基于故障树分析与组合测试的测试用例生成方法

张卫祥 刘文红
(北京跟踪与通信技术研究所 北京 100094)

摘 要 随着软件的日益复杂,如何从巨大的可用测试用例空间中选择少量的测试用例进行有效的测试,成为了软件测试的重要课题之一。给出了一种易于自动化实现的软件测试用例生成方法。首先运用故障树分析原理,获取待测软件故障树的最小割集;然后利用等价类划分法、边界值分析法等经典黑盒测试方法,获取各最小割集中每个元素的典型取值集合;最后利用组合测试技术,在充分考虑各元素两两之间关系的基础上生成测试用例集。该方法对传统的故障树分析方法进行了改造,具有更大的适用范围;综合利用组合覆盖技术与黑盒测试方法,能有效保障测试充分性并减少测试用例数目。工程实践表明,该方法能够大幅提高测试效率。

关键词 软件测试,测试用例生成,组合测试,故障树分析,最小割集,测试用例集,软件工程
中图法分类号 TP311.5 **文献标识码** A

Test Suite Generation Based on Interaction Testing and Fault Tree Analysis

ZHANG Wei-xiang LIU Wen-hong
(Beijing Institute of Tracking and Telecommunications Technology, Beijing 100094, China)

Abstract Due to the increasing software complexity, how to select a few representative test cases to test software effectively has become an outstanding problem. This paper gave an integrative method for test suite generation, which can be divided into three parts, first uses an amended Fussell-Vesely algorithm to get minimal cut sets of the software, second uses black-box testing methods to obtain the typically discrete values of each element in minimal cut sets, finally uses an interaction testing algorithm to generate the test suite with a view to the interactional relationship between any two elements. Practice shows that the method can significantly reduce test cases count on the basis of ensuring software testing effect.

Keywords Software testing, Test suite generation, Interaction testing, Fault tree analysis, Minimum cut set, Test suite, Software engineering

1 引言

随着信息技术的迅速发展,计算机软件的应用日益广泛,软件失效造成的后果也愈加严重,特别是在航空航天、金融保险、交通通信、工业控制等重要领域,软件一旦失效将造成重大损失,对软件质量提出更高的要求。

软件测试是保障软件质量的重要手段,是构建高可信软件的关键环节。统计数据表明,软件测试占软件开发总成本的比例一般达到50%以上[1]。但随着待测软件的日益复杂和庞大,如何选择一组有效的测试方法以及如何从巨大的可用测试用例空间中选择少量的测试用例,对其进行有效的测试,是软件测试面临的重要课题之一,测试用例的选择与自动生成技术成为软件测试的重要研究方向。

目前已有不少关于故障树分析在软件测试中应用的成果[3-6],主要集中于故障树建模及最小割集算法,缺少基于最小割集生成测试用例的方法研究,往往会导致庞大的测试用例集,影响测试效率。组合测试考虑了系统中参数之间的相互作用,能够较好地提高软件缺陷的发现几率,是一种有效的软件测试用例生成技术[7-9],但对组合覆盖所必需的基础数据获取方法的研究尚不多见。

本文结合软件评测实践,提出了一种测试用例生成方法。该方法基于故障树分析原理,对传统的Fussell-Vesely算法进行了改造,使之能够适应对航天领域软件常用的"三判二"逻辑的处理,扩大了故障树分析法的适用范围;基于最小割集采用组合测试技术进行用例设计,考虑了多参数间的相互作用并覆盖了所有参数的两两组合关系,在提高测试充分性的同时还能够大幅削减测试用例的数量。本文首先介绍故障树分析法和改造Fussell-Vesely算法,然后给出基于最小割集与组合测试的用例生成算法,最后通过例子进一步说明该方法的使用流程及使用效果。

2 改造Fussell-Vesely算法

2.1 故障树分析与最小割集

故障树分析是一种系统化、形式化的分析方法,通过演绎手段找出导致系统故障的各种可能原因,利用故障树——倒立树状的逻辑因果关系图——表示故障和导致该故障的诸多

张卫祥(1979—),男,硕士,工程师,主要研究方向为软件工程、软件测试与质量保证,E-mail:wxchung@msn.com;刘文红(1968—),女,硕士,高级工程师,主要研究方向为软件工程。

