一种软件故障诊断过程框架

单锦辉[1),2)] 徐克俊[2)] 王 戟[3)]

[1)](北京特种工程设计研究院 北京 100028)

[2)](中国酒泉卫星发射中心 甘肃 酒泉 732750)

[3)](国防科学技术大学计算机学院 长沙 410073)

摘 要 软件在国民经济和社会生活中发挥着重要作用.软件出现故障给人们的工作、生活带来不便,甚至造成严重危害.但是当前所进行的多为软件故障诊断中单项活动的研究,较少有对各项诊断活动及其相应方法进行有效集成的研究.文中分析软件失效机理和软件故障产生原因,讨论软件故障模型,提出一种由故障检测、故障定位、故障排除、交付等组成的集成化的软件故障诊断过程框架,研究软件故障检测、定位和排除中所采用的方法及相应的过程,并且将该框架应用于实际的软件故障诊断.

关键词 软件故障;故障诊断;故障检测;故障定位;故障排除

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.00371

A Kind of Software Faults Diagnosing Framework

SHAN Jin-Hui[1),2)] XU Ke-Jun[2)] WANG Ji[3)]

[1)](*Beijing Special Engineering Design and Research Institute*, *Beijing* 100028)

[2)](*Jiuquan Satellite Launching Center*, *Jiuquan*, *Gansu* 732750)

[3)](*School of Computers*, *National University of Defense Technology*, *Changsha* 410073)

Abstract Software plays an important role in our society. The occurrence of software fault will bring inconvenience to one's work and life, even lead to severe disaster. However, the majority of research on software faults diagnosis focuses on single diagnosing activity at present, and there is little research on effective integration of the diagnosing activities yet. This paper analyzes the mechanism of software failure and the cause of software faults, discusses software fault models. A kind of integrated software fault diagnosing framework is proposed, which is composed of fault detection, fault localization, fault removal, and software release. Then various kinds of methods and corresponding process of fault detection, fault localization, and fault removal are studied. The proposed framework is applied to real software fault diagnosis to validate its effectiveness.

Keywords software fault; fault diagnosis; fault detection; fault localization; fault removal

1 引 言

随着社会的不断进步和计算机科学技术的飞速发展,计算机在国民经济和社会生活中的应用越来越广泛.作为计算机的灵魂,软件起着举足轻重的作用.随着软件逐渐取代部分硬件功能以及一些大型工程呈现软件化的趋势,软件在整个计算机系统中

收稿日期:2008-01-03;最终修改稿收到日期:2010-06-30.本课题得到国家自然科学基金(60970022)资助.**单锦辉**,男,1970 年生,博士,高级工程师,主要研究方向为软件测试、软件质量管理、软件构件技术等. E-mail: shanjh@163.com. **徐克俊**,男,1943 年生,教授级高级工程师,主要研究领域为航天发射故障诊断等. **王 戟**,男,1969 年生,博士,教授,博士生导师,主要研究领域为高可信软件、软件工程与分布式计算等.

所占比重越来越大[1].人们对软件质量要求越来越高.许多计算机科学家在讨论21世纪计算机科学发展方向和策略时,将提高软件质量放在优先于提高软件功能和性能的地位[2].软件一旦出现故障,轻则给人们工作、生活带来不便,重则可能造成巨大的经济损失,甚至危及人的生命安全.例如,1996年Ariane 5运载火箭的发射失败是由软件故障引起的.因此,研究软件故障诊断具有重要意义.

目前国内外开展的与软件故障诊断有关的研究工作,包括针对软件故障概念[3-7]、软件故障模型[8-11]、软件故障描述方法[12-13]、软件故障检测方法[14-20]、软件故障定位方法[21-26]、软件故障预防方法[5,27-29]等的研究.

在软件故障概念方面,文献[3]将软件故障定义为计算机程序中不正确的步骤、处理或者数据定义.文献[5]将软件故障定义为软件运行过程中出现的一种不希望或不可接受的内部状态.文献[6]则将软件故障区分为语法大小和语义大小,语法大小为受一个故障影响的代码行数,语义大小为其输出结果不正确的输入空间的大小.文献[7]将软件故障定义为软件系统中的结构不完善,它可能导致系统的最终失效.

在软件故障模型方面,已有研究工作根据错误发生阶段、故障引起后果、错误性质、错误类型等建立软件故障模型[14].文献[8]针对科学计算程序的软件建立计算型、分支型、循环型、功能型、死锁型、测试型等几种故障模型.文献[9]针对C语言程序建立内存泄漏等几种故障模型.文献[10]提出面向侧面(AOP)编程语言AspectJ的Pointcut的故障模型.

在软件故障描述方面,文献[12]采用调用序列描述软件故障.文献[13]对已有各种描述软件运行正确结果的方法进行分类和总结.

软件测试是一种重要的软件故障检测手段.文献[20]介绍软件测试的基本思想,讨论面向路径的测试数据自动生成[17]等问题,探讨软件测试的发展趋势.代码审查是一种重要的软件测试方法.检查表是代码审查的核心.文献[16]针对C、PL/M、汇编语言所编写的程序容易出现的故障,指出检查表应该包括的一些内容.文献[19]研究在对源代码进行编译之前自动检测和改正静态数组下标越界、函数调用时参数类型不正确、switch语句无default处理三类故障.模型检验也是一类重要的软件故障检测手段[15].

在软件故障定位方面,文献[21]提出一种组合运行产生正确结果和错误结果的两组测试用例的中间状态,采用Delta调试算法[22]从空间上定位引起故障的变量,从时间上定位引起软件故障的时刻.文献[23]提出一种方法将Delta调试算法与前向、后向动态切片方法结合起来,以缩小搜索有缺陷代码范围的方法.文献[24]对软件组合测试基本模型进行研究,并提出一种基于组合测试的软件故障定位方法.文献[25]采用谓词切换的思想定位软件故障.文献[26]提出一种优化测试用例集的方法,使得它具备较强的软件故障定位能力.

当前硬件系统故障诊断研究和实践取得较大进展,形成基于解析模型、信号处理、人工智能等的故障诊断方法[30-32],并有大量成功案例.硬件系统故障诊断过程包括故障检测、故障分离、故障治理与预防等步骤.故障检测,包括状态监测,是采集系统运行参数、获取系统状态信息、识别系统是否存在故障而进行的检查和测试过程.故障分离,或称分析与隔离,是根据故障检测所获得的系统状态信息,进行分析,包括故障分析,以确定系统是否存在故障,识别故障特性,进行故障隔离,即对故障定位.治理与预防,是指根据故障诊断的结果,按照已识别的系统状态的不同情况,或者针对故障采取纠正措施,彻底消除故障;或者预测未来、决定防治潜在故障发生的维修对策.

