的执行时间也减少了 40%以上,在满足需求覆盖率的同时,有效提高了测试效率。

**关 键 词**:软件测试;航天软件;测试用例生成;用例约简;测试需求;模型
**中图分类号**:TP301.6 **文献标识码**:A **doi**:10.3788/OPE.20162405.1185

# Test case generation of aerospace software based on modeling requirements

HA Qing-hua[1,2], LIU Da-you[1,3*], SHEN Xiang-heng[2], LIU Luo[2]

(1. *College of Computer Science and Technology, Jilin University, Changchun* 130012, *China*;
2. *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun* 130033, *China*;
3. *Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Jilin University, Changchun* 130012, *China*)
* *Corresponding author, E-mail: liudy@jlu.edu.cn*

**Abstract**: A model-driven software test case generation approach was researched to improve the efficiency and coverage rate of aerospace soft test and to increase the validity of test. According to the characteristics of aerospace software test, the strategy of step-layer test case optimization was put forward instead of case-layer optimization. Firstly, a model for testing requirements was presented based on meta-modeling method. Then, the test requirements were split by model building to obtain the relationship between the test requirements and the dependence of the test requirements on the test steps. With the constructed path graph of test requirements, test cases were gotten based on

**收稿日期**:2015-05-11;**修订日期**:2015-06-01.
**基金项目**:国家自然科学基金资助项目(No. 61133011);国家自然科学基金面上项目(No. 61373053, No. 61472161)

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

traversing for the graph. Finally, the approach was applied to the test of aerospace software test. The results of application show that this approach reduces the risk of test case optimization and is effective to ensuring the adequacy and validity of testing activities. The number of test cases are reduced more than 40%, and the whole workload of test is saved more than 18%. It meets the coverage rate of software test and improves the efficiency of test.

**Key words**: software test; aerospace software ; test case generation; optimization of test case; test requirement; model

## 1 引 言

软件质量会直接影响航天型号产品的质量，航天器发射、航空器试飞的失事原因60%源于软件[1]。保障软件质量和可靠性的主要手段是软件测试[2]。对于每个测试需求，都要设计一定量的测试用例以满足测试的充分性。由此所产生的测试用例数量往往比较大，对于测试用例的维护需要投入大量的人力和物力[3]。因此测试用例的约简已成为软件测试工作的重点[4]。测试用例约简方法主要可分为两类：一类是传统的测试用例约简，包括：贪心方法、启发式方法、整数规划算法、机器学习方法等[5-7]。这类方法的效果取决于最初选定的测试用例集，它不能根据测试目标完全实现测试用例集的整体优化，而且相关研究也显示上述方法没有一种方法总是能获得比其他方法更好的结果[8]；另一类是基于测试需求的约简[9-10]，这类方法在设计用例前先对需求进行约简，再进行测试用例设计，进而达到约简测试用例的目的。但约简过程十分繁琐，也不能保证每次都得到最简的测试用例集。随着研究的深入，人们开始尝试在用例的设计阶段引入用例约简思想，王俊杰[11]，郭曦[12]等人提出了基于参数和系统状态转移进行测试用例设计。该方法在面向对象类软件的测试中具有较好的效果。杨波[13]等人提了一种基于需求模型描述的测试用例设计方法，将软件建模与测试建模进行了统一。

已有的测试用例约简方法在航天软件测试过程中主要存在以下两个问题：第一，实现过程中引入较大的约简风险，约简后的测试充分性难以保证；第二，约简级别为用例层，对于不同的用例的执行时间估计不足，约简效果难以保证。本文针对航天软件测试的特点。提出了一种基于需求模型的航天软件测试用例生成方法，并对算法性能进行了实验验证。

## 2 航天软件测试的特点

在航天领域中，软件的设计需要考虑硬件、环境、人员误操作、安全性等多方因素，软件的控制约束多，因此规模大、逻辑关系复杂。这导致测试用例约束条件多，步骤繁琐，一个用例长度可包括几十个测试步骤，但各功能之间有较为严格的顺序与层次关系，因此测试的重复步骤多。另外，航天软件的主要用途在于进行硬件系统的控制与数据的运算，以嵌入式软件居多。软件的功能相对独立，各分系统间和人机间的交互很少，因此软件的需求相对单一和明确。这导致航天软件的测试用例虽然步骤多而复杂，但在一条用例中真正起到有效覆盖对应测试需求点的步骤仅有几步甚至是一步，进而测试效率低下。

本文针对航天软件测试的两大特点：(1)测试需求严格有序；(2)测试用例步骤多但需求覆盖效率低。提出了测试用例的约简由用例层深入至步骤层的策略。在需求分析阶段，先使用测试需求模型将测试需求进行整合与拆分，建立被测软件的需求路径图；再使用测试需求路径图生成树算法(GWBFS)对需求路径图进行剪枝，最后用生成树遍历算法(GWDFS)对生成树进行遍历，以实现测试用例的设计。

## 3 测试需求模型

### 3.1 相关的定义

定义1：使用一个测试步骤执行一次或对同一测试步骤执行多次，以实现完全测试覆盖的测试需求称为元测试需求，记为 $r$。

定义2：由多个元测试需求，通过交、并等运算形成的需求称为复合测试需求，记为 $R$。

定义3：设有复合测试需求 $R$ 以及元测试需

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

求 $r_1$、$r_2$，当 $r_1$ 与 $r_2$ 实现完全覆盖，即表示 $R$ 实现完全覆盖时，则称 $R$ 为 $r_1$ 与 $r_2$ 的并测试需求。表示为：$R=r_1 \cup r_2$。

定义 4：设有复合测试需求 $R$ 以及元测试需求 $r_1$、$r_2$，当 $r_1$ 与 $r_2$ 在执行同一测试步骤时实现完全覆盖，即表示 $R$ 实现完全覆盖，则称 $R$ 为 $r_1$ 与 $r_2$ 的交测试需求。表示为：$R=r_1 \cap r_2$。