因素之间的逻辑关系。利用故障树分析方法，有助于确定一个正确的或不正确的状态转变成不安全状态的环境、触发条件及其转移概率，从而可以有针对性地设计测试用例，找到系统的潜在缺陷和薄弱环节，最终达到减少故障发生可能、提高软件可靠性的目的。

故障树分析方法的一般过程为：首先，进行系统定义，确定故障树分析的范围并收集系统故障的资料；其次，进行顶事件选择，确定要分析的系统故障状态；第三，建立故障树，从顶事件到中间事件直至底事件，自上而下逐级分析各结果事件的原因；最后，识别导致系统失效的最小组元集合，即获取故障树的最小割集。如果需要，在得到全体最小割集后，可根据各底事件发生的概率确定顶事件发生的概率，对系统作进一步定性或定量分析。

获取故障树的最小割集是故障树分析的关键。Fussell-Vesely 算法由 Fussell 提出，是求解最小割集的标准算法。在工程实际中，特别是在航天软件领域，除了通常的与非门等逻辑关系外，常见的还有“三判二”逻辑关系[11,12]。“三判二”是航天测控中的惯用语，指的是：进行具有三个条件的命题判断时，当三个条件中至少两个条件成立时，命题为真；否则，命题为假。为适应“三判二”逻辑关系，我们对 Fussell-Vesely 算法进行改造。

2.2 改造 Fussell-Vesely 算法

加入对“三判二”逻辑关系处理后的改造 Fussell-Vesely 算法，其主要步骤如下所示：

a)建立一个仅含有故障树根节点 T 的集合 S，并以根节点 T 为当前节点，向下搜索当前节点的所有子节点；

b)若当前节点与其子节点之间是逻辑与运算，则把其所有子节点都加入到当前节点所在的集合中，并把当前节点从该集合中删除；若当前节点与其子节点之间是逻辑或运算，则将当前节点所在的集合复制该当前节点子节点的个数份，在所述复制的集合中分别加入一个子节点并删除当前节点；若当前节点与其子节点之间为三判二运算，则将当前节点所在的集合复制 4 份，删除当前节点并分别加入其 3 个子节点的如下组合之一：子节点 1，子节点 1 与子节点 2，子节点 1 与子节点 3，子节点 2 与子节点 3；

c)如果当前节点不为根节点并且当前节点已扩展完毕，则选择当前节点的父节点为当前节点；如果当前节点为根节点并且当前节点已扩展完毕，执行步骤 e)；

d)从当前节点的子节点集合中选择一个未扩展的子节点为当前节点，执行步骤 b)；若所有子节点都已扩展完毕，则标记当前节点为已扩展完毕，执行步骤 c)；

e)在已生成的所有集合中，分别剔除重复元素；

f)运用布尔代数中的吸收律 A+AB=A 和分配律 A+(B+C)=AB+AC，剔除多余集合，即得最小割集。

3 测试用例生成方法

运用故障树分析获得的最小割集为测试用例设计提供了很好的基础，但当系统比较复杂、最小割集中元素较多时，还需要结合有效的测试用例生成技术，否则仍然会导致庞大的测试用例数量。本文的方法在最小割集的基础上，综合采用黑盒测试与组合测试技术，能够取得较好的测试效果。

3.1 组合测试

待测软件出现故障的原因可能是由于某个参数的单独作用，但更多情况下是因为多个参数的相互作用。组合测试考虑了多个参数之间的相互关系，能够以较少的测试用例数量来实现对各种组合的覆盖[10]。

要使用组合测试方法，首先需找出待测软件中给定测试场景的各个参数，包括待测软件的状态、配置参数、用户输入参数或者外部事件参数等等，每个参数都有各自连续的或离散的取值范围。遍历所有这些参数的全部取值是不可能也是不必要的，组合测试的实质就是依据一定规则，对这些参数的部分组合进行覆盖测试。