软件故障诊断研究较之硬件系统故障诊断起步较晚.软件种类繁多,其功能各式各样,内部状态、路径高度复杂[14],运行时内部不透明,出现故障时外部表现形式复杂多样,致使软件故障诊断比较困难.并且从软件故障诊断研究进展可知,目前所进行的多为软件故障诊断中单项活动的研究,较少有对各项诊断活动及其相应方法进行有效集成的研究,使得软件故障诊断过程缺乏系统的框架指导,比较盲目.

本文借鉴硬件系统故障诊断的思路,研究将软件故障检测、定位、排除这些活动及其相应方法集成在一起的软件故障诊断过程框架,为规范地运用各种方法、系统地进行软件故障诊断提供指导.

2 软件失效机理与软件故障产生原因

2.1 软件失效机理

研究软件失效机理是软件故障诊断的前提.由于软件内部逻辑复杂,运行环境动态变化,且不同的软

件差异很大，因而软件失效机理常常有不同的表现形式. 但总的来说，软件失效机理可以描述为：软件错误→软件缺陷→软件故障→软件失效的过程[5].

让 D,R 分别为一个可数的集合，代表软件的输入、输出数据集合；P 为从 D 到 R 的可计算函数集合的子集，它代表以 D 和 R 中的数据为输入和输出的程序集合；S 为从 D 到 R 的二元关系的一个子集，它代表软件规约的集合[2].

定义 1.　软件 $Sftw=\langle p,D,R,Dcmt\rangle$ 是一个四元组，其中 $p\in P$，$Dcmt$ 是与该软件开发、运行、维护、使用和培训有关的图文材料，即软件文档[33].

定义 2.　软件错误(software error)是指导致产生含有缺陷的软件的人为行动[4].

定义 3.　软件缺陷(software defect)是存在于软件(文档、数据、程序)之中的那些不希望或不可接受的偏差，其结果是软件运行于某一特定条件时出现软件故障. 软件缺陷是存在于软件内部的、静态的一种形式[5].

定义 4.　对于软件 $Sftw=\langle p,D,R,Dcmt\rangle$ 和 $Sftw$ 的规约 $s\in S$，若 $\exists D_1\subseteq D$，使得 $D_1\xrightarrow{p}R\neq D_1\xrightarrow{s}R$，则称 $Sftw$ 中存在软件故障(software fault)[6].

非形式化地说，软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态，即无法满足人们所预期的正确规约. 出现软件故障时若无适当措施(如容错)加以及时处理，便产生软件失效. 软件故障是一种动态行为.

定义 5.　软件失效(software failure)是指软件运行时产生的一种不希望或不可接受的外部行为结果[5].

需要说明的是，故障、错误、缺陷、失效等概念的含义非常接近，当前有关这些概念的定义各文献不统一，容易混淆. 至今很多文献都没有严格区分这些概念. 例如，文献[3]和文献[4]均将故障、缺陷视为同义词. 本文通过给出以上定义区分这些概念.

2.2　软件故障产生原因

我们认为，软件发生故障的原因主要有两个方面：(1) 内部因素. 随着计算机控制对象复杂程度提高和软件功能增强，软件规模不断增大. 例如，Windows NT 操作系统的代码大约有 3200 万行. 并且软件内部状态、路径高度复杂. 参与软件开发各个阶段的人的思维和交流不可能完美无缺，在设计这样复杂的系统时难免犯错误，导致软件留下缺陷. (2) 外部因素. 通常软件是在开发环境中进行测试，在运行环境中使用. 对软件进行充分测试往往很困难，进行穷举测试，达到完全的路径覆盖、状态覆盖几乎不可能. 所以，即使是在开发软件产品时进行过大量测试，也不能完全发现和消除可能存在的缺陷；同时软件产品的开发环境与使用环境往往存在较大的差异，因此软件产品在使用时仍然可能发生故障.

3　软件故障模型

文献[8]认为软件故障模型是一些基本故障的组合，即软件故障模型应该是软件物理错误的抽象，并能反映其本质的一定程度的组合. 文献[11]认为故障模型是一类有某种相同属性的故障集合.

我们认为，软件故障模型是软件故障的外部宏观表现形式. 这里的外部宏观表现形式不仅包括软件运行时可观察到的状况征兆和可检测到的内部状态，还包括程序代码的结构、处理逻辑、复杂性以及软件文档、数据的有关静态信息等. 软件故障模型提供了软件故障的一种分类方法，将某些具有相同属性的软件故障划分在一起. 根据软件缺陷的产生阶段，可以将软件故障模型分成需求型故障、设计型故障、实现与编码型故障、集成型故障、测试型故障和配置管理型故障等.

(1) 需求型故障包括功能或者性能规定错误、遗漏某些功能，多余某些功能，功能不完善，为用户提供的信息有错误或不确切，对意外情况的处理错误，对用户的设计界面要求理解错误，用户接口不完善，需求分析文档有误，叙述不清晰，设计人员、测试人员没有在需求分析阶段就介入，需求变更后软件设计、代码未及时更改等.

(2) 设计型故障包括软件结构设计缺陷、数据结构或数据处理设计缺陷、中断和现场保护缺陷、初始化和复位缺陷、容错和防错设计缺陷、软件设计文档有误等. 软件结构设计缺陷包括程序控制流或控制顺序有误、处理过程有误等. 程序控制流方面的故障又包括分支型故障和循环型故障等. 分支型故障包括谓词的错误、判定变量赋值错误、谓词操作符不正确或少操作符、谓词的结构不正确(if 与 else 不匹配，分号位置不对)、switch 语句缺少默认情况的处理、switch 语句中的 case 处理后没有通过 break 退出等. 循环型故障主要包括永不循环故障和死循环故障. 数据结构或数据处理设计缺陷包括数据定义或数据结构定义有误、数据存取有误、数据存在冗

余、算法不完善或不当、对累积误差和舍入误差考虑不周、边界值处理不当、堆栈处理不当、忽视异常情况的处理、整数或浮点数的上溢/下溢时缺乏保护，数据的有效位数不够等.

(3)实现与编码型故障包括编码错或按键错、编程笔误、违背编码标准、语句使用不当，或者存在不可达代码、数据类型使用不当、数据类型不匹配、非法的数据类型转换、正负号处理不当、用拷贝方式复制的程序段修改不到位、数组下标越界、数组下标从0开始与从1开始相混淆、全局变量与局部变量相混淆、静态变量与动态变量相混淆、指针变量使用不当、坏的存储分配，或者内存泄漏、非法的算术运算、两个浮点数进行相等比较，或者在一定范围内的整数进行相等比较，函数调用不当等.