根据以上定义可获得如下推论：

(1) $r_1 \cap r_2 = r_2 \cap r_1$；

(2) $r_1 \cup r_2 = r_2 \cup r_1$；

(3) $r_1 \cap (r_2 \cup r_3) = (r_1 \cap r_2) \cup (r_1 \cap r_3)$。

### 3.2　测试需求元模型

测试需求建模是一种较为有效的测试需求分析方法。本文针对航天软件测试的自身特点基于元建模技术[14]提出了一种测试需求的元模型。如图 1 所示。主要由 7 个测试模型组成。

(1) 测试场景模型：从用户的观点出发对系统建立模型是一个重要的获取测试需求的手段。一方面可以从用户的各类描述中获取此类测试需求，另一方面主要从被测系统的需求文档中获取测试需求。

(2) 需求目标模型：从整体系统测试的角度，依据各任务要求提取测试活动的各项目标。对于每个测试目标，需要描述其是执行某一特定的场景或路径，还是特定需求对系统的一个精确目标描述。一个测试目标可能需要从多个维度加以描述。例如可从多次执行的平均时间、多次执行的最大时间等不同的维度对处理时间进行描述。测试目标可能需在一定的系统约束（Test Constraint）下进行验证。

(3) 部件功能模型：该模型从系统设计的角度，以被测对象的构建组成情况为核心，对系统的功能需求进行描述。从系统的各类设计文档中提取此类测试需求。在部件功能模型中，各部件作为相互独立的个体进行详细分析和描述。模型中需明确各部件的接口对应关系，包括输入与输出接口，以便于在测试需求模型的描述中明确需求活动的先后顺序，以及测试用例的边界条件。

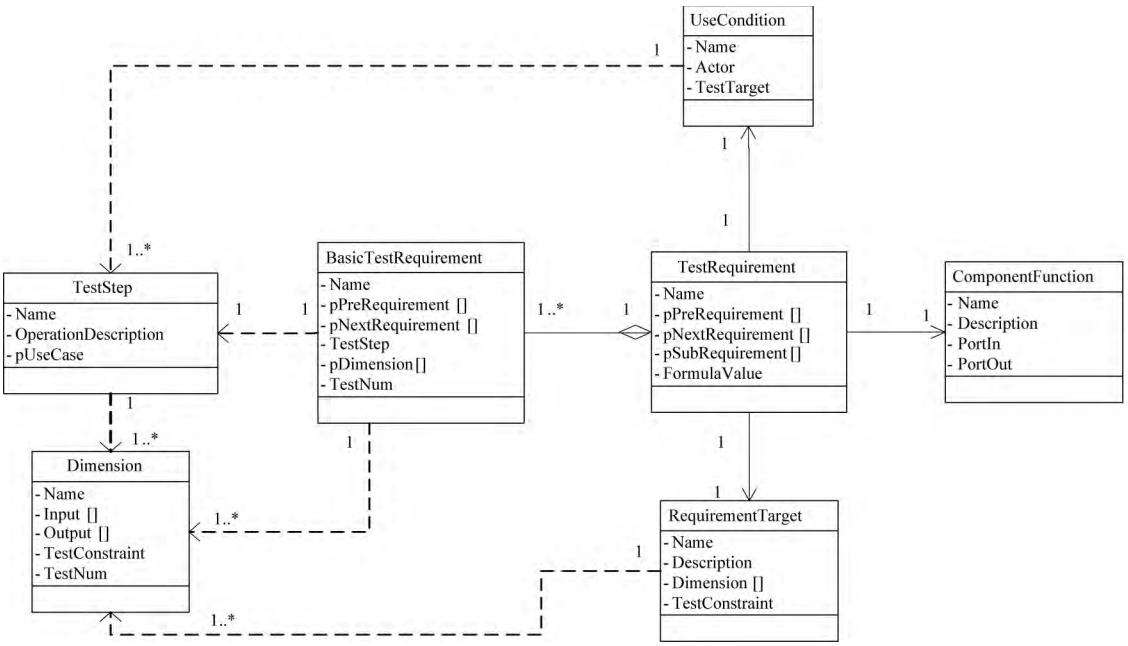



图 1　测试需求元模型

Fig. 1　A model of test requirements

(4) 测试需求模型是需求的核心，需要将已经实现的源需求逐条转换为测试需求模型，其过程为：

a) 建立源模型与测试需求模型的对应关系；

b) 梳理本测试需求的入口需求（pPreRequirement），即本测试需求需要通过哪些需求可直接进入，并建立链接关系。

c) 梳理本测试需求的出口需求

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

(pNextRequirement),即通过本需求可直接进入的需求,并建立链接关系。例如:存在需求链"初始化"→"自检"→"拍照"。则"初始化"是"自检"的入口需求,而"自检"是"拍照"的入口需求;相反"拍照"是"自检"的出口需求。

d)分析本测试需求,并将测试需求依据所需的测试步骤及可覆盖的测试点进行拆分,表示为子需求的运算集合(FormulaValue),并建立链接关系。

(5)元测试需求模型是测试内容的核心,其变量的定义与测试需求模型的定义相同,但其指针链接的需求仅为元测试需求模型。TestNum 表示覆盖本测试需求 TestStep 需要执行的最少次数,其等于本需求的 Dimension 的 TestNum 之和。需将每个测试需求模型转换为元测试需求模型。转换规则如下:

(a)当测试需求模型覆盖的测试需求 $R$ 为元测试需求时,即 $R=r$,直接进行等值转换;

(b)当测试需求模型覆盖的测试需求 $R$ 为复合测试需求,且有 $R=r_1\cup r_2$,则分别按照 $r_1$ 和 $r_2$ 建立独立的元测试需求模型,其中入口需求与出口需求需要进行对应拆分并重新赋值;

(c)当测试需求模型覆盖的测试需求 $R$ 为复合测试需求,且有 $R=r_1\cap r_2$,则将 $r_1$ 和 $r_2$ 进行合并,形成新的元测试需求 $r_3$,合并的准则为:$r_3$ 的入口需求和出口需求为 $r_1$ 和 $r_2$ 的交集;$r_3$ 的 TestStep 为 $r_1$ 和 $r_2$ 的 TestStep 变量的相加;$r_3$ 的 Dimension 从 $r_1$ 和 $r_2$ 的 Dimension 变量中按照共同作用原则(在 $r_1$ 和 $r_2$ 特定的 Dimension 下 $r_3$ 有效)进行选取;