一般地，根据覆盖程度的不同，可将其分为单因素覆盖、两两组合覆盖、三三组合覆盖等。单因素覆盖指的是对每个参数的各个取值进行覆盖，要求每个参数的各个取值都至少出现一次；两两组合覆盖要求对任意两个参数的所有取值组合进行覆盖，保证每一个取值组合都至少出现一次；三三组合覆盖要求对任意 3 个参数的所有取值组合进行覆盖；依此类推。覆盖要求越高，缺陷发现能力越强；但随着参数组合覆盖要求的增加，测试用例的数量也呈指数上升，测试代价越来越大。在实际应用中，两两组合覆盖最为常用[7]。在本文中，我们使用了两两组合覆盖，也可以将其推广到其它组合覆盖。

3.2 用例生成方法

我们的用例生成方法结合了故障树分析、黑盒测试与组合测试技术(如图 1 所示)，共有 4 个大步骤。

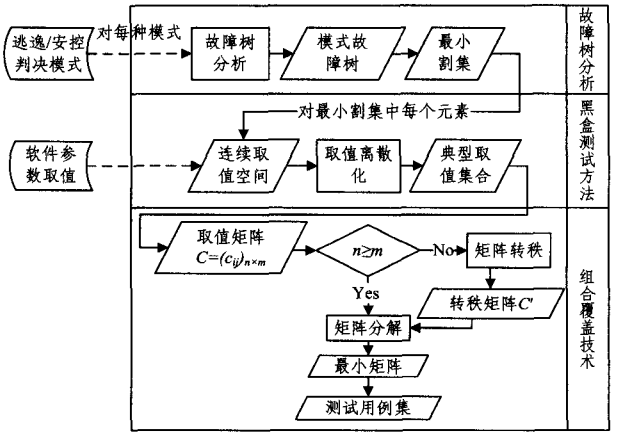

图 1 测试用例生成方法的主要流程

Step 1 根据故障树分析方法绘制待测软件故障树，运用改造 Fussell-Vesely 算法获取软件故障树的最小割集。

改造 Fussell-Vesely 算法已在上文介绍，此处不再详述。这里不妨假设共得到 p 个最小割集，分别为 $K_1, K_2, \cdots, K_p$。

Step 2 采用等价类划分、边界值分析等黑盒测试方法将最小割集中每个元素的取值范围离散化，得到各最小割集中每个元素的典型取值集合。

假设任一给定最小割集 K 中有 m 个元素 $c_1, c_2, \cdots, c_m$，分别取离散值 $T_1, T_2, \cdots, T_m$。令 $\alpha_j = |T_j|$ 表示 T_j 所含元素的个数，不失一般性，不妨设 $\alpha_m \leqslant \alpha_{m-1} \leqslant \cdots \leqslant \alpha_1 = \max\limits_{1\leqslant j\leqslant m} \alpha_j \equiv n$。

Step 3 采用组合覆盖技术，生成任意一个最小割集 K 的测试用例集。该步骤分为 3 小步。

1. 构造矩阵 $C=(c_{ij})_{n\times m}$，矩阵 C 由最小割集 K 中所有元素的典型取值组成。

具体方法是：从最小割集 K 的第一个元素 c_1 开始，将 c_1 的 n 个典型取值依次填入到矩阵 C 的第一列中，以此类推，将 c_j 的 α_j 个典型取值依次填入到矩阵 C 的第 j 列中；

若矩阵 C 中某列有空位，则将该列其它任意值填入到该列空位中，使得矩阵 C 的每列均有 n 个值；所述的空位为补充位，填入到空位中的值为补充值，则其它位为正常位，填入到正常位的值为正常值；

2. 展开矩阵 C，获得最小割集 K 的初始测试用例集 I_0。

该步骤根据 n 与 m 的大小关系，分为 2A 和 2B 两种情况：

2A $n \geqslant m$ 时，采用矩阵分解的方法将矩阵 C 展开。

2A.1 将矩阵 C 展开至最小矩阵 D

如果 $n>m$，则最小矩阵 D 是 $(n-m)\times 1$ 的矩阵；如果 $n=m$，则最小矩阵 D 是 2×2 的矩阵。

具体方法是：从矩阵 C 的第一列开始，依次相加，用该列的每一个值乘以划去该值所在行与列得到的矩阵，并循环递推，直至展开到最小矩阵 D；在此过程中，若在某相加项中除矩阵外的部分出现补充值，则把该项剔除。