(4)集成型故障包括接口型故障、时序和时限型故障、死锁型故障、运行环境型故障等. 接口型故障包括接口协议不合理、各分系统软件的接口设计未考虑全系统的接口协议、接口设计不完善、状态定义不一致或指令格式定义不一致、未遵循已有接口协议、软件的内部接口、外部接口有误、软件各相关部分在时间配合、数据吞吐量等方面不协调，未考虑到设备或组合间的并发性等. 时序和时限型故障包括未处理好特殊、例外和异常情况，未处理好软件与硬件或硬件与硬件之间的协同，主机、从机时间关系不协调，实时软件的任务和中断的优先级定义不当，未考虑硬件性能的容差，存在无限循环的程序段，时间开销计算不准、不全，未留必要的余量等. 运行环境型故障包括对所用的可编程器件了解不够、对操作系统了解不够、在调用操作系统、恢复、诊断或者引用操作系统环境的过程中出错、对引进的操作系统中存在的问题不了解、对开发环境与运行环境的差异考虑不够、对运行环境中人的操作时间考虑不够等.

(5)测试型故障是指测试目的不明确、测试计划有误、测试设计有误、测试实施有误、测试文档有误、测试用例不充分、测试环境有误等造成的故障. 由于这些故障的存在，导致软件测试时不符合正确的软件规约，将本来是正确的运行结果判断为错误的，或者将本来是错误的运行结果判断为正确的. 前者造成对软件进行不必要的修改而带来缺陷[5]，后者造成遗漏本应能发现的缺陷.

(6)配置管理型故障包括配置管理不严格导致软件版本不一致、软件变更控制不严、变更时未考虑到对软件其余部分的影响、软件变更后未进行严格的回归测试、使用含有“多余物”的“实验版”等.

4 一种软件故障诊断过程框架

定义 6. 软件故障诊断是根据软件 $Sftw=\langle p,D,R,Dcmt\rangle$ 的静态表现形式和动态运行状态信息查找故障源，并确定相应纠错决策的一门技术.

4.1 软件故障诊断的含义

人在参与软件生存周期各个阶段工作时难免出现错误. 因此，从广义上说，软件故障诊断的目标包括软件需求分析、设计、编码、测试、使用、维护等软件生存周期各阶段所造成的缺陷.

软件故障诊断，“诊”在于进行客观的状态检测，包括采用各种测量、分析和诊别方法；“断”则需要确定软件故障特性、软件故障模型、软件故障部位以及说明软件故障产生的原因，并且提出相应的纠正措施和预防措施等.

4.2 一种软件故障诊断过程框架

本文提出一种软件故障诊断过程框架，它由故障检测、故障定位、故障排除和交付等组成，见图1.

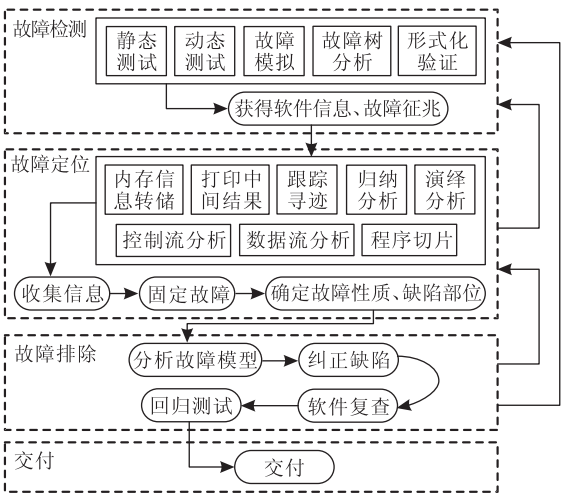

图1 一种软件故障诊断过程框架

(1)软件故障检测是根据系统提出的软件故障诊断请求，采取一定的检测方法，获取软件中的各种信息，获得可能出现软件故障的征兆，识别软件是否正常运行或存在故障，并为软件故障定位提供依据. 这些检测方法包括根据检查表和规则等进行静态测试、设计测试用例进行动态测试、建立软件故障树进行分析、采用变异测试方法进行软件故障模拟或者建立系统模型进行形式化验证等.

(2)软件故障定位是根据软件故障检测提供的能反映软件状况的征兆，与某故障模式进行比较，采

用内存信息转储、打印中间结果的方法，进行跟踪寻迹、归纳分析、演绎分析、控制流分析、数据流分析和程序切片等技术进一步收集软件的历史和使用信息，固定软件故障，从而诊断软件故障的性质和程度、确定缺陷产生原因或发生部位，为纠正缺陷、排除软件故障做好准备.

(3) 软件故障排除是当诊断出软件中存在缺陷，就其原因、部位和危险程度进行分析，确定纠正缺陷、排除软件故障的措施，包括针对软件缺陷产生的具体原因修改程序代码、数据或文档等，并进行软件复查和回归测试，确认软件故障排除，软件功能正常.

(4) 交付是当对软件故障进行检测、定位，采取措施排除软件故障后，即可将软件交付使用.

从图 1 可以看出，软件故障诊断是一个信息反馈的过程，可根据需要进行迭代.

该框架的特点是：比较系统、全面地规范了软件故障诊断过程中的故障检测、故障定位、故障排除和交付活动，并提供了一些有效的检测、定位、排除软件故障的方法，对软件故障诊断过程进行系统的指导，以避免盲目地进行故障诊断.

一般来说，在工程应用中进行软件故障诊断的前提是：系统故障经分析、检测确认不是由硬件故障引起，或重点怀疑是由软件故障引起. 盲目地进行软件故障诊断将影响系统故障诊断效率. 系统出现复杂故障时，也可结合硬件检测，同时进行软件静态检测，这样可以提高系统故障诊断速度.

4.3　软件故障检测

软件故障检测是软件故障定位和排除的前提.

4.3.1　软件故障检测方法

软件故障检测的方法包括软件测试、软件故障树分析、软件故障模拟和形式化检测方法等.

(1) 软件测试[2,14,18,20]

现有软件测试技术通常分为静态测试和动态测试. 静态测试是不执行程序代码而寻找程序代码中可能存在的缺陷或评估程序代码的过程. 静态测试主要包括由人工进行的桌面检查、代码审查、代码走查等. 动态测试通过在抽样测试数据上运行程序来检验程序的动态行为和运行结果以发现缺陷. 动态测试分为基于规约的测试(又称黑盒测试或功能测试)、基于程序的测试(又称白盒测试或结构测试)以及程序与规约相结合的测试.