(d)当测试需求模型覆盖的测试需求 $R$ 为复合测试需求,且有 $R=r_1\cap(r_2\cup r_3)$,则按照推论(3)将 $R$ 进行分配,变为 $R=(r_1\cap r_2)\cup(r_1\cap r_3)$,再按照以上规则进行对应的测试需求分解;

(e)当获取的元测试需求模型中,需要通过不同的 TestStep 变量来实现对 Dimension 的覆盖时,需将该元测试需求模型按照不同的 TestStep 进行分解,以实现元测试需求模型与 TestStep 的一一对应。

(6)测试维度模型(Dimension)用以描述对应的元测试模型的不同维度的测试点情况。Input 表示对该维度的测试对应的测试步骤需要输入的信息,包括系统的输入变量,外部环境的数值变化等;Output 表示对该维度的测试需要覆盖的输出信息,包括系统的输出变量,系统的状态变化等。

(7)测试步骤模型(TestStep)用以描述测试对应的测试需求时所需要的测试活动内容。OperationDescription 变量用以描述本测试步骤所需要的各项活动内容。

## 4 基于测试需求模型的测试用例生成

定义5:顶点加权有向图 $G_v$ 是一个二元组 $\langle V,E\rangle$,其中:$V=\{v_s,v_1,v_2,\cdots,v_n,v_e\}$,是非空集合,称 $V$ 为加权顶点集,$v_s$ 称为图 $G_v$ 的初始节点,$v_e$ 称为图 $G_v$ 的结束节点;$E=\{e_1,e_2,\cdots,e_m\}$,是 $V\times V$ 的子集,称为边集;对于任意顶点 $v_i$,其权值为 $w_i$,则可用 $v_i:w_i$ 表示。

定义6:在图 $G_v$ 中从初始节点 $v_s$ 出发沿边集 $E$ 拓展的的一条通路 $p$ 称为图 $G_v$ 的一条路径;若 $p$ 的最后节点为图 $G_v$ 的结束节点 $v_e$,则称 $p$ 为完全路径。

定义7:有路径 $p$,$v$ 为其中的一个节点,则 $v$ 的层数 $l$ 为 $v$ 由 $v_s$ 开始计算的遍历序号。

使用项目中建立的元测试需求模型,依据各需求的入口与出口需求链接关系,可以实现对本项目的测试需求路径图 $G_v$ 的构建。

测试用例的自动生成问题变成对图 $G_v$ 的遍历问题。其描述为:

找到路径集合 $P=\{p_1,p_2,\cdots,p_x\}$ 使得对图 $G_v$ 中各节点进行完全遍历,即任意节点 $v_i$ 的遍历次数 $s_i$ 大于等于该节点的权值 $w_i$,同时各节点的遍历次数之和 $S=s_1+s_1+\cdots+s_n$ 最小。

本文基于贪心策略提出了一种测试需求路径图遍历算法。先使用 GWBFS 算法对测试需求路径图剪枝,去除无效的遍历路径以获取生成树,再使用 GWDFS 算法对生成树进行深度遍历以及动态维护,最终获取测试路径集合 $P$。

### 4.1 测试路径获取

推论4:设有图 $G_v$ 及其节点集 $V=\{v_s,v_1,v_2,\cdots,v_n,v_e\}$,任意节点 $v_i$ 的层数为 $l_i$,权值为 $w_i$,则有:

$$\text{节点遍历次数总和 } S\leqslant\sum_{i=1}^{n}(l_i\times w_i)\text{。}$$

根据推论4,获得贪心策略1):使各节点 $v$ 的层数 $l$ 最小。

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

**推论 5**：设有图 $G_v$ 及其节点集 $V=\{v_s, v_1, v_2, \cdots, v_n, v_e\}$，$s_i$ 为节点 $v_i$ 的遍历次数，则有：

$s_i \geqslant \max\{w_i, w_{next}\}$，其中 $w_{next}$ 表示 $G_v$ 的生成树中节点 $v_i$ 的子孙节点的最大权值。

根据推论 5，获得贪心策略 2)：使用权值较大的节点构造子生成树。

根据以上两条策略，设计测试需求路径图生成树算法(GWBFS)，具体策略为：通过宽度优先搜索算法获取各节点的层数，再对图进行剪枝，删除低层向高层以及同层之间的节点遍历的路径，形成宽度优先生成树。在剪枝过程中，对于具有共同父节点的待处理节点，优先选择权值大的节点进行树扩展。算法具体描述如下。

**算法 1**：GWBFS

**输入**：测试需求路径图 $G_v$ 的初始节点 $v_s$

**输出**：测试需求路径图 $G_v$ 的生成树 $T_v$

**初始化**：建立先进先出的 FIFO 队列 Buff1$=\{v_s\}$，设 $v_s$ 的层数为 0，先进先出的 FIFO 队列 Buff2 为空。

**步骤**：

1)如果 Buff1 为空，跳转至步骤 6；

2)Buff1 不为空，从 Buff1 中取出一个节点 $v$，将 $v$ 的出口节点按照 TestNum 值从大到小排列并依序放入 Buff2，且将出口节点层数为 $v$ 的节点层数加 1；

3)如果 Buff2 为空，判定 $v$ 是否还存在出口节点，当 $v$ 已不存在出口节点时，建立节点 $v$ 指向结束节点 $v_e$ 的链接，否则不修改，跳转至步骤 1；

4)若 Buff2 不为空，从 Buff2 中取出一个节点 $v_{next}$，对 $v_{next}$ 的状态进行判定：

a)当 $v_{next}$ 的标志 flag 为 false 时，表示该节点未被遍历，置 $v_{next}$ 的标志 flag 为 true，将 $v_{next}$ 放入 Buff1；

b)当 $v_{next}$ 的标志 flag 为 true 时，表示该节点已被遍历，若 $v_{next}$ 的层数大于等于 $v$ 节点的层数，则删除 $v_{next}$ 与父节点 $v$ 的链接；

