软件故障诊断初探

单锦辉　徐克俊
（中国酒泉卫星发射中心）

摘　要　介绍国内外软件故障诊断技术的研究现状，并比较软件与硬件的差异，分析软件故障产生的原因和软件失效机理，各种硬件系统故障诊断技术，并对软件故障诊断进行了初步的探讨。

关键词　软件　故障　故障诊断　硬件

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。人们越来越离不开软件。软件一旦出现故障，轻则给人们工作、生活带来不便，重则有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[1]。研究如何诊断软件故障具有重要意义。

目前国内外开展了与软件故障诊断有关的一些研究工作。文献[2]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[3]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[4]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[5]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。

文献[6]认为软件故障模型是一些基本故障的组合，或者说，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。已有的研究工作建立了以下软件故障模型[7]：①根据错误发生阶段进行分类，有需求分析错误、概要设计错误、详细设计错误、编码错误、测试错误等；②根据故障引起后果进行分类，有较小错误、中等错误、较严重错误、严重错误、非常严重错误、最严重错误；③根据错误性质进行分类，有需求错误、功能和性能错误、结构错误、数据错误、实现和编码错误、集成错误、系统结构错误、测试错误；④根据错误类型进行分类，有文档错误、语法错误、联编打包错误、赋值错误、接口错误、检查错误、数据错误、函数错误、系统错误、环境错误等。

文献[8]针对 C 语言程序建立以下几种故障模型：坏的存储分配、内存泄漏故障、初始化变量错误、指针和越界指针的引用错误、数组越界错误、非法的算术运算错误（除数为 0、负数开方、对数幂变量为 0 或负数等）、整数或浮点数的上溢/下溢错误、非法类型转换错误（如 long to short, float to integer）、不可达代码错误等。

综上所述，目前尚未明确提出软件故障诊断的概念，也没有系统地开展软件故障诊断技术的研究，甚至软件故障的概念都比较模糊，与软件错误、缺陷、失效等概念相混淆，至今很多文献都没有严格区分这些概念。

当前硬件系统故障诊断研究和实践取得了较大的进展，并且有大量的成功案例。我们认为，软件故障诊断可以借鉴硬件系统故障诊断的思路。本文在分析现有硬件系统故障诊断技术的基础上，对软件故障诊断进行初步探讨。

2　软件与硬件的区别

计算机软件（以下简称软件）是与计算机系统操作有关的程序、规程、规则及任何与之有关的文档及数据，它由可执行部分和不可执行部分组成。其中可执行部分包括程序和数据。程序是按事先设计的功能和性能需求执行的指令序列，数据则是程序能正常操纵信息的数据结构。不可执行部分为与软件开

发、运行、维护、使用和培训有关的图文材料，即软件文档。

软件是人脑逻辑思维的产物，有着自身的特点，在开发、生产、维护、报废及复杂性、发生故障的原因等方面与硬件有着很大的差别[9,10]。

(1)软件和硬件在开发、生产、维护、报废等方面的差异。

① 开发的差异。硬件设计虽然千差万别，但多数可用标准零部件、标准电路搭配构成。这些标准件，其质量和可靠性有较好的保证，各种标准零部件的故障模式也有公认的分类方法。在软件领域，可以通用的标准模块非常匮乏，软件开发更多依赖于开发人员的业务素质、智力、人员的组织、合作和管理。在大多数场合，软件的开发、设计几乎都是从头开始，多数程序需要设计人员独创构思，是比较典型的个体劳动产物。

② 制造差异。硬件试制成功之后，批量生产需要建生产线，投入大量的人力、物力和资金。原材料完全相同，制造过程完全相同，生产过程要进行质量控制，对每件产品进行严格检验，但同型号和规格的单个硬件产品在微观结构仍然存在独特的个性。软件在开发成功之后，可以大量地复制，复制件和原版本完全一样，不存在这种微观个性。

③ 维修和维护差异。硬件的一部分维修工作，可以由用户在使用现场进行。软件对普通的用户是不可维护的，只能通过软件开发方的再设计和纠错来消除错误。软件在使用过程中的维护工作比硬件复杂得多。首先，软件在运行期间可能会发生故障，需要进行“纠错性维护”；其次，用户有时需要提高软件的性能和完善软件的功能，必须对软件进行修改，进行“完善性维护”；最后，由于支撑软件运行的硬件或软件环境的改变，也需要对软件产品进行修改，进行“适应性维护”。软件内部逻辑关系复杂，在维护过程中还可能产生新的错误，因此，软件在使用过程中的维护工作远比硬件的维护复杂。