(2) 软件故障树分析[27]

软件故障树分析是一种用于分析软件故障产生原因的技术. 软件的故障树分析法在原理、所用的标志符、建立步骤等方面与硬件故障树分析法完全相同. 软件故障树分析的这些特点，使得硬件故障树与软件故障树可以在接口处相互联接，从而使整个系统都可以用故障树进行分析.

(3) 软件故障模拟[2,14]

变异测试技术能够系统地模拟软件故障，并构造有效的测试数据将这些故障检测出来. 其基本原理是：使用变异算子每次对被测程序作一处微小的合乎语法的变动(例如将关系运算符“>”用“<”替换)，产生大量的新程序，每个新程序称为一个变异体；然后根据已有的测试数据，运行变异体，比较变异体和原程序的运行结果：如果两者不同，就称该测试数据将该变异体杀死. 杀死变异体的过程一直执行到杀死所有变异体或变异充分度已经达到预期的要求. 变异充分度是已杀死的变异体数目与所有已产生的非等价变异体数目的比值.

在变异测试过程中，变异算子模拟软件的各种缺陷，作用到源程序上得到变异体. 这种有缺陷的软件运行将有可能导致某些软件故障. 变异测试工具集 Mothra 针对 Fortran 语言定义关系运算符替换等 22 种变异算子. 当前人们还研究对模块接口、面向对象软件和软件合约[34]进行变异测试.

(4) 形式化检测方法

形式化方法是关于在计算系统的开发中进行严格推理的理论、技术和工具，可用于检测从高层规范至最终实现的过程中的软件缺陷. 形式化方法主要包括形式化规约技术和形式化验证技术[35].

形式化规约技术使用具有严格数学定义语法和语义的语言刻画软件系统，以尽早发现需求和设计中的错误. 顺序系统的形式化规约技术侧重于描述状态空间，其主要思想是利用集合、关系和函数等离散结构表达系统的状态，用前置断言、后置断言表达状态的迁移，例如 Z、VDM；并发系统的形式化规约技术侧重于描述系统并发特性，其主要思想是用序列、树、偏序等表达系统的行为，例如 CSP、CCS、Statecharts、时序逻辑；RAISE 语言和方法综合这两种思路.

形式化验证技术是在形式化规约技术的基础上建立软件系统及其性质的关系，即分析系统是否具有所期望性质的过程. 模型检验是一种重要的形式化验证方法，通过搜索待验证软件系统模型的有穷状态空间来检验系统的行为是否具备预期性质[15]. 在模型检验中，系统用有穷状态模型建模；其要验证

的系统性质通常是时序逻辑或模态逻辑公式,也可以用自动机语言描述;通过有效的自动搜索检验有穷状态模型是否满足性质,如果不满足,它还能给出使性质公式为假的系统行为轨迹.

为了更快(有效率)地发现软件缺陷,一方面应提高软件测试、软件故障树分析、软件故障模拟和形式化检测方法等软件故障检测方法的效率,另一方面可以进行查错设计,即在设计中赋予程序某些特殊的功能,使程序在运行中自动查找存在的缺陷[27].

4.3.2 软件故障检测过程

根据所采用的软件故障检测方法的不同,软件故障检测的过程有所不同.

可以建立检查表或者制定规则进行静态测试,对照检查表或所制定的规则,获取程序代码的结构、处理逻辑、复杂性以及软件文档、数据的有关信息,确定软件是否存在缺陷.

也可以设计测试用例进行动态测试,在软件运行的过程中,获取其动态信息,包括中间状态、所经过的控制流、所出现的数据流,将实际运行结果与预期结果进行比较,确定软件是否存在故障,并获取故障征兆,为软件故障定位提供依据.

当对软件异常现象进行分析,确认是故障,而且故障性质复杂、有很多潜在因素影响,又难以使用一般诊断技术解决问题时,可采用软件故障树分析方法.先确定软件故障树顶事件,从顶事件开始,按照建树步骤和方法,建立完整的故障树.然后进行故障树分析.定性分析故障树最小割集,找出所有故障模式,在数据充分、真实的情况下,定量分析出各故障模式的重要度,为针对各种故障模式逐一进行故障定位提供依据.

或者采用变异测试方法进行软件故障模拟.利用变异算子产生变异体,生成能够杀死变异体的测试数据,从而获得软件缺陷引入的位置以及能够找出该缺陷的测试数据.

还可以采用形式化方法建立系统模型、给出性质规范,在工具软件的支持下,采用模型检验等方法进行形式化验证,判断系统是否满足性质规范.若系统满足性质规范,则软件不存在有关该性质规范的故障;否则,软件存在故障,获取系统违反性质规范的反例路径.

4.4 软件故障定位

排除软件故障的前提是准确地隔离故障,即对软件故障定位.应从分析故障征兆着手,查找引起软件故障的缺陷的位置.比如在测试中发现文件记录中丢失最后一个字符,这是一个故障征兆.需要利用各种信息和经验,判断问题发生的真正原因,找到有缺陷的程序单元或语句.

4.4.1 软件故障定位方法

在软件故障定位的过程中,经常采用以下方法.

(1)内存信息转储[14]

内存信息转储是在执行测试中发现问题以后,把所有寄存器和内存有关部分的内容记录、打印出来,设法保留有关的现场信息,供分析研究使用.使用这种方法可以得到出现故障的关键信息.当已经对程序中某个故障假设有了进一步认识,用这种方法取得相关信息,做出推断,常常很有效.但是进行内存信息转储需要占用一些主机时间、较多的 I/O 时间以及分析时间,辨认这些信息,找出和源程序的对应地址也较费事.因此,必须有目标、有限制地使用这种方法.

(2)打印中间结果[14,27]

为取得关键变量的动态值,在程序中设置若干打印语句,打印这些变量的名称及当时的数值,进而检查这些输出的中间结果,判断哪一点开始发生故障.这种方法可以检验在某事件以后某个变量是否按预计的要求发生变化.而且多个打印语句可以给出变量的动态特性.

(3)跟踪寻迹[14,27]

在软件故障定位中,跟踪是指跟踪程序的运行.可以在屏幕上显示程序的执行过程及状态,或将指定的程序段在运行中的状态变化列表打印输出,从而寻找程序执行过程中的故障痕迹,进行故障定位.最简单的跟踪打印出所有变量数值的变化和所有控制流的变化.选择式跟踪在源程序的指定区段跟踪指定的变量和控制流.跟踪寻迹方法可以分为回溯法、向前追踪法和二分查找逼近法等几种方法.回溯法也称向后追踪.考察错误征兆,从它在程序显露的位置起沿着程序的控制流向后(反方向)追踪,直到征兆消失处为止.再仔细分析邻接的程序段,进而找出错误的原因.向前追踪法则与回溯法的检查方向相反.如果已经知道程序中某些关键点上变量应取的正确值,可以采用二分查找的办法.例如,在程序的中间部位引入一组输入,并检验其输出结果,如果输出结果是正确的,那么故障发生在程序的前半部分,否则在程序的后半部分.多次重复这个过程,直至逼近到故障的准确位置.