2A.2 展开最小矩阵 D

将步骤 2A.1 所得到的相加项中的最小矩阵继续展开，具体做法是：

a) $n>m$ 时，如果最小矩阵中存在正常值，逐一分别取所有正常值与该相加项中除最小矩阵外的部分相乘，所得的每一个乘积即为一个测试用例，全部添加到初始测试用例集 I_0 中；否则，最小矩阵中全为补充值，则任选且仅选其中一个补充值与该相加项中除最小矩阵外的部分相乘，并把所得的乘积添加到初始测试用例集 I_0 中；

b) $n=m$ 时，按矩阵分解方法将最小矩阵展开，逐一与该相加项中除最小矩阵外的部分相乘，所得的每一个乘积即为一个测试用例，全部添加到初始测试用例集 I_0 中。

2B $n<m$ 时，首先将矩阵 C 转秩得到矩阵 C^{T}，然后采用矩阵分解的方法将矩阵 C^{T} 展开。该步骤分如下两步：

2B.1 将矩阵 C^{T} 展开至最小矩阵

最小矩阵是 $(m-n)\times 1$ 矩阵，具体做法同步骤 2A.1；

2B.2 展开最小矩阵

将最小矩阵中的各值逐一取出与除最小矩阵外的部分相乘，所得的每一个乘积即为一个测试用例，全部添加到初始测试用例集 I_0 中；

3. 对得到的初始测试用例集 I_0 进行调整，以获得最小割集 K 的最终测试用例集 I。

该步骤分如下两步：

3.1 在矩阵 C 的任意一行中，如果有至少两个正常值，则将该行中各值相乘，所得的乘积即为一个测试用例，添加到初始测试用例集 I_0 中；

3.2 如果在初始测试用例集 I_0 中至少有两个测试用例完全相同，则仅保留其中一个，剔除多余的，即得最小割集 K 的测试用例集 I。

Step 4　重复 Step 3，得到所有最小割集的测试用例集。所有最小割集的测试用例集的并集即为待测软件的测试用例集。

4　例子

通过一个例子进一步说明本方法的使用流程及其使用效果。本例子取自某地面逃逸软件的一个判决模式，为了方便，对其进行了一般化处理。

假设已绘制故障树，在该故障树中，存在“三判二”的逻辑，如图 2 所示。利用本文测试用例生成方法，主要步骤如下。

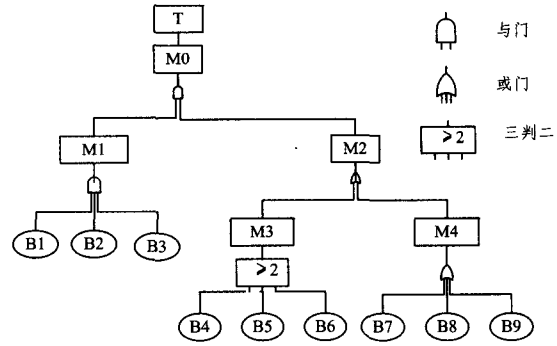

图 2　软件故障树示例

首先，依据改造 Fussell-Vesely 算法获取故障树的最小割集，具体如图 3 所示。

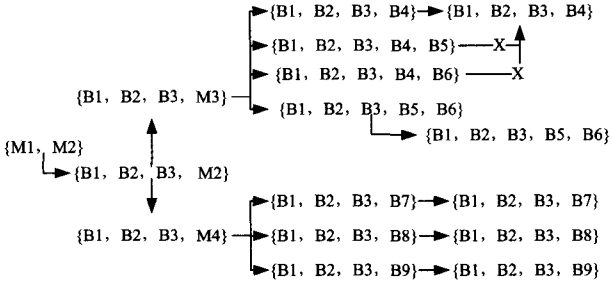

图 3　获取最小割集

1)建立集合 $S=\{T\}$，设置当前节点为 T；

2)搜索 T 节点的所有子节点，由于 $M0$ 是逻辑与运算，把 T 的所有子节点 $M1$ 与 $M2$ 都加入到 T 所在的集合中，并把 T 从该集合中删除，则集合 $\{T\}$ 变为集合 $\{M1,M2\}$；