④ 报废原因的差异。软件经过一段时间的使用之后，会被废弃，但是软件被废弃的原因与硬件不同。软件报废的主要原因是：软件版本的功能已经不能满足用户新的需求；硬件报废的主要原因则是：产品经过长时间的使用，已经损坏，无法继续正常工作。

(2)软件的状态、路径复杂性。软件和硬件的差异还体现在软件状态、路径的高度复杂性方面。

① 状态复杂性。例如，设程序 P 有输入变量 X 和 Y，并且有输出变量 Z，在字长为 32 位的计算机上运行。如果 X 和 Y 均只取整数，考虑把所有可能的 X 值和 Y 值作为测试数据，逐个用以驱动程序 P，进行穷举测试，力图全面、无遗漏地“挖掘”出程序中的所有缺陷。这样做可能采用的测试数据组<Xi, Yi>，其中 i 的最大值为：

$$2^{32}\times 2^{32}=2^{64}\approx 10^{20}$$

如果程序 P 测试一组<X, Y>值数据需用 1 毫秒，要完成 10^{20} 组测试，共需 5 亿年的时间。

② 路径复杂性。软件内部逻辑高度复杂。例如，图 1 所示的控制流图大致对应一个 10 至 20 行高级程序设计语言构成的程序，其中有四个条件判断和一个至多执行 20 次的循环，在循环体内有五条路径。则从入口到出口总计有 $5^1 + 5^2 + \ldots + 5^{20}\approx 10^{14}$ 条逻辑路径。假设每运行一组测试用例平均花费 1 毫秒，总共需要 3170 年才能穷尽测试这些逻辑路径。如果再考虑程序输入数据的变化，那情况就更复杂了。

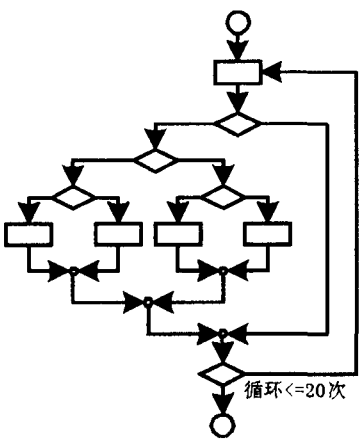

图 1　一个典型程序的控制流图

(3) 软件和硬件在发生故障原因上的差异。硬件是由自然界中基本元素构成的、可用感观或仪器识别的物质实体。硬件的存在形式和生存规律，服从基本的物理和化学规律。软件是抽象的计算机运算程序和数据，不是物质实体，只能书写在文件上或者存储在介质中。软件不受任何物理、化学规律的制约。软件和硬件这种本质差别，使得它们发生故障的原因很不相同，主要体现在以下几点：

① 软件故障是由程序运行过程中软件缺陷被触发所致。软件缺陷是设计原因造成的，如果不进行更改，则将经久不变，时间的长短对软件缺陷不起作

用。硬件故障则是由于物理和化学的作用，使物质的性质和结构发生变化，致使硬件发生故障。

② 软件的故障率将随着使用时间的增加而下降。原因是软件在使用过程中会不断暴露故障、消除故障。随着软件故障的消除，软件可靠性将不断增长。硬件在使用过程中，最终会呈现出可靠性衰减的趋势，因而硬件的故障率一般将随时间增加而上升，特别是具有耗损特性的设备。

③ 环境因素方面。影响软件故障的环境因素是指软件开发环境与使用环境的差异，如操作系统、硬件配置等，外部的自然环境对软件发生故障没有直接的影响。硬件故障与使用现场的自然环境或人为环境密切相关，因此硬件可靠性设计工作中的环境设计十分重要。

软件质量首先是设计、生产出来的，其次是试验、管理出来的。设计质量对于软件质量具有特殊的重要意义。硬件质量除受到设计的影响之外，还受到物质生产过程和物质老化过程的影响，产品的老化是无法避免的。相比之下，软件的质量对设计的依赖程度更大。这里所指的软件设计是广义的设计，它包括了从需求分析开始，直至实现的全过程。