当采用模型检验方法获得系统违反性质规范的反例路径后,可根据该反例路径采用跟踪寻迹法定

位软件故障.

(4)归纳法[14,18]

归纳法研究与出错相关的信息,找出特征,得到"原因假设",然后确认或否认这个假设.归纳法定位软件故障的步骤为:① 收集有用信息.列举已知失败的测试用例情况和成功的测试用例情况.弄清哪些是观察到的故障征兆,何时出现故障,什么情况下出现故障等.② 确定故障类型.检查收集到的信息,注意区别失败的测试用例和成功的测试用例间的差别,找到软件故障特征,以确定软件故障类型.③ 构想出一个或若干假设原因.根据观察到的关系和故障类型,提出一个或多个假设.如果假设的原因并不是显而易见的,再来考察已有信息,收集更多的其它信息,也许还需要运行更多的测试用例.如果有几个假设,则将其按成立的可能性排列,最大可能的假设列在前面.④ 审查假设原因,看其能否成立.再次检查有关信息,以便确定假设能否解释观察到的问题的各种表现.

(5)演绎法[14,18]

演绎法正好和归纳法的过程相反,首先列举一些可能的原因或假设,然后再进行逐个分析,排除那些不能确立的原因和假设,直到仅剩下一个被证实.演绎法定位软件故障的步骤为:① 枚举若干可能的原因和假设.首先列出所有可能的故障原因,这些原因并不要求有充分的理由,只是提供一些分析的线索.② 利用掌握的资料排除一些原因.仔细地分析已掌握的资料,找出一些矛盾,力图排除那些不能成立的原因.但如果所有的原因都被排除,就应另作推测.如果保留下来多个原因,就应把最大可能的原因列为优先考虑的对象.③精心研究保留的假设.这时保留的可能原因也许是对的,但对于确认故障可能还不够充分.要充分利用已知的线索作精心的研究,估计到特殊情况.

(6)控制流分析[14]

对于程序结构一般有以下 4 点基本要求,即写出的程序不应包含:① 转向并不存在的标号;② 没有用的语句标号;③ 从程序入口进入后无法达到的语句;④ 不能达到停机语句的语句.前两种情况容易发现.通过构造控制流图,计算从入口结点可达的所有结点,可以发现第 3 种情况的软件故障.采用类似的方法可以发现第 4 种情况的软件故障.

(7)数据流分析

数据流分析方法可用来查找引用未定义变量等软件故障以及查找对以前未曾使用的变量再次赋值等数据流异常的情况.这些故障是诸如错拼名字、名字混淆或丢失语句等程序缺陷的表现形式.

首先构造可达定义表[14],然后按下述方法找出对未定义变量的引用:对每个结点 i,依次考虑 i 引用的每个变量,如果对任何这样的变量 v,并没有 v 的定义达到 i,那么程序中含有缺陷.可以用下面的方法找出未曾使用的定义:对于每个 $DEF(v,i)$(表示结点 i 定义变量 v),依次考虑由 $DEF(v,i)$ 达到的每个结点 j,若没有引用变量 v 的相应语句,则程序中含有缺陷.

(8)程序切片[36]

对程序进行切片,使所得到的程序代码片断仍能反映原程序的部分特征,以缩小软件故障定位时的检查范围.进行程序切片时依据切片准则来裁剪程序.切片准则是 $C=\langle q,v\rangle$,其中 q 是程序中的一条语句,v 是一个程序变量.当程序的下一执行语句为 q 时,删去源程序中不会影响变量 v 之值的语句,所得到的即是关于 $\langle q,v\rangle$ 的程序切片.程序内部主要由控制流和数据流组成,并构成语句之间的控制依赖关系和数据依赖关系.程序切片主要依据语句之间的控制依赖关系和数据依赖关系计算得到.

4.4.2　软件故障定位过程

软件故障定位的过程一般包括以下步骤:

(1)透彻地理解所面临的问题.为实现软件故障定位,必须深入地研究程序及其执行结果,分析它在什么情况下得到正确结果,在什么情况下得到错误结果.由此提出关于软件故障性质及其原因的若干假设.可以采用 4W 表格来组织和分析有关的故障信息[18],如表 1 所示.表中的 What 栏用于记录基本征兆,Where 栏记录故障征兆的位置,When 栏记录故障征兆发生的时间,To what extent 栏记录征兆波及的范围,Is 和 Is not 栏是 4W 表格中的关键内容,通过这两栏内容的对比,有助于形成关于故障原因和部位的假设.

表 1　4W 表格

?	Is	Is not
What		
Where		
When		
To what extent		

(2)固定软件故障.若能使某一软件故障稳定地发生,则能方便地确诊它,否则对其进行诊断非常困难.软件故障不定期发生通常是由于未对变量进行初始化、使用悬挂指针[37]或者与软件运行的时间

长短有关.固定一个软件故障通常需要一个以上的测试用例来产生故障.它包括将测试用例减至最少而仍能产生故障的情况.为简化测试用例,可以采用以下方法:①通过重复实验收集数据;②建立假设以解释尽可能多的相关数据;③设计实验以便证实或否定假设;④证实或否定假设;⑤按要求重复以上步骤.

(3)制定查错计划.这一步要求排错人员提出错误原因、性质、部分的假设,根据这些假设制订查找错误的计划.

(4)执行查错计划.实施计划以验证各种假设的正确性,对计划执行过程的每一步都要进行检查.

(5)评估查错结果.经过执行计划,如果故障的位置已经查明,在动手改正之前还要再作认真的检查,分析它们是否确系测试中出现的故障征兆的真正原因,验证故障征兆涉及的部分有无遗漏.如果故障的位置尚未查明,返回第(3)步修改查错计划.

4.5 软件故障排除

软件故障排除属于软件维护的范畴.软件维护是在软件产品交付之后对其进行修改,以纠正故障,改进性能和其它属性,或使产品适应改变了的环境活动.

4.5.1 软件故障排除方法

排除软件故障的方法为:根据软件故障现象和引起软件故障的缺陷位置,分析软件故障模型,确定是由需求规格说明还是软件设计、软件编码、软件集成、软件测试或者配置管理中的缺陷所引起的,对相应的缺陷进行修改、纠正;分析故障影响范围,对受影响的部分进行相应的修改.

(1)文档齐全情况下的软件故障排除.需求型故障主要是由需求分析过程中的错误所造成的,应更改需求规格说明,然后继续执行软件工程后续步骤进行所需要的设计、编码、测试等.