3)从 T 的子节点集合中选择节点 $M1$ 为当前节点；由于节点 $M1$ 是逻辑与运算，把节点 $M1$ 的所有子节点都加入到节点 $M1$ 所在的集合中，并把节点 $M1$ 从该集合中删除，因此集合 $\{M1,M2\}$ 变为 $\{B1,B2,B3,M2\}$；由于节点 $B1$、$B2$、、$B3$ 都是叶子节点，因此节点 $M1$ 扩展完毕；

4)由于节点 $M1$ 不是根节点，选择其父节点 T 的未扩展子节点 $M2$ 为当前节点；由于节点 $M2$ 是逻辑或运算，将节点 $M2$ 所在的集合依据子节点的数目复制两份，并在复制的两个集合中分别加入一个子节点并删除节点 $M2$，因此集合 $\{B1,B2,B3,M2\}$ 变为集合 $\{B1,B2,B3,M3\}$ 与集合 $\{B1,B2,B3,M4\}$；

5)从节点 $M2$ 的子节点集合中选择 $M3$ 为当前节点；由于节点 $M3$ 是“三判二”运算，则将节点 $M3$ 所在的集合复制 4 份，并在每个集合中分别加入指定组合并删除节点 $M3$，则集合 $\{B1,B2,B3,M3\}$ 变为 $\{B1,B2,B3,B4\}$、$\{B1,B2,B3,B4,B5\}$、$\{B1,B2,B3,B4,B6\}$、$\{B1,B2,B3,B5,B6\}$；

6)由于节点 $M3$ 不为根节点并且已扩展完毕，因此选择其父节点 $M2$ 作为当前节点；搜索 $M2$ 的子节点，还有 $M4$ 未扩展，故设置当前节点为 $M4$；由于节点 $M4$ 为逻辑与运算，因此集合 $\{B1,B2,B3,M4\}$ 变为集合 $\{B1,B2,B3,B7\}$、$\{B1,B2,B3,B8\}$、$\{B1,B2,B3,B9\}$；

7)节点 $M4$ 扩展完毕后，其父节点 $M2$ 也扩展完毕，选择 $M2$ 的父节点 T 为当前节点；由于节点 T 为根节点且已扩展完毕，整个扩展过程结束。检查所有的集合，并运用布尔代数规则中的吸收律和分配律进行整理，得到 5 个最小割集：

$\{B1,B2,B3,B4\}$、$\{B1,B2,B3,B5,B6\}$、$\{B1,B2,B3,B7\}$、$\{B1,B2,B3,B8\}$、$\{B1,B2,B3,B9\}$。

其次，采用等价类划分、边界值分析等黑盒测试方法将最小割集中每个元素的取值范围离散化，得到各最小割集中每个元素的典型取值集合；然后，采用组合覆盖技术，获得各最小割集的测试用例集。

以最小割集 $K=\{B_1,B_2,B_3,B_4\}$ 为例，假设其典型取值分别为 $B_1:a_1,a_2,a_3,a_4$；$B_2:b_1,b_2,b_3$；$B_3:c_1,c_2,c_3$；$B_4:d_1,d_2$。则按照算法，可得到矩阵 C 并逐步展开，其中 * 表示空位记号：

$$C=\begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & \\ a_4 & & & \end{bmatrix}=\begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_1^* \\ a_4 & b_1^* & c_1^* & d_2^* \end{bmatrix}$$

$$=\cdots$$

$$=a_1b_2\begin{bmatrix} c_3 & d_1^* \\ c_1^* & d_2^* \end{bmatrix}+a_1b_3\begin{bmatrix} c_2 & d_2 \\ c_1^* & d_2^* \end{bmatrix}+$$