5)继续判定其他出口节点，跳转至步骤 3；

6)输出 $T_v$，算法运行结束。

根据测试用例的设计要求，本文设定如下两个贪心策略：

**贪心策略 3)**：获取的路径 $p$ 必须为完全路径；

**贪心策略 4)**：及时删除遍历次数 $s$ 大于权值 $w$ 的节点。

根据以上两条策略以及策略 2，设计生成树遍历算法(GWDFS)。具体策略为：通过深度优先搜索算法获取路径 $p$，并调整 $p$ 上各节点的遍历次数 $s$，对遍历次数大于等于权值的点进行删除判定，当其所有的子节点均可通过其他父节点遍历或者子节点为结束节点时，将其删除，并记录该路径 $p$。再对调整后的树循环进行深度遍历，直到删除所有节点。算法具体描述如下。

**算法 2**：GWDFS

**输入**：生成树 $T_v$ 的初始节点 $v_s$

**输出**：路径集合 $P$

**初始化**：路径集合 $P$ 为空集，启动节点 $v=v_s$，缓存路径集合 Buff$=\{v_s\}$。

**步骤**：

1)以 $v$ 为根节点，按照权值最大原则选取子节点，对扩展树进行深度遍历，将遍历的节点依次放入 Buff，直至到达节点 $v_e$；

2)生成路径 $p$，Buff 中节点按照遍历的先后顺序依次放入 $p$ 的节点序列中，且 $p.\mathrm{num}=0$；

3)从 Buff 中取各节点的权值(0 除外)中最小的权值 $w_{min}$，调整 $p$ 的权值 $p.\mathrm{num}=p.\mathrm{num}+w_{min}$ 并将 Buff 中各节点的权值重新赋值 $w_{new}=w_{old}-w_{min}$，当 $w_{new}<0$ 时，设 $w_{new}=0$；

4)从后向前遍历 Buff 中的节点 $v$，当 $w_{new}=0$ 时，调整 $v$ 在树 $T_v$ 的链接情况：

a)当 $v$ 的子节点只有 $v_e$ 时，从 $T_v$ 中删除 $v$，修改 $v$ 的父节点的链接，将没有其他子节点的父节点指向 $v_e$；

b)当 $v$ 的子节点还有除 $v_e$ 外的其他节点，且各子节点均有其他父节点可遍历时，从 $T_v$ 中删除 $v$，修改 $v$ 的父节点的链接，将没有其他子节点的父节点指向 $v_e$；

c)当 $v$ 的子节点还有除 $v_e$ 外的其他节点，且存在子节点只有父节点 $v$ 可遍历时，不删除节点 $v$，在 $v$ 的子节点中有其他父节点可遍历的，删除该子节点与 $v$ 的链接；

5)如果 $T_v$ 中没有节点或链接被删除，或者删除的链接不在 Buff 的链接路径中，则跳转至步骤

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

3，重新计算 $p$ 的权值，否则跳转至步骤6；

6）将生成的 $p$ 放入集合 $P$，再从前向后遍历 Buff，找到序号最小的在 $T_v$ 中被删除或者后续链接中断的节点，并将该节点以及后面的节点从 Buff 中删除，获得新的 Buff，以及 Buff 的结尾节点 $v_{be}$；

7）如果 $v_{be}=v_s$，则跳转至步骤9；

8）设 $v=v_{be}$，并跳转至步骤1；

9）输出路径集合 $P$，算法结束。

### 4.2 路径与用例转化

通过测试需求模型可以将获取的路径集 $P$ 转化为用例集。具体步骤为：

1）选取 $P$ 中一个路径 $p$，将 $p$ 中的节点按照遍历的顺序进行排列；

2）找出 $p$ 中节点对应的测试步骤，并按照节点的顺序进行排列，形成测试用例 $c_p$ 的测试步骤；

3）将节点对应的 Dimension 中的测试约束 TestConstraint 依据类型分别放入测试用例 $c_p$ 的测试约束和测试初始化内容中；

4）将节点对应的 Dimension 中的 Input 变量放入对应的步骤中，作为该步骤的输入；

5）将节点对应的 Dimension 中的 Output 变量放入对应的步骤的期望结果中，作为该步骤的输出；

6）$p$ 的 num 变量确定了对应的测试用例 $c_p$ 需要执行的次数，多次的执行过程中只有 Input 和 Output 值进行对应的改变，其他内容不变。

## 5 实验分析

本次试验的对象为某航天相机项目软件，该系统软件由3个配置项软件组成，均采用嵌入式C语言编程。用来控制航天相机的对地成像、卫星平台通信以及相机硬件信息采集。软件情况见表1。

表1 测试对象

Tab.1 Test object

| 软件名称 | 代码行数 |
|---|---|
| 成像控制软件 | 2 157 |
| 温控与调焦控制软件 | 6 518 |
| 主控软件 | 12 199 |

本次实验的测试人员均为同一软件测评中心的测评人员。将测试人员分为4组，每组测试人员使用同一种方法对3个软件进行配置项测试。A组直接进行人工测试用例设计，不进行测试约简；B组使用G算法进行测试用例设计；C组使用参考文献[10]提出的需求约简方法进行测试用例设计；D组按照本文提出的需求模型驱动方法进行测试用例设计。

### 5.1 实验过程

本文以主控软件的性能测试需求“10 ms 内输出像移计算结果”为例，介绍D组的实验过程。

（1）分析测试需求。性能测试需要使被测对象处于最大负载下以进行功能处理。经分析，主控软件在进行像移计算的同时，最大负载情况为在1次像移计算周期中同时进行1次“指令解析”以及1次“温度采集”处理。据此将需求描述修改为“10 ms 内执行1次像移计算、1次指令解析和1次温度采集”。同时可能触发本需求的入口需求为“拍照指令处理”和“接收卫星平台数据”，并选择“3次测试结果的最大值”作为评价标准。因此依据表2中已建立的元测试需求节点，以及新构建的元测试需求 $r_{16}$ 建立本需求的目标需求模型和测试需求模型，如图2所示。