设计型故障主要是由软件设计过程中的错误所造成的,应更改软件设计,然后继续执行后续步骤.

实现与编码型故障主要由编码过程中的错误造成,应更改程序代码,并继续进行所需要的测试等工作.

集成型故障主要由软件设计过程中的错误引起,应更改软件设计,然后继续执行后续步骤.为排除集成型故障中的接口型故障,主要应抓住接口协议的拟订与签署、接口分析、接口设计和接口测试.为排除集成型故障中的死锁型故障,应更改设计,建立全局资源分配视图,对资源申请、分配和释放进行集中统一管理.

测试型故障主要由测试过程中的错误引起,应更改测试计划,设计相应测试用例,进行所需要的测试.

配置管理型故障主要由配置管理过程中的错误引起,应加强配置管理,保持软件版本的一致性;严格控制软件变更,变更时充分考虑对软件其余部分的影响,软件变更后进行严格的回归测试.

(2)缺少文档情况下的软件故障排除.当缺少软件文档时,可采用逆向工程[33]等技术手段从程序中抽取数据结构、体系结构和程序设计信息,排除软件故障.

(3)系统重构.在实时执行任务的过程中如果出现严重故障,没有充足的时间排除故障,此时对于有软件容错设计的系统常用的方法是进行系统重构,启动软件容错措施,排除软件故障[27].

4.5.2 软件故障排除过程

排除软件故障的过程一般包括以下步骤:(1)维护的申请与审批.(2)制定维护计划.(3)执行维护计划,改正错误.(4)软件复查.验证所作的修改是否正确,重新确认整个软件.软件复查工作包括进行影响域分析,即分析所更动部分是否会对其余部分造成不良影响.(5)回归测试.验证软件更动的正确性和对原有功能、质量特性的不损害性.(6)评审与验收.

5 案例研究

为验证以上软件故障诊断过程框架的有效性,将其应用于实际的软件故障诊断中.

5.1 案例 1.某控制台工控机显示信号异常问题

在某控制台与装置故检系统进行联调过程中,当控制台操作手按下“逃逸”按键,向故检系统发送“逃逸”指令(简称“逃逸”),装置逃逸指令控制器收到该指令后给控制台回传“有线逃逸指令收到”信号(简称“有线逃逸”)时,发现控制台工控机甲机和乙机显示信号异常.在多次联调中,两台工控机有时均显示“有线逃逸”,有时均不显示,有时仅某一台显示,正常情况是均应显示.通过检测排除硬件和通信链路问题.因此怀疑故障由工控机软件中与处理“有线逃逸”和“逃逸”有关的控制检测线程引起.应用以上框架进行软件故障诊断.

(1)故障检测.分析该线程,它用 4 字节保存输入端口信息,其中两位分别记录“有线逃逸”和“逃

逸”.采用静态测试中的代码审查和代码走查方法,结果均未发现软件缺陷.进行动态测试,先采用白盒测试方法,设计测试用例,进行语句覆盖、分支覆盖、路径覆盖,结果故障不复现.然后采用黑盒测试中的等价类划分方法,根据“有线逃逸”和“逃逸”设计等价类.当设计测试用例,将“逃逸”信号位置“1”,模拟人工操作控制台的“逃逸”按键,并且将“有线逃逸”信号位置“1”,模拟故检系统正确返回“有线逃逸”,驱动该线程运行,结果故障能够复现,表明该线程处理“有线逃逸”和“逃逸”的流程有误.

(2)故障定位.

(a)固定故障.采用打印中间结果的方法,当向故检系统发送“逃逸”后,输出该线程中保存输入端口信息的变量的值,发现“有线逃逸”信号位已被置“1”,说明故检系统确已正确返回“有线逃逸”.

用人工按下“逃逸”按键后人工操作故检等效器上的“有线逃逸”开关按钮故障无法复现,它与故检系统联调时人工按下“逃逸”按键后由故检系统自动返回“有线逃逸”故障经常能够复现的区别,主要是两者从发出“逃逸”到“有线逃逸”返回的时间间隔不同,前者为数百毫秒以上,后者为数十毫秒.因此,怀疑故障与巡检周期的时间长短有关.该线程设计的巡检周期为30ms.对该线程程序代码中的休眠语句的时间值(10ms)进行修改,当将休眠值改为很小的值,例如0ms时,与故检系统联调故障也很难复现;而当将休眠值改为很大的值,例如5000 ms时,与故检系统联调故障能够稳定地复现,甚至人工按下“逃逸”按键后人工操作故检等效器上的“有线逃逸”开关按钮故障也能够稳定地复现.

(b)故障定位.该线程设计时没有考虑到在同一巡检周期内可能同时出现“有线逃逸”和“逃逸”,当出现这种情况时,只处理优先级高的“逃逸”,未处理“有线逃逸”,就直接转到下一巡检周期开头.而在下一巡检周期内,由于“有线逃逸”信号位已被置“1”,在没有其它信号到来的情况下,保存输入端口信息的变量的值不发生变化,程序休眠10ms后,再次回到下一巡检周期开头,使工控机一直未能显示该信息.因此,引起该故障的软件缺陷定位在:处理“逃逸”后,未处理“有线逃逸”,就直接转到下一巡检周期开头.这是由于软件设计不当所引起的,属于设计型故障.

(3)故障排除.为排除该故障,修改该线程的设计,优先处理“逃逸”,然后由原来的直接转到下一巡检周期开头,修改为转向处理输入信息,在此过程中处理“有线逃逸”,使工控机显示该信息,最后转到下一巡检周期开头.根据修改后的设计修改程序代码,并进行回归测试.利用修改后的程序进行联调,结果表明缺陷已被排除,故障不再发生.并且通过多次联调和实际应用进行验证和确认.

5.2　案例2.某装置误用变量名称导致装置失控

某系统进行试验时点火正常.在制导阶段装置失控,若干秒后自毁.通过检测排除硬件和通信链路问题.应用以上框架进行软件故障诊断.

(1)故障检测.分析试验所采集的数据结果,发现测量方位角偏差的滤波结果在个别时间段上不一致,在某时刻滤波结果与实测值的符号相反.为此,采用动态测试的方法,编写专门的滤波验证程序,将实测的方位角偏差数据作为输入量进行滤波,获得滤波处理的正确结果;然后将主控程序滤波实际结果与正确结果进行比较,结果发现两者不一致,表明主控程序滤波处理有问题.

(2)故障定位.

(a)固定故障.对主控程序进行修改,使之能够将角偏差字输入,改用本次试验采集到的相应数据,模拟监测装置的数据输入过程,并让修改后的主控程序在实时方式下运行,复现了故障现象.