$$a_2b_1\begin{bmatrix} c_3 & d_1^* \\ c_1^* & d_2^* \end{bmatrix}+a_2b_3\begin{bmatrix} c_1 & d_1 \\ c_1^* & d_2^* \end{bmatrix}+$$

$$a_3b_1\begin{bmatrix} c_2 & d_2 \\ c_1^* & d_2^* \end{bmatrix}+a_3b_2\begin{bmatrix} c_1 & d_1 \\ c_1^* & d_2^* \end{bmatrix}+$$

$$a_4b_1\begin{bmatrix} c_2 & d_2 \\ c_3 & d_1^* \end{bmatrix}+a_4b_2\begin{bmatrix} c_1 & d_1 \\ c_3 & d_1^* \end{bmatrix}+a_4b_3\begin{bmatrix} c_1 & d_1 \\ c_2 & d_2 \end{bmatrix}$$

继续展开最小矩阵，可得：

$$C=a_1b_2(c_3d_2^*+c_1^*d_1^*)+a_1b_3(c_2d_2^*+c_1^*d_2)+a_2b_1(c_3d_2^*+c_1^*d_1^*)+a_2b_3(c_1d_2^*+c_1^*d_1)+a_3b_1(c_2d_2^*+c_1^*d_2)+a_3b_2(c_1d_2^*+c_1^*d_1)+a_4b_1(c_2d_1^*+c_3d_2)+a_4b_2(c_1d_1^*+c_3d_1)+a_4b_3(c_1d_2+c_2d_1)$$

$$=a_1b_2c_3d_2^*+a_1b_2c_1^*d_1^*+a_1b_3c_2d_2^*+a_1b_3c_1^*d_2+a_2b_1c_3d_2^*+a_2b_1c_1^*d_1^*+a_2b_3c_1d_2^*+a_2b_3c_1^*d_1+a_3b_1c_2d_2^*+a_3b_1c_1^*d_2+a_3b_2c_1d_2^*+a_3b_2c_1^*d_1+a_4b_1c_2d_1^*+a_4b_1c_3d_2+a_4b_2c_1d_1^*+a_4b_2c_3d_1+a_4b_3c_1d_2+a_4b_3c_2d_1$$

因此，得到初始测试用例集为：

$$I_0=\{a_1b_2c_3d_2^*,a_1b_2c_1^*d_1^*,a_1b_3c_2d_2^*,a_1b_3c_1^*d_2,a_2b_1c_3d_2^*\}\cup\{a_2b_1c_1^*d_1^*,a_2b_3c_1d_2^*,a_2b_3c_1^*d_1,a_3b_1c_2d_2^*,a_3b_2c_1d_2^*\}\cup\{a_3b_1c_1^*d_2,a_3b_2c_1^*d_1,a_4b_1c_2d_1^*,a_4b_1c_3d_2\}\cup\{a_4b_2c_1d_1^*,a_4b_2c_3d_1,a_4b_3c_1d_2,a_4b_3c_2d_1\}$$

经过补充 $a_1b_1c_1d_1$，$a_2b_2c_2d_2$，$a_3b_3c_3d_1^*$ 并剔除多余项，得到最小割集 K 的测试用例集：

$$I_1=\{a_1b_2c_3d_2,a_1b_2c_1d_1,a_1b_3c_2d_2,a_1b_3c_1d_2,a_2b_1c_3d_2\}\cup\{a_2b_3c_1d_1,a_3b_1c_2d_2,a_3b_2c_1d_1,a_4b_1c_3d_2\}\cup\{a_4b_1c_2d_1,a_4b_2c_3d_1,a_4b_3c_1d_2,a_4b_3c_2d_1\}\cup\{a_1b_1c_1d_1,a_2b_2c_2d_2,a_3b_3c_3d_1,a_4b_2c_1d_1\}$$

I_1 共 17 个测试用例。相比于采用全覆盖需要 $4\times3\times3\times2=72$ 个测试用例，测试用例的数量缩减了 76.4%。

最后，采用同样的方法得到其余 4 个最小割集的测试用例集 I_2,I_3,I_4,I_5，则全部测试用例的集合为 $I=I_1\cup I_2\cup I_3\cup I_4\cup I_5$。