总起来看，软件发生故障的原因主要有两个方面：① 内部因素。软件本身状态、路径高度复杂，人们在分析、设计这样复杂的系统时难免犯错误，导致软件中留下缺陷。② 外部因素。软件的运行环境与其开发环境往往差异很大，通常是在开发环境中进行测试，而且要达到充分的测试往往很困难，要进行穷举测试，达到完全的路径覆盖、状态覆盖几乎是不可能的。因此，即使在开发软件产品时进行过相当多的测试，但是也不能完全排除可能存在的故障，更何况软件产品的开发环境与使用环境往往存在较大差异，因此在使用软件产品时仍然可能发生故障。

3　软件失效机理

明确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，且不同的软件差异可能更大，因而软件失效机理常常有不同的表现形式。譬如有的失效过程比较简单，易于追踪分析、诊断故障；而有的失效过程非常复杂，难于甚至不可能加以详尽描述和分析，尤其是运行于复杂实时环境中的大型软件。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[11]。

3.1 软件错误(software error)

软件生存周期的各个阶段都需要人的参与。因为人的思维和交流不可能完美无缺，出现错误是难免的。与此同时，随着计算机控制对象的复杂程度不断提高和软件功能的不断增强，软件的规模也在不断增大。例如，Windows NT 操作系统的代码大约有 3200 万行[12]。这使得错误更可能发生。软件错误是指在软件生存周期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。可见软件错误是一种人为过程，相对于软件本身，是一种外部行为。

下面举例说明。考虑一个 C 语言程序，对用户任意输入的 10 个整数，采用冒泡排序算法进行排序，结果按照不减序排列。正确的程序为：

```
void Bubble_Sort()
{
  int temp, i, j, X[10];
  scanf("%d%d%d%d%d%d%d%d%d%d",
     &X[0], &X[1], &X[2], &X[3], &X[4],
     &X[5], &X[6], &X[7], &X[8], &X[9]);
  for (i = 0; i <= 8; i++)
    for (j = 0; j <= 8; j++)
      if (X[j] > X[j+1]){
          temp = X[j];
          X[j] = X[j+1];
          X[j+1] = temp;
         }
  printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d",
     X[0], X[1], X[2], X[3], X[4],
     X[5], X[6], X[7], X[8], X[9]);
}
```

如果由于人为错误，将外层 for 循环的循环控制变量 i 的终值错误地由“8”写成了“7”，这就是一个软件错误，其结果是得到如下不可接受的程序：

```
void Bubble_Sort()
{
  int temp, i, j, X[10];
  scanf("%d%d%d%d%d%d%d%d%d%d",
     &X[0], &X[1], &X[2], &X[3], &X[4],
     &X[5], &X[6], &X[7], &X[8], &X[9]);
  for (i = 0; i <= 7; i++)
    for (j = 0; j <= 8; j++)
      if (X[j] > X[j+1]){
          temp = X[j];
          X[j] = X[j+1];
          X[j+1] = temp;
```

```
        }
    printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d",
        X[0], X[1], X[2], X[3], X[4],
        X[5], X[6], X[7], X[8], X[9]);
}
```

3.2 **软件缺陷**(software defect)

软件缺陷是存在于软件(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，此时称软件缺陷被激活。当软件意思是指程序时，软件缺陷(defect)与软件(程序)污点(bug)同义。

现在仍然考虑以上冒泡排序程序。由于将外层 for 循环的循环控制变量 i 的终值错误地由"8"写成"7"，导致产生一个软件缺陷。该缺陷由以下语句构成：

```
for (i = 0; i <= 7; i++)
  for (j = 0; j <= 8; j++)
    if (X[j] > X[j+1]){
        temp = X[j];
        X[j] = X[j+1];
        X[j+1] = temp;
    }
```

可见软件缺陷是存在于软件内部的、静态的一种形式。

3.3 **软件故障**(software fault)

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。譬如软件处于执行一个死循环过程或者其中数组元素的下标越界时，就说软件出现故障。出现软件故障时若无适当措施（容错)加以及时处理，便产生软件失效。显然，软件故障是一种动态行为。

例如，对于以上包含缺陷的冒泡排序程序，对于输入数据<9,1,1,1,1,1,1,1,1,0>，在外层 for 循环执行结束之后，printf 语句执行之前，数组 X 的值为<1,0,1,1,1,1,1,1,1,9>，此时软件出现故障。

3.4 **软件失效**(software failure)

