文章编号：1672-2892(2010)05-0594-04

故障树分析测试用例生成技术研究与应用

漆莲芝，张 军，谢 敏

(中国工程物理研究院 电子工程研究所，四川 绵阳 621900)

摘 要：文章对采用故障树分析技术造成系统失效的各种软件因素进行分析，确定造成系统失效的各种原因组合。通过构造软件系统故障树，生成故障树的最小割集，分析各个割集的安全性约束条件，由安全性约束条件分析测试数据，最后生成测试用例的算法。将故障树分析技术用于指导软件测试设计，体现了以系统工程方法研究软件测试的系统性、准确性和预测性。

关键词：故障树分析；故障树最小割集；安全性约束条件；测试用例

中图分类号：TN91；TP311.5 **文献标识码：**A

Research and application on FTA testing case generating technology

QI Lian-zhi，ZHANG Jun，XIE Min

(Institute of Electronic Engineering，China Academy of Engineering Physics，Mianyang Sichuan 621900，China)

Abstract：This paper discusses Fault Tree Analysis(FTA) technology, uses FTA to analyze a variety of software elements causing software system failure, and constructs software system fault tree through determining all software failure combination of reasons. Then generates the minimum cut sets of fault tree, analyses the security constraint conditions of cut sets, and analyzes the testing data according to the security constraint conditions. Thereafter, the testing case algorithm is generated. Practice shows the fault tree analysis technique can be used to guide the design of software testing, reflecting the systematic, accuracy and predictability of systems engineering methods to study software testing.

Key words：Fault Tree Analysis；the minimum cut sets；security constraint condition；testing case

软件测试是一项批判性的工作[1]，随着当今软件规模和复杂性以及软件在整个系统中的重要性日益增加，进行专业化、高效的软件测试的要求越来越迫切。测试用例设计是软件测试的核心，是软件测试过程必须遵守的准则，也是软件测试活动中最重要的部分之一。只有采用合适的测试用例设计方法，才能设计出高效和测试充分的测试用例。故障树分析(FTA)[2]技术是通过对可能造成产品故障的硬件、软件以及人为等因素进行分析，画出产生故障原因的各种可能组合方式和其发生概率的一种分析技术。故障树分析技术广泛应用于各个领域，包括核工业、航空、航天、机械、电子、兵器、船舶、化工等。本文中被测软件是某武器型号关键级别为 A 的软件分系统，该分系统中对于软件可控的故障，输入参数较多，参数配置和故障判决条件较为复杂，要保障测试的充分性和测试用例的有效性比较困难，因此本文将 FTA 技术用于指导软件测试用例设计。将 FTA 技术用于指导软件测试设计，有助于以系统工程方法研究软件测试。

1 构造故障树

故障树分析法[3]是把系统不希望发生的故障，通过分析寻找出导致系统故障发生的所有可能直接原因，接着分析寻找每一个直接原因发生的所有可能原因，以此类推，直至追踪到最后一级基本事件。软件缺陷是软件产品的固有成分，它的存在对软件功能的实现是一个潜在威胁[4]，因此可以使用故障树对造成产品故障的各种可能的因素进行分析，如硬件、软件、环境或人为因素等，通过故障树确定导致产品故障的可能原因组合方式。

本文采用故障树分析法从软件系统故障即顶事件出发，以逐层细化的方式从分支结构中找出所有导致软件故障的直接和间接原因。如图 1 所示，具体的故障树的构造过程为，从系统顶事件 T 出发，导致 T 发生的所有可

收稿日期：2010-03-22；修回日期：2010-04-17

能原因(包括直接原因和间接原因)为中间事间 M_1 和 M_2，M_1 和 M_2 由或门连接，表示系统中若出现事件 M_1 和 M_2 之一或两者同时出现，均会导致顶事件的发生。以分支结构和自顶向下逐层细化的方式分别寻找出导致 M_1 和 M_2 的所有直接原因和间接原因，以此类推，直到找到所有的最后一级基本事件为止。通过故障树有助于分析软件系统的所有可能发生的故障模式和原因，故障树的底事件直观地反映了软件系统故障点和各模块对系统产生影响的传播路径，而各种逻辑门形式化地表示了各个事件间的内在联系和相互作用。