至此，得到了本实例的测试用例集。从表 1 可以看到，本方法对测试用例的缩减效果非常明显。事实上，被测软件的参数越多、各参数的取值越复杂，测试用例的缩减效果就越好。

在测试充分性方面，从理论上讲，本文方法弱于全覆盖测试。但是，在工程实际中，全覆盖是不可行的，而相比于随机选取或经验选取等其它方法，本文方法从方法层面保证了对两两参数组合的覆盖（不会产生漏项），不是降低而是大幅增强了测试充分性。工程实践表明，本文方法可大幅提高测试效率并取得很好测试效果。

表 1　测试用例集比较

	全覆盖方法	本文方法	缩减数	缩减比
$K_1=\{B_1,B_2,B_3,B_4\}$ $B_1:a_1,a_2,a_3,a_4$；$B_2:b_1,b_2,b_3$ $B_3:c_1,c_2,c_3$；$B_4:d_1,d_2$	72	17	55	76.40%
$K_2=\{B_1,B_2,B_3,B_5,B_6\}$ $B_5:e_1,e_2,e_3,e_4,e_5$ $B_6:f_1,f_2,f_3,f_4,f_5,f_6$	1080	89	991	91.80%
$K_3=\{B_1,B_2,B_3,B_7\}$ $B_7:g_1,g_2,g_3,g_4,g_5,g_6,g_7$	252	58	194	77.00%
$K_4=\{B_1,B_2,B_3,B_8\}$ $B_8:h_1,h_2,h_3,h_4,h_5,h_6,h_7,h_8$	288	70	218	75.70%
$K_5=\{B_1,B_2,B_3,B_9\}$ $B_9:i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8,i_9$	324	82	242	74.70%
小计	2016	316	1700	84.30%

结束语　软件测试领域存在着很多困难的问题，比如测试方法的选择、测试用例的约简以及测试何时终止等，其根本原由是测试充分性与测试代价之间的矛盾。以最小的测试代价获得最好的测试效果，是软件测试研究的最终目标。

本文给出了一种基于故障树分析与组合测试的用例生成方法，并给出了具体的算法步骤，举例说明了其效果。通过把测试需求转化为最小割集并采用组合测试来覆盖多参数的两两组合，该方法能够较好地保证测试充分性并大幅削减测试用例数量，在工程实践中取得了不错的效果。今后，我们将就测试用例约简及最优测试用例集生成技术做进一步研究。

参 考 文 献

[1] Ammann P, Offutt J. Introduction to Software Testing [M]. Cambridge University Press, 2008

[2] 朱继洲. 故障树原理及应用[M]. 西安：西安交通大学出版社，1989

[3] 肖英柏，向剑文，徐仁佐. 软件可靠性稳定增长与安全性测试的故障树分析法[J]. 小型微型计算机系统，1999，20(9)：668-671

[4] 刘文红. 故障树分析技术在软件测试中的应用[J]. 系统工程与电子技术，2004，26(7)：985-988

[5] 胡智，殷人昆. 基于最小割集的安全性测试用例的动态生成[J]. 计算机工程与设计，2006，27(16)：3018-3020

[6] 朱云鹏. 基于故障树分析法的软件测试技术研究[J]. 计算机工程与设计，2008，29(13)：3387-3390

[7] 张卫祥，刘文红. 一种基于组合覆盖的黑盒测试用例自动生成方法[J]. 飞行器测控学报，2008，27(5)：53-56

[8] 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[J]. 计算机学报，2004，27(3)：382-388

[9] Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pair-wise covering designs for software testing[J]. Information Processing Letters, 2002, 81(2): 85-91

[10] 陈翔，顾庆，王新平，陈道蓄. 组合测试研究进展[J]. 计算机科学，2010，37(3)：1-5

[11] 何国伟. 软件可靠性[M]. 北京：国防工业出版社，1998

[12] 张卫祥，刘文红，杜会森. 基于软件测试与知识发现的软件定量评估方法[J]. 计算机科学，2012，39(11A)：28-30