(b)故障定位.利用上述修改后的主控程序,采取跟踪寻迹方法向后追踪与方位角偏差有关的变量,找出有缺陷的语句.引起该故障的软件缺陷位于以下程序段中:

if ($mazd\_shi[0] < -0.020$) $mazd\_shi[0] = -0.020$;
if ($mazd\_shi[0] > 0.020$) $mazd\_shi[0] = 0.020$;
if ($meld\_shi[0] < -0.020$) $mazd\_shi[0] = -0.020$;
if ($meld\_shi[0] > 0.020$) $mazd\_shi[0] = 0.020$.

语句中 $mazd\_shi[0]$ 和 $meld\_shi[0]$ 分别为输入的测量的方位角偏差和高低角偏差.后两条语句中变量名 $mazd\_shi[0]$ 是错误的,应为 $meld\_shi[0]$.本例中的软件故障是由于用拷贝方式复制的程序段修改不到位所引起,属于实现与编码型故障.

(3)故障排除.排除该故障的措施是:将有缺陷的程序段中后两条语句中变量名由 $mazd\_shi[0]$ 改为 $meld\_shi[0]$.对改正后的程序进行测试,并且再次利用这次试验的采集数据和故意设置的一些异常数据进行验证,结果表明缺陷已被排除,故障不再发生.并进一步通过多次的模拟试验进行验证和确认.

6　结束语

软件在国民经济和社会生活中发挥着重要作

用.软件出现故障会给人们造成很大的危害.研究软件故障诊断意义重大.本文分析软件失效机理和软件故障产生原因,讨论软件故障模型,提出一种软件故障诊断过程框架,将软件故障检测、故障定位、故障排除等活动集成在一起,为系统地进行软件故障诊断提供指导.并且将该框架应用于实际的软件故障诊断,验证了其有效性.

软件故障诊断研究领域非常广泛,包括软件缺陷和软件故障预防等,由于不是本文的研究重点,故没有将软件缺陷和软件故障预防等纳入本文所提出的框架中.实际上,为了更有效地发现问题,以避免严重后果的发生,一方面应预防软件缺陷的引入,另一方面应预防软件故障的发生.前者的主要方法包括加强软件开发过程管理(如实施 CMM 五级中的缺陷预防关键过程域等)[28-29]和进行避错设计[27]等.后者的主要方法包括软件容错设计[5,27]等.我们相信,随着软件缺陷和软件故障预防等研究的不断深入,必将推动软件故障诊断研究的不断向前发展.

在今后的工作中,我们将分析、研究已有的硬件系统故障诊断案例,深入研究软件故障的特点和各种硬件系统故障诊断方法,进一步完善所提出的软件故障诊断过程框架,并且继续在软件故障诊断实践中进行检验.

致 谢 审稿人对本文提出了宝贵的修改意见,作者在此表示衷心感谢!

参 考 文 献

[1] Chen Bing-Zhong, Wang Peng. The trend toward software of manned spaceflight engineering and its heuristics. Manned Spaceflight, 2006, 12(4): 1-4(in Chinese)
(陈炳忠,王朋. 载人航天工程软件化趋势及其启示. 载人航天, 2006, 12(4): 1-4)

[2] Zhu Hong, Jin Ling-Zi. Quality Assurance and Testing of Software. Beijing: Science Press, 1997(in Chinese)
(朱鸿,金凌紫. 软件质量保障与测试. 北京:科学出版社, 1997)

[3] IEEE. IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology. December, 1990

[4] National Standard of the People's Republic of China. GB/T 11457-1995, Standard Glossary of Software Engineering Terminology, 1995(in Chinese)
(中华人民共和国国家标准. GB/T 11457-1995,软件工程术语, 1995)

[5] Cai Kai-Yuan. The Foundations of Software Reliability Engineering. Beijing: Tsinghua University Press, 1995(in Chinese)
(蔡开元. 软件可靠性工程基础. 北京:清华大学出版社, 1995)

[6] Offutt A J, Hayes J H. A semantic model of program faults//Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'96). San Diego, CA, USA, 1996: 195-200

[7] Munson J C, Nikorab A P, Sherif J S. Software faults: A quantifiable definition. Advances in Engineering Software, 2006, 37(5): 327-333

[8] Zhu Rong, Xu Shi-Yi. The building of fault model in software test. Computer Engineering and Applications, 2003, 39(17): 69-71, 91(in Chinese)
(朱荣,徐拾义. 软件测试中故障模型的建立. 计算机工程与应用, 2003, 39(17): 69-71, 91)

[9] Gong Yun-Zhan. An introduction of software technique oriented software faults test model. Journal of Academy of Armored Force Engineering, 2004, 18(1): 21-25(in Chinese)
(宫云战. 一种面向故障的软件测试新方法. 装甲兵工程学院学报, 2004, 18(1): 21-25)

[10] Baekken J S, Alexander R T. A candidate fault model for AspectJ Pointcuts//Proceedings of the 17th International Symposium on Software Reliability Engineering. Raleigh, North Carolina, USA, 2006: 169-178

[11] Xu Shi-Yi. Design and Analysis of Dependable Computing Systems. Beijing: Tsinghua University Press, 2006(in Chinese)
(徐拾义. 可信计算系统设计与分析. 北京:清华大学出版社, 2006)

[12] Lee I, Iyer R K. Diagnosing rediscovered software problems using symptoms. IEEE Transactions on Software Engineering, 2000, 26(2): 113-127

[13] Baresi L, Young M. Test oracles. Department of Computer and Information Science, University of Oregon: Technical Report CIS-TR01-02, 2001

[14] Zheng Ren-Jie. Computer Software Testing Technologies. Beijing: Tsinghua University Press, 1992(in Chinese)
(郑人杰. 计算机软件测试技术. 北京:清华大学出版社, 1992)

[15] Bérard B, Bidoit M, Finkel A, Laroussinie F, Petit A, Petrucci L, Schnoebelen P, McKenzie P. Systems and Software Verification-Model-Checking Techniques and Tools. Berlin: Springer, 2001

[16] Almeida J R, Camargo J B, Basseto B A, Paz S M. Best practices in code inspection for safety-critical software. IEEE Software, 2003, 23(3): 56-63

[17] Shan J H, Wang J, Qi Z C, Wu J P. Improved method to generate path-wise test data. Journal of Computer Science and Technology, 2003, 18(2): 235-240

[18] Myers G J. The Art of Software Testing. Second Edition. Hoboken, New Jersey: John Wiley & Sons, 2004