表2 已分析的元测试需求

Tab.2 Analyzed basic test requirements

| 元测试需求 | 内容 |
|---|---|
| $r_1$ | 像移计算 |
| $r_2$ | 卫星平台指令解析 |
| $r_3$ | 温度采集 |
| $r_4$ | 向卫星平台返回相机状态 |
| $r_5$ | 拍照指令处理 |
| $r_6$ | 向成像控制软件发送数据 |
| $r_7$ | 接收卫星平台数据 |
| $r_8$ | 软件自检 |
| $r_9$ | 调焦指令处理 |
| $r_{10}$ | 停止拍照指令处理 |
| $r_{11}$ | 停止调焦指令处理 |
| $r_{12}$ | 自校图像输出指令 |
| $r_{13}$ | 系统初始化 |
| $r_{14}$ | 遥测数据返回 |
| $r_{15}$ | 向温控与调焦软件发送数据 |

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

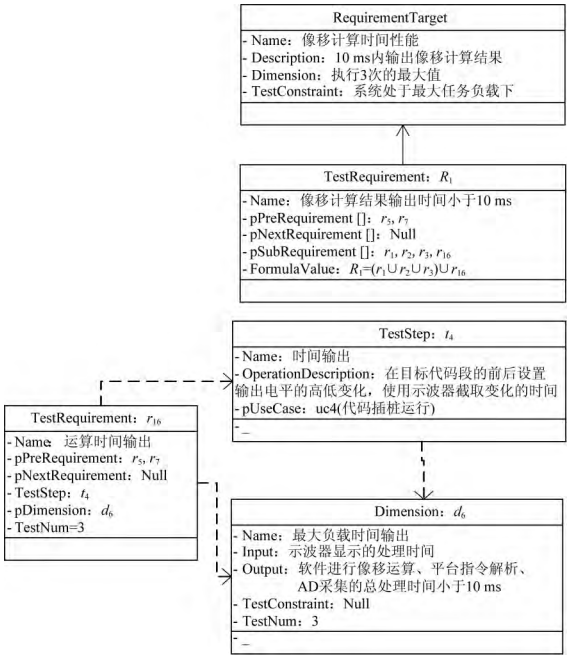



**图 2　测试目标与测试需求模型**

Fig. 2　Requirement target and test requirement

(2)将测试需求模型分解，构建新的元测试需求。依据元测试需求的转换规则将测试需求 $R_1$ 进行拆分，可获得 3 个元测试需求：$r_{17}=r_1\cap r_{16}$、$r_{18}=r_2\cap r_{16}$、$r_{19}=r_3\cap r_{16}$，同时删除无效的测试需求 $r_{16}$，生成的新需求结果见图 5。将性能测试分割为 3 个独立的子测试项，即单独对像移计算的时间、指令解析的时间和温度采集的时间进行测试，并要求 3 个时间的和小于 10 ms。

(3)基于已有的 15 个元测试需求和新构建的 3 个测试需求，依据各元测试需求的入口与出口需求链接关系，构造性能测试子路径图。如图 3 所示。

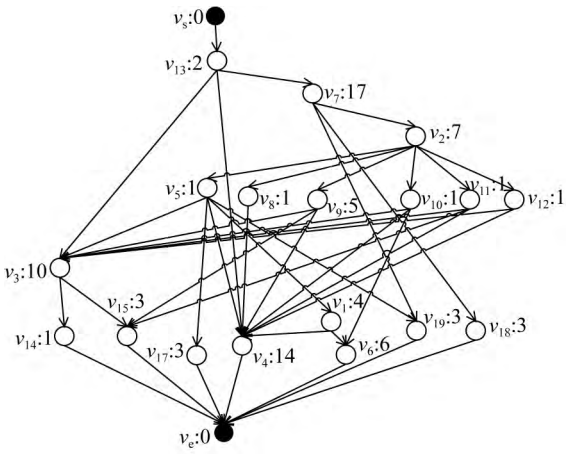



**图 3　测试需求路径子图**

Fig. 3　Subgraph of test requirements

(4)使用测试需求路径图生成树算法(GWBFS)获取子图的生成树，如图 4 所示。

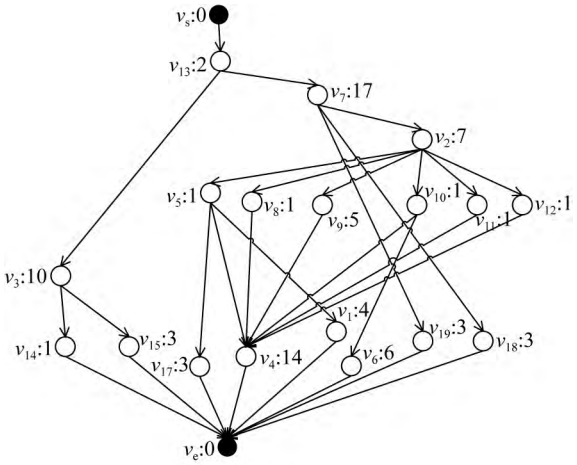



**图 4　测试需求路径子图的生成树**

Fig. 4　Spanning tree of test requiremens

(5)使用生成树遍历算法(GWDFS)获取路径集合 $P$，如表 3 所示。

(6)将路径转化为测试用例，获取用例 13 条。

**表 3　生成的路径**

Tab. 3　Generated pathes

| 路径 | 权值 | 节点 |
|---|---|---|
| $p_1$ | 5 | $r_{13}\to r_7\to r_2\to r_9\to r_4$ |
| $p_2$ | 9 | $r_{13}\to r_7\to r_2\to r_5\to r_4$ |
| $p_3$ | 4 | $r_{13}\to r_7\to r_2\to r_5\to r_1$ |
| $p_4$ | 3 | $r_{13}\to r_7\to r_2\to r_5\to r_{17}$ |
| $p_5$ | 1 | $r_{13}\to r_7\to r_2\to r_8$ |
| $p_6$ | 6 | $r_{13}\to r_7\to r_2\to r_{10}\to r_6$ |
| $p_7$ | 1 | $r_{13}\to r_7\to r_2\to r_{11}$ |
| $p_8$ | 1 | $r_{13}\to r_7\to r_2\to r_{12}$ |
| $p_9$ | 3 | $r_{13}\to r_7\to r_{19}$ |
| $p_{10}$ | 3 | $r_{13}\to r_7\to r_{18}$ |
| $p_{11}$ | 3 | $r_{13}\to r_3\to r_{15}$ |
| $p_{12}$ | 1 | $r_{13}\to r_3\to r_{14}$ |
| $p_{13}$ | 6 | $r_{13}\to r_3$ |