本文将底事件和顶事件分别由 x_i 和 T 表示。

底事件表示为：

$$x_i=\begin{cases}1 & \text{底事件}x_i\text{发生}\\ 0 & \text{底事件}x_i\text{不发生}\end{cases}$$

当 T 取 1 表示顶事件发生，取 0 表示顶事件不发生。故障树结构函数表示为：$T=T(\vec{X})$，其中 $\vec{X}=(x_1,x_2,\cdots,x_n)$。

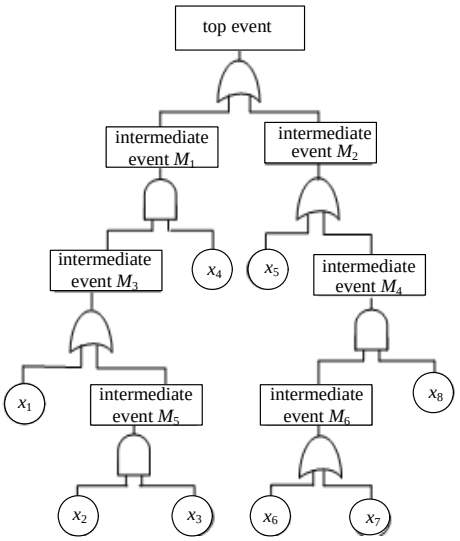

Fig.1 Fault tree

图 1 故障树

2　最小割集

所谓割集[5]是导致软件失效的一组事件的集合，当这些事件全部发生时，顶事件必然发生，不能再简化的割集被称为最小割集。在一个最小割集中，只要存在一个事件没有发生，就不能促使顶事件的发生。因此，如果能使所有最小割集中至少有一个底事件不发生或发生概率很低，则顶事件也不会发生或发生概率极低，与顶事件对应的故障模式也不会发生。反之，一旦系统中某一故障模式发生了，则一定是系统中与故障模式对应的某一个最小割集中的底事件发生了。最小割集是系统的基本故障模式，对于降低系统潜在的风险具有重大意义，同时，最小割集还可用于指导系统诊断与系统故障的维修。最小割集的求解方法一般有上行法(Semanderes 算法)、下行法(Fussell-Vesely 算法)和质数法。本文基于 Fussell-Vesely 算法[6]求解故障树的最小割集，其基本算法如下：

1) 从故障树的顶事件自顶向下，若是或门则将其输入事件列入不同的行，若是与门则将其输入事件列入同一行，直到将所有底事件列入为止。

2) 将分解的集合中的事件表示成结构函数的形式，运用布尔代数规则中的结合律和分配律进行等价变换，得到最小割集的组合，其中：

结合律为：$(x_1\cup x_2)\cup x_3=x_1\cup x_2\cup x_3$，$(x_1\cap x_2)\cap x_3=x_1\cap x_2\cap x_3$

分配律为：$(x_1\cap x_2)\cup(x_1\cap x_3)=x_1\cap(x_2\cup x_3)$，$(x_1\cup x_2)\cap(x_1\cup x_3)=x_1\cup(x_2\cap x_3)$

将图 1 所示的故障树，按 Fussell-Vesely 算法一共生成 5 个最小割集，分别是 $k_1=\{x_1,x_2\}$, $k_2=\{x_2,x_3,x_4\}$, $k_3=\{x_5\}$, $k_4=\{x_6,x_8\}$, $k_5=\{x_7,x_8\}$，最小割集的生成过程如表 1 所示。阶数越小的最小割集越重要，即 k_5 最重要；低阶最小割集中的底事件较高阶最小割集中的底事件更重要，如 k_1 中的事件 x_4 比 k_2 中的 x_3 重要。对于阶数相同的割集，重复出现次数越多的底事件越重要，如 x_2 在 k_1 和 k_2 中均出现，则 x_2 较为重要。