软件失效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。在上述的软件故障例子中，由于没有容错措施，即没有限制和排除软件故障的措施，所以将得到一个不可接受的程序输出："1, 0, 1, 1, 1, 1, 1, 1, 1, 9"。这便是一个软件失效。而正确的排序结果应该是"0, 1, 1, 1, 1, 1, 1, 1, 1, 9"。

综上所述，软件错误是一种人为错误。一个软件错误必定产生一个或多个软件缺陷。当一个软件缺陷被激活时，便产生一个软件故障。同一个软件缺陷在不同条件下被激活，可能产生不同的软件故障。出现软件故障时如果没有及时的容错措施加以处理，便不可避免地导致软件失效。同一个软件故障在不同条件下可能产生不同的软件失效。

值得注意的是，在有些情况下，虽然软件缺陷被激活，但是并不一定产生软件故障。例如，对于以上含有缺陷的冒泡排序程序，只要输入数据中最小元素的值不是恰好位于数组的最后一个元素(即 $X^{[9]}$)，该程序运行时都不会产生软件故障，即最终都能得到正确的排序结果。

4　硬件系统故障诊断技术

当前硬件系统故障诊断研究和实践取得了较大的进展，并且有大量的成功案例。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，或称状态监测，是采集系统运行参数，获取系统状态信息，为系统进行状态分析、状态识别奠定基础，提供条件。故障分离，或称分析诊断，是根据故障检测所获得的系统状态信息，进行分析，包括进行故障分析，以识别系统是否存在故障，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，按照已识别的系统现有状态的不同情况，或者采取有针对性的纠正措施，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。

硬件系统故障诊断的技术一般可按以下三种方法分类：按方法复杂程度分类、按检测手段分类和按方法理论基础分类。

(1)按方法复杂程度分类

按故障诊断方法的复杂程度分类，一般可分为简易诊断法和精密诊断法。简易诊断相当于人的初级健康诊断，一般由现场工作人员实施，对监测结果不需要作进一步的处理和分析，仅用有限的检测数据和合格范围加以比较，即可判断设备或系统是否存在故障。通常所称的"故障诊断"不是这种简易诊断，而指的是较为复杂的精密诊断。一般情况下，当简易诊断发现设备或系统存在异常现象时，应转入精密诊断。

精密诊断是对异常设备或系统进行的精确诊断，通常由故障诊断水平较高的专家来进行。精密诊断不仅要对检测数据作进一步处理、分析，确认设备或系统是否存在故障，还要检测、隔离故障，分析产生故障的原因和机理，判断故障性质和程序，提出治

理和预防的措施。

(2)按检测手段分类

按检测手段分类,故障诊断一般可分为以下六种。

① 直接观察诊断法。根据人的经验,通过人的感觉器官对设备的状态进行视、听、闻、触摸等直接观察判断的方法称为直接观察法。这种方法可以获得可靠的第一手资料,但观察的对象主要是静止的、能直接观察到的设备现象,如发热、发光、发声、有气味、有东西晃动等,这是一种较为简单的物理检查手段。在实际诊断中,主要用于设备状态设置错误、接口连接错误以及设备磨损、电流过载等故障的判断。

② 性能参数测定法。通过测定系统或设备的性能参数来判定系统或设备状态的方法称为性能参数判定法。性能参数一般包括电信号、磁特性、功率、物体运动、系统特征量、机械设备性能等。性能参数测定可分为三类:一类是用物理和化学手段直接测定设备的物理与化学性能;二类是用输入和输出之间的关系来表示系统的性能;三类是用两个或两上以上的变量之间的关系来识别系统的特性。采用这种方法,通常需要使用专门的检测设备或手动、自动的检测系统。

③ 无损检测诊断法。对被测设备不造成损坏的非接触式检测方法称为无损检测诊断法,一般包括声、光、热等检测方法。主要用于一些无法或不便于安装接触式传感器的设备故障的诊断。采用这种方法,通常需要使用专门的无损检测设备,如超声波检测仪、红外检测仪及 X 光透视仪等设备。

④ 振动与噪声测定诊断法。对于机械和电子系统,振动和噪声是很重要的诊断信息,应用比较广泛。振动与噪声测定诊断法主要用于机电设备故障的诊断。通过测量振动体的位移、速度、加速度的大小,以及振动的频率、周期、相位角、频谱图等,可以对设备的故障进行诊断。通过对设备噪声测量与分析,研究其频率组成和各分量的变化情况,可以确定设备故障的部位和程度。