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

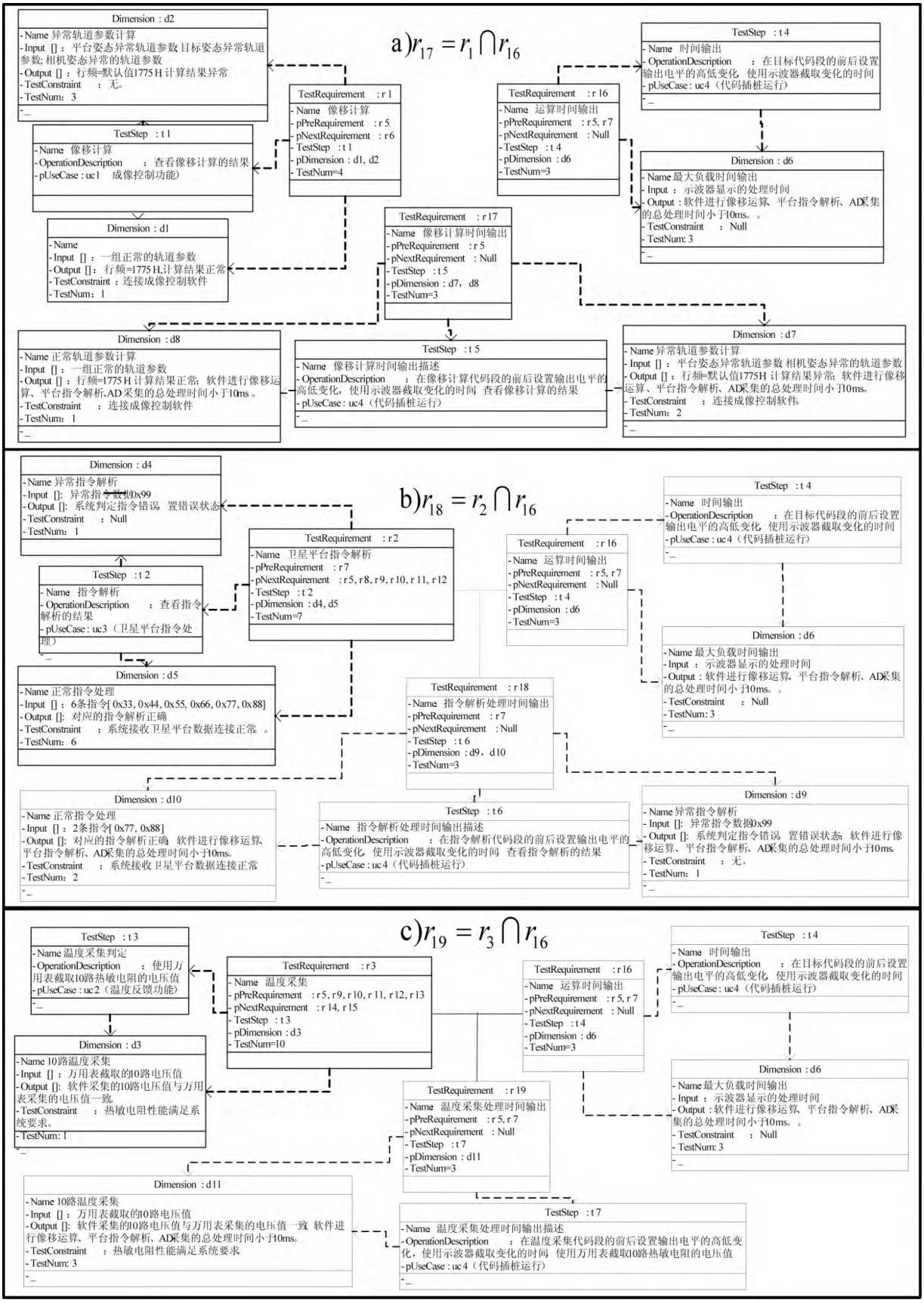



**图 5 元测试需求结构**

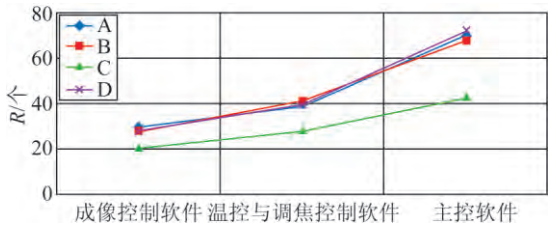
Fig. 5 Construction of basic test requirement

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

### 5.2　实验结果

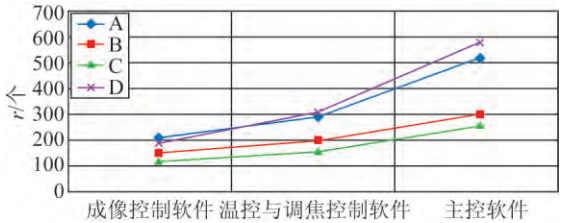
为了比较不同测试用例集约简方法的性能，分析各种约简方法的适用范围，对于测试用例集约简算法，本文采用以下 3 方面属性来评价其约简效果：

(1)**充分性**(Adequacy)：约简测试用例集应与原始用例集具有相同的测试充分性准则，包括对测试需求的充分性和代码覆盖的充分性，本文选取的衡量指标有：测试需求项($R$)、用例中的测试功能点($r$)、代码的语句覆盖率(LC)和分支覆盖率(BC)，数据内容见表 4。将数据进行比较，从图 6 可知，随着软件规模的增加，各种测试用例约简方法的测试覆盖性均有不同程度的下降。本文提出的需求模型驱动的用例设计方法对测试需求进行了充分的分析，同时与其他方法相比优化的重点由用例级转为了步骤级，测试点损失少，因此覆盖性与不进行约简的方法相比基本一致，而且在用例功能点的覆盖性上还有所提高。