[19] Deeprasertkul P, Bhattarakosol P, O'Brien F. Automatic detection and correction of programming faults for software applications. The Journal of Systems and Software, 2005, 78(2): 101-110

[20] Shan Jin-Hui, Jiang Ying, Sun Ping. Research progress in software testing. Acta Scientiarum Naturalium Universitatis Pekinensis, 2005, 41(1): 134-145(in Chinese)
(单锦辉, 姜瑛, 孙萍. 软件测试研究进展. 北京大学学报(自然科学版), 2005, 41(1): 134-145)

[21] Cleve H, Zeller A. Locating causes of program failures// Proceedings of the 27th International Conference on Software Engineering(ICSE 2005). St. Louis, MO, USA, 2005: 342-351

[22] Zeller A, Hildebrandt R. Simplifying and isolating failure-inducing input. IEEE Transactions on Software Engineering, 2002, 28(2): 183-200

[23] Gupta N, He H, Zhang X, Gupta R. Locating faulty code using failure-inducing chops//Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), Long Beach, CA, USA, 2005: 263-272

[24] Xu Bao-Wen, Nie Chang-Hai, Shi Liang, Chen Huo-Wang. A software failure debugging method based on combinatorial design approach for testing. Chinese Journal of Computers, 2006, 29(1): 132-138(in Chinese)
(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): 132-138)

[25] Zhang X, Gupta N, Gupta R. Locating faults through automated predicate switching//Proceedings of the 28th International Conference on Software Engineering (ICSE 2006). Shanghai, China, 2006: 272-281

[26] Baudry B, Fleurey F, Traon Y L. Improving test suites for efficient fault localization//Proceedings of the 28th International Conference on Software Engineering (ICSE 2006). Shanghai, China, 2006: 82-91

[27] Huang Xi-Zi. Software Reliability, Safety, and Quality Assurance. Beijing: Publishing House of Electronics Industry, 2002(in Chinese)
(黄锡滋. 软件可靠性、安全性与质量保证. 北京: 电子工业出版社, 2002)

[28] Humphrey W. Managing the Software Process. MA: Addison-Wesley, 1989

[29] Chang C P, Chu C P. Defect prevention in software processes: An action-based approach. The Journal of Systems and Software, 2007, 80(4): 559-570

[30] Gan Mao-Zhi, Kang Jian-She, Gao Qi. Military Equipment Maintenance Engineering. 2nd Edition. Beijing: National Defense Industry Press, 2005(in Chinese)
(甘茂治, 康建设, 高崎. 军用装备维修工程学. 第 2 版. 北京: 国防工业出版社, 2005)

[31] Zhou Bin, Sun Jun, Li Jin-Xia. Status quo of fault diagnosing research and its application to manned spaceflight. Manned Spaceflight, 2006, 12(5): 47-51,54(in Chinese)
(周彬, 孙军, 李晋霞. 故障诊断技术研究现状及其在载人航天中的应用. 载人航天, 2006, 12(5): 47-51, 54)

[32] Xu Ke-Jun, Zheng Yong-Huang, Sun Chuan-Fei. The Astronaut Life Saving System in Time of Emergency in the Launching Phase. Beijing: National Defense Industry Press, 2006(in Chinese)
(徐克俊, 郑永煌, 孙传飞. 待发段航天员地面应急救生系统. 北京: 国防工业出版社, 2006)

[33] Qi Zhi-Chang, Tan Qing-Ping, Ning Hong. Software Engineering. 2nd Edition. Beijing: Higher Education Press, 2004 (in Chinese)
(齐治昌, 谭庆平, 宁洪. 软件工程. 第 2 版. 北京: 高等教育出版社, 2004)

[34] Jiang Y, Hou S-S, Shan J-H, Zhang L, Xie B. Contract-based mutation for testing components//Proceedings of the 21st International Conference on Software Maintenance (ICSM 2005). Budapest, Hungary, 2005: 483-492

[35] Chen Huo-Wang, Wang Ji, Dong Wei. High confidence software engineering technologies. Acta Electronica Sinica, 2003, 31(12A): 1933-1938(in Chinese)
(陈火旺, 王戟, 董威. 高可信软件工程技术. 电子学报, 2003, 31(12A): 1933-1938)

[36] Weiser M. Program slicing. IEEE Transactions on Software Engineering, 1984, SE-10(4): 352-357

[37] Steve McConnell. Tian A translator. Xiong K Y proofreader. Code Complete. Beijing: Xue Yuan Press, 1993(in Chinese)
(Steve McConnell 著. 天奥译. 熊可宜校. 代码大全. 北京: 学苑出版社, 1993)

SHAN Jin-Hui, born in 1970, Ph.D., senior engineer. His research interests include software testing, software quality management, and software component technology.

XU Ke-Jun, born in 1943, professor. His research interests focus on fault diagnosis in spaceflight launching.

WANG Ji, born in 1969, Ph.D., professor, Ph.D. supervisor. His research interests include high confidence software and systems, software engineering and distributed computing.

Background

This work was completed under a grant from the National Science Foundation of China, Research on Software Fault Detecting Methods Based on Fault Simulation (No. 60970022).

Software plays an important part in our everyday life. And the demand one makes on software is becoming even

higher.

Every phrase in software life cycle involves people's participation. Since the thoughts of people and communication between people are far from perfect, the mistake made by people is inevitable. With the increasing of the complexity degree of objects controlled by software and the strengthening of the software functions, the size of software is becoming larger, and more bugs are left in the software. As a result, when the software with bugs gets running, software fault occurs for sure. Software fault may bring about severe disaster. For instance, one software fault led to the launching failure of Ariane 5 rocket of European Space Agency in 1996. Therefore it is of important significance to investigate software faults diagnosis.

Some researches on software faults diagnosis have been done at present, which includes software faults concepts, software models, software faults description methods, software faults detection methods, software faults localization methods, software faults prevention methods, etc. However, the majority of these researches focus on single pursuit in software faults diagnosis, and there is hardly any research on how to integrate various software diagnosis pursuits effectively.

It has made rapid progress to research on the faults diagnosis of hardware systems. And faults diagnosis approaches based on analytic models, signal processing, and artificial intelligence have been proposed. The faults diagnosis process of hardware systems is made up of faults detection, fault isolation, faults treatment, and faults prevention.

This paper draws on the faults diagnosis of hardware systems to do research on that of software. To begin with, this paper analyzes the mechanism of software failure and the cause of software faults, discusses software fault models. Then a kind of integrated software fault diagnosing framework is proposed, which is composed of fault detection, fault localization, fault removal, and software release. Furthermore various kinds of methods and corresponding process of fault detection, fault localization, and fault removal are studied. Finally the proposed framework is applied to real software fault diagnosis to validate its effectiveness.