表 1 最小割集的生成过程

Table1 Generation of minimal cut sets

step	1	2	3	4	5
process	M_3,x_4 M_2	M_3,x_4 x_5 M_4	x_1,x_4 M_5,x_4 x_5 M_6,x_8	x_1,x_4 x_2,x_3,x_4 x_5 M_6,x_8	x_1,x_4 x_2,x_3,x_4 x_5 x_6,x_8 x_7,x_8

3　安全性约束条件

在某种程度上，可以认为最小割集是安全性需求的具体表现，可以通过安全性约束条件来阻止所有最小割集的出现，使得输出满足软件的安全性需求[7]。以图 1 所示的故障树为例，顶事件为 T 的故障树的安全性约束条件 C 就是顶事件 T 不能出现，即 $C=\neg T$。以某个最小割集为例，从顶事件的结点自上而下逐层扩展，就形成了一个系统的安全性需求。不失一般性，一个系统中的割集为 $T=\{k_1,k_2,\cdots,k_n\}$，则系统的安全性约束条件 $C=\neg T=\neg(k_1\wedge k_2\wedge\cdots\wedge k_n)=\neg k_1\vee\neg k_2\vee\cdots\vee\neg k_n$。

最小割集是软件系统安全性需求的具体表现，在一个最小割集中，只要有一个事件没有发生，则顶事件就不能发生，由此可以通过安全性约束条件来满足安全性需求。不失一般性，如果一个故障树的最小割集为 $k_i=\{x_1, x_2, \cdots, x_n\}$，假设最小割集中的各个事件都是软件可控的，则软件必须满足的安全性约束条件是 $C_i=\neg(x_1 \wedge x_2 \wedge \cdots \wedge x_n)=\neg x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n$。

以上面的软件故障树为例，软件系统的总的安全性约束条件 $C=\neg(k_1 \wedge k_2 \wedge k_3 \wedge k_4 \wedge k_5)=\neg k_1 \vee \neg k_2 \vee \neg k_3 \vee \neg k_4 \vee \neg k_5$。以最小割集 k_2 为例，假设 k_2 中的各个事件都是软件可控的，则软件必须满足安全性约束条件 $C_2=\neg(x_2 \wedge x_3 \wedge x_4)=\neg x_2 \vee \neg x_3 \vee \neg x_4$。

4 测试用例生成算法

综上所述，根据故障树求出最小割集，在最小割集的基础上运用安全性约束条件对所有测试数据进行划分，自动产生软件测试用例。可以将所有用于测试软件系统的数据，进一步划分为不同类型的测试数据，所有这些测试数据对应于所有最小割集的各个安全性需求侧面，使得软件的安全性输出约束条件都得到充分测试。在形成安全性约束条件后，可将测试数据划分为 3 种类型，分别是：不包含在任何约束条件内的输入数据集合，当这些数据输入后，软件系统不会产生功能失败，即软件功能成功实现；部分包含在安全约束条件中的数据集，这些输入数据可能会导致软件功能失败，但是这些数据可以用于验证软件是否进行了有效的防护；最后一种就是完全包含在安全性约束条件中的数据，当这些数据出现时，必然会导致软件故障树中顶事件的发生，即出现软件故障。以本文故障树的最小割集 k_2 为例，假设 D 是所有用于测试软件系统的安全性数据，$D=\{d_i \mid d_i$ 是所有用于测试软件的测试数据$\}$，函数 $Enable\ x_i(d_j)$ 表示使得割集 k_i 中的事件 x_j 发生，k_2 对应的安全性约束条件为：

$$\forall d_1, d_2, d_3 \in D,\ C_2 = (d_1 \neq d_2 \neq d_3) \wedge (\neg Enablex_2(d_1) \vee (\neg Enablex_3(d_2) \vee \neg Enablex_4(d_3))$$