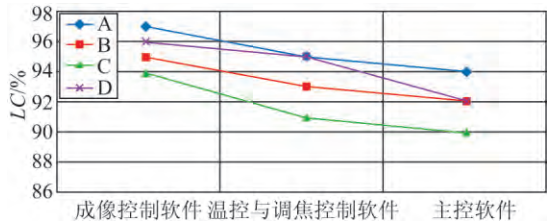

(a)**测试需求**

(a)Test requirement



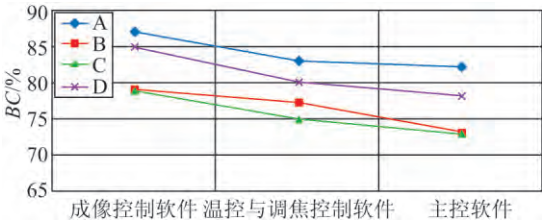

(b)**用例功能点**

(b)Function points of example




(c)**语句覆盖率**

(c)Sentence covering rate




(d)**分支覆盖率**

(d)Branch covering rate

**图 6　测试充分性比较**

Fig. 6　Comparison of testing adequacy

(2)**有效性**(Validity)：约简的测试用例集，应具有与原用例集相同的有效性，即具有一定的缺陷检出率。如果约简的用例集检出率过低可以认为测试无效。本次测试的有效性情况见图 7。从图中可以看出本文提出的测试用例约简与设计方法在有效性上较不约简的方法有所提高，这主要因为本方法在需求分析阶段对每个测试需求都进行了充分且有效的分析与分解，且对各种数据情况做了充分的研究。另外，与其他约简方法相比，本方法对需求与用例进行了小尺度约简，因此测试点遗漏少，有效性更高。

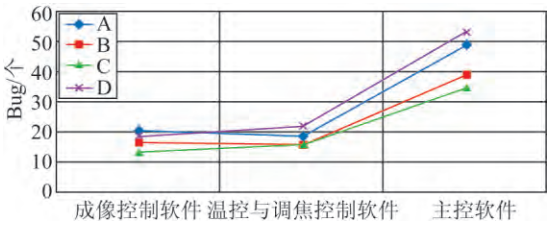


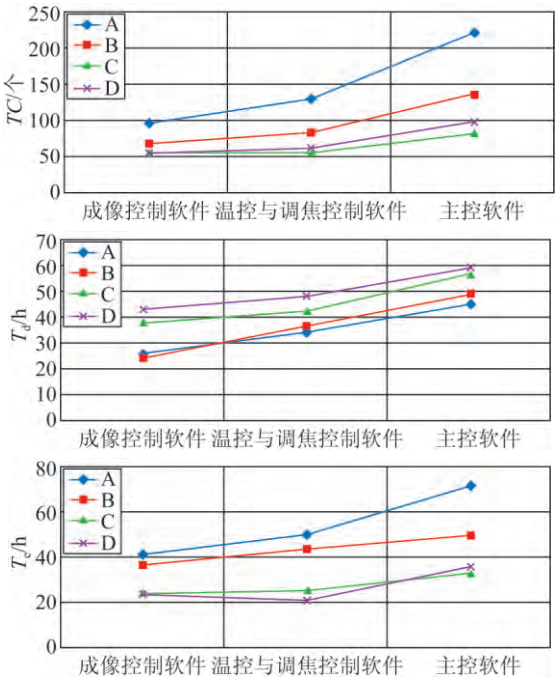

**图 7　测试有效性比较**

Fig. 7　Comparison of testing validity

(3)**效益**(Cost-effectiveness)：测试用例集的约简应是有效益的，即约简后的测试费用应小于约简前的测试费用。本文选取了测试用例数量(TC)、测试用例设计时间($T_d$)、测试用例执行时间($T_e$)、测试总时间($T$)作为测试效益的衡量指标。图 8 为各种方法的对比结果，从图中可知本方法构造的测试用例达到了约简的目的，与已有方法相比，测试用例设计所占时间更长，但设计用例数量更少，测试执行时间更短，测试总时间与其他约简方法相当。与不约简方法相比，测试用例

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

的设计时间增加了 30%以上，但测试用例数量可减少 45%以上，测试执行时间节约 40%以上，整体测试时间有所减少，同时随着测试对象规模的扩大，整体测试时间的减少程度可达 18%以上。



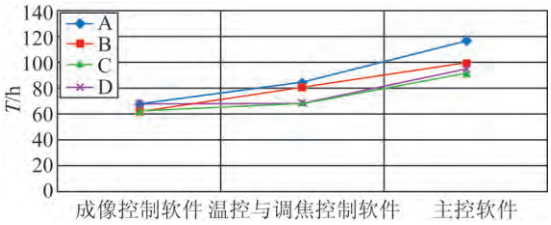




图 8 测试效益比较

Fig. 8 Comparison of testing cost-effectiveness

本次测试的 4 组结果的具体数值如表 4 所示。通过以上分析可知，与其他约简方法相比，本文测试用例的约简效益与其他方法相当，但测试需求模型驱动的测试用例设计方法更加充分、有效，具有可操作性，约简风险较小。对于航天软件测试来说属于有效的测试用例约简方法。

与不约简的测试用例设计方法相比，在保证了测试覆盖性以及测试缺陷检出率的前提下，测试的设计时间增加了 30%，但测试的用例数量减少了 40%，测试的执行时间减少了 40%，而且随着软件规模的逐渐增大，总的测试时间节约效果也逐渐增强。与不约简的方法相比整体工作量可减少 18%以上。