测试用例的生成算法可表示为：

1) 从测试数据输入空间中任取一个满足条件的而且还未被用作测试数据的输入向量作为测试用例输入数据；

2) 检查软件功能是否能成功实现，若软件功能成功实现，说明当前测试用例不会导致顶事件发生，否则转移到 3)；

3) 检查软件输出是否导致顶事件的发生，若未导致顶事件发生，说明软件进行了有效性防护，否则转移到 4)；

4) 检查软件输出是否导致顶事件的发生，若是，则说明当前测试用例会导致软件功能失败；

5) 检查割集中各事件是否覆盖完全，若是，则结束本算法，否则转移到 1)。

5 应用

软件质量侧重于产品、过程和项目的质量细节[8]，软件测试是提高软件质量的必要条件。本文所述被测软件是某武器型号的关键级别为 A 的分系统，硬件模块多，软件规模大，参数配置和故障判决条件较为复杂。为了实现对该软件分系统的充分测试，作者所在软件评测中心依据有关技术规范，对软件的需求规格说明书进行划分，采用 FTA 技术，对软件系统的各种可能失效的原因进行分析与组合。在故障树基础上生成了软件系统的所有最小割集，运用上述基于割集的测试用例生成算法，结合黑盒测试等价类划分方法对软件进行测试，发现软件缺陷 78 个，其中反映的典型问题是软件系统的安全联锁设计比较欠缺，存在隐患，如工作流程没有相互限制、手动操作与自动操作没有相互限制等。软件测试结果如表 2 所示。

表 2 软件测试结果分类表

Table2 Classification results of software testing

software defect types	number of defects	percentage/(%)
adverse impact on the main function, and no alternative solution	18	23.08
adverse impact on the main function, but alternative solution	34	43.59
bring inconvenience or trouble, but did not affect the operation or task requested	12	15.38
other problems	14	17.95

6　结论

测试结果表明，将 FTA 技术用于指导软件测试设计，实现了以系统工程方法研究软件测试技术，增强了软件测试的系统性、准确性和预测性，同时有助于了解软件系统的各功能模块的组成和分析软件失效的各种因素。

参考文献：

[1] Ron Patton. 软件测试[M]. 张小松,王钰,曹跃,等,译, 北京:机械工业出版社, 2006.
[2] Kirsten M Hansen,Anders P Ravn,Victoria Stavridou. From safety analysis to software requirements[J]. IEEE Transactions on Software Engineering, 1998,24(7):573-584.
[3] 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.
[4] 漆莲芝. 基于过程度量的软件测试质量管理[J]. 计算机测量与控制, 2008,16(7):935-938.
[5] 徐中伟,吴芳美. 形式化故障树分析建模和软件安全性测试[J]. 同济大学学报, 2001,29(11):1329-1302.
[6] 郑明. 计算机系统脆弱性评估方法研究[D]. 哈尔滨:哈尔滨工业大学, 2005.
[7] 王闯,刘东飞. 故障树分析技术在软件系统测试中的应用[J]. 软件导刊, 2008,7(1):78-79.
[8] 漆莲芝,高峰. 基于 GQM 的软件过程性能度量与分析模型[J]. 计算机应用研究, 2009,26(z1):785-786.

作者简介：

漆莲芝(1978-)，女，成都市人，硕士，主要研究方向为计算机软件与理论.email:qlzscu@126.com.

张　军(1972-)，男，四川省资阳市人，硕士，主要研究方向为嵌入式软件测试.

谢　敏(1981-)，男，四川省三台县人，硕士，主要从事测控技术、测控系统应用、面向信号处理的可配置计算技术研究.

（上接第 587 页）

[6] Wiznet. High-Performance Internet Connectivity Solution w5300[Z]. w5300 version 1.2.2, 2008.
[7] 王磊. 基于 MicroBlaze 软核的 FPGA 片上系统设计[J]. 单片机及嵌入式系统应用, 2004(7):52-54. (WANG Lei. System on chip design based on FPGA with MicroBlaze soft-core[J]. Microcontroller and Embedded Systems, 2004.)
[8] 汉泽西,孙燕妮. DSP+FPGA 技术[J]. 电子技术, 2007,34(2):18-21. (HAN Zexi,SUN Yanni. DSP+FPGA technology[J]. Electronics, 2007.)

作者简介：

王　龙(1986-)，男，四川省巴中市人，在读硕士研究生，主要研究方向为基于 FPGA 平台的数字信号处理.email:09210720102@fudan.edu.cn.

陈晓光(1964-)，男，安徽省五河市人，博士，副教授，主要研究方向为移动通信及 RF 技术.