表 4 测试结果统计

Tab. 4 Statistical results of test

| | | 充分性 | | | | 效益 | | | | 有效性 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $R$ | $r$ | $LC$ | $BC$ | $TC$ | $T_d$ | $T_e$ | $T$ | Bug |
| 成像控制软件 | A | 31 | 212 | 97% | 87% | 97 | 25 | 42 | 67 | 21 |
| | B | 29 | 153 | 95% | 79% | 69 | 24 | 37 | 61 | 17 |
| | C | 21 | 127 | 94% | 79% | 55 | 38 | 25 | 63 | 14 |
| | D | 28 | 198 | 96% | 85% | 53 | 43 | 24 | 67 | 19 |
| 温控与调焦控制软件 | A | 41 | 296 | 95% | 83% | 128 | 34 | 51 | 85 | 19 |
| | B | 42 | 201 | 93% | 77% | 84 | 37 | 44 | 81 | 16 |
| | C | 29 | 162 | 91% | 75% | 57 | 42 | 26 | 68 | 16 |
| | D | 39 | 307 | 95% | 80% | 61 | 48 | 21 | 69 | 22 |
| 主控软件 | A | 71 | 522 | 94% | 82% | 221 | 45 | 72 | 117 | 49 |
| | B | 68 | 306 | 92% | 73% | 135 | 49 | 50 | 99 | 39 |
| | C | 43 | 268 | 90% | 73% | 83 | 57 | 34 | 91 | 35 |
| | D | 73 | 582 | 92% | 78% | 97 | 59 | 36 | 95 | 53 |

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

## 6 结　论

本文基于航天软件的特点，提出了基于需求模型的测试用例设计方法。该方法分为需求模型构建和测试用例设计两部分。通过建立需求模型，梳理了测试的依据，构建了测试用例设计的基础框架，进而形成了测试需求路径图；通过使用 GWBFS 方法将图进行剪枝，获得了测试需求途径图的生成树，进而使用 GWDFS 方法获得了遍历树的路径集合；在此基础上构造了测试用例。

通过实验可知，该方法与以往用例约简方法相比，对测试活动的充分性和有效性有更高的保证，降低了测试用例的约简风险；与此同时与不约简的测试方法相比测试工作量可减少 18%，特别是该方法可有效减少的测试用例数量达 40% 以上，软件的测试执行时间也可减少了 40% 以上。该方法达到了预期的效果。

### 参考文献：


[1] 张义德，王国庆，杨平，等. 基于需求的航空嵌入式软件测试技术研究[J]. 计算机工程与设计，2002，23(10)：4-7.
ZHANG Y D, WANG G Q, YANG P, *et al*.. A research on aeronautic embedded software testing technique based on requirement [J]. *Computer Engineering and Design*, 2002, 23(10): 4-7. (in Chinese)

[2] 刘逻，郭立红，方艳超，等. 系统级软件可靠性屋在 RTI 管理模块上的应用[J]. 光学 精密工程，2014，22(3)：787-796.
LIU L, GUO L H, FANG Y CH, *et al*.. The application of system-level house of software reliability on RTI management module[J]. *Opt. Precision Eng.*, 2014, 22(3): 787-796. (in Chinese)

[3] PETRENKO A, SIMAO A. Model-based testing of software and systems: recent advances and challenges[J]. *International Journal on Software Tools for Technology Transfer*, 2012, 14(4): 383-386.

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

[5] CHAE H S, WOO G, KIM T Y, *et al*.. An automated approach to reducing test suites for testing retargeted C compilers for embedded systems[J]. *The Journal of Systems and Software*, 2011, 84(12): 2053-2064.

[6] WANG SH, SHAUKAT A, ARNAUD G, *et al*.. Cost-effective test suite minimization in product lines using search techniques[J]. *The Journal of Systems and Software*, 2015, 103: 370-391.

[7] BESTOUN S A, TAIB SH A, MOAYAD Y P, *et al*.. Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm[J]. *Information and Software Technology*, 2015, 66: 13-29.

[8] ZHONG H, ZHANG L, ME H. An experimental comparison of four test suite reduction techniques [C]. *Proceeding of the 28th International Conference on Software Engineering*, 2006: 636-640.

[9] 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.
ZHANG X F, XU B W, NIE CH H, *et al*.. An approach for optimizing test suite based on testing requirement reduction[J]. *Journal of Software*, 2007, 18(4): 821-831. (in Chinese)

[10] 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，2006，26(12)：1690-1695.
NIE CH H, XU B W. A minimal test suite generation method [J]. *Chinese Journal of Computers*, 2006, 26(12): 1690-1695. (in Chinese)

[11] 王俊杰，沈湘衡，张波，等. 环境参数与状态参数融合的测试用例集约简方法[J]. 光学 精密工程，2009，17(7)：1678-1685.
WANG J J, SHEN X H, ZHANG B, *et al*.. Optimal test suite generation methods based on fusion of environment and state parameters [J]. *Opt. Precision Eng.*, 2009, 17(7): 1678-1685. (in Chinese)

[12] 郭曦，张焕国. 基于谓词抽象的测试用例约简生成方法[J]. 通信学报，2012，33(3)：35-51.
GUO X, ZHANG H G. Approach for reduced test suite generation based on predicate abstraction[J]. *Journal on Communications*, 2012, 33(3): 35-51. (in Chinese)

[13] 杨波，吴际，徐珞，等. 一种软件测试需求建模及测试用例生成方法[J]. 软件学报，2014，37(3)：522-538.
YANG B, WU J, XU L, *et al*.. An approach of


?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

modeling software testing requirements and generating testCase [J]. *Journal of Software*, 2014, 37(3):522-538. (in Chinese)

[14] BUDINSKY, FRANK. *Eclipse Modeling Framework: A Developer's Guide* [M]. Boston, USA: Addiseon-Wesley Professional, 2004.

**作者简介:**

**哈清华**(1983—),**男,吉林长春人,回族,助理研究员,博士研究生**,2007**年于哈尔滨工业大学获得硕士学位,主要从事软件测试方面的研究**。E-mail: haqinghuaha@hotmail.com

**导师简介:**

**刘大有**(1942—),**男,河北人,博士生导师,教授**,1981**年获吉林大学计算机科学系硕士学位,研究方向:人工智能、数据挖掘**。E-mail: liudy@jlu.edu.cn

**(版权所有 未经许可 不得转载)**

?1994-2016 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net