⑤ 高低温诊断法。通过设置高温环境或低温环境,对设备性能进行测定的方法称为高低温诊断法。高低温诊断法主要用于受温度影响较大的设备的故障判定,例如电子产品、焊接设备等。采用这种方法,一般要使用高低温箱或环境实验室等检测设备。

⑥ 化学成份测定法。通过测定设备零部件或元器件内部化学成份来判定设备故障的方法称为化学成份测定法。化学成份测定法主要用于设备腐蚀、磨损、变质的产物或设备中多余物成份的判定,以分析故障形成的机理。采用这种方法,一般需要专门的理化实验室进行检测。

(3)按方法理论基础分类

按故障诊断方法理论基础分类,一般可分为以下三类[13]。

① 基于模型的诊断方法。核心思想是用解析冗余取代硬件冗余,通过构造观测器估计出系统的输出值,将其同输出测量值相比较,从中获取故障信息。基于模型的诊断方法一般可分为基于参数估计的诊断方法和基于状态估计的诊断方法两种。

基于参数估计的故障诊断方法的思路是:由机理分析确定系统的模型参数和物理元器件参数之间的关系方程 $u=f(v)$,由实时辨识求得系统的实际模型参数 u',由 $u=f(v)$ 和 u' 求解实际的物理元器件参数 p',将 p 和 p' 的标称值比较从而得知系统是否有故障及故障的程度。

基于状态估计的故障诊断方法最早是由贝尔德(Beard)于 1971 年首先提出的,这种诊断方法发展至今已形成三种基本方法:(a) 贝尔德提出的故障检测滤波器的方法;(b) 梅恩乐(Menra)和培松(Peshon)提出的基于卡尔曼滤波器的方法、德拉克(Dlark)提出的构造卡尔曼滤波器阵列;(c) 德克特(Deckert)提出的一致性空间的方法。一致性空间法实现故障诊断可分为两步:一是形成残差,即真实系统的输出与状态观测器或卡尔曼滤波器输出的差值;二是从残差中提出故障特征进而实现故障诊断。

多年以来,基于模型的诊断方法获得了深入的研究,但在工程实践中,由于获得系统精确模型的困难性,限制了其使用范围和效果。

② 基于信号的诊断方法。核心思想是利用信号模型,如相关函数、频谱等,提取诸如方差、幅值、频率等特征值,检测出故障。例如,旋转机械中的滚动轴承在出现疲劳脱落、压痕或局部腐蚀等故障时,其振动信号的功率谱就会出现相应的特征,利用这种特征就可对系统或设备的故障进行诊断。基于信号的诊断方法一般可分为以下四种:直接测量系统输入输出方法、小波变换方法、主元分析方法和基于信号融合的方法。

③ 基于知识的诊断方法。基于知识的诊断方法不需要对象的精确数学模型,诊断对象作为一个有机整体被研究,以知识处理技术为基础,诊断问题的求解致力于通过模拟领域专家在推理过程中控制和

运用各种诊断知识的行为而获得解决，目前研究工作发展迅速、成果迭出。基于知识的诊断方法一般可分为以下五种：专家系统诊断法、模糊故障诊断法、基于故障树的故障诊断法、基于神经网络的故障诊断法和基于智能体的故障诊断法。

5 软件故障诊断

我们认为，软件故障诊断是根据软件（包括程序、数据和文档）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。

人在参与软件生存周期的各个阶段工作时都难免会出现错误。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、维护等各阶段所造成的缺陷，所采用的软件评审等也属于软件故障诊断的手段。

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、软件故障模式、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，这是软件诊断技术的关键。软件故障诊断突出了诊断的目的性，即寻找和发现软件故障状态而进行诊断，也包括无故障状态在内，但强调故障状态的重要性。

软件故障诊断的过程包括故障检测、故障定位、故障排除、回归测试、系统测试和交付等几个阶段。软件故障检测是软件故障诊断的第一步，通过静态检查、动态运行等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，并为软件故障定位提供依据。

软件故障定位，是指根据软件故障检测提供的能反映软件状况的征兆或特征参数的变化情况，或与某故障状态参数（模式）进行比较，并进一步收集软件的历史和使用信息，识别软件是否正常运行或存在故障，复现软件故障过程，诊断软件故障的性质和程度、产生原因或发生部位，确定缺陷，为纠正缺陷、排除软件故障做好准备。

软件故障排除是指当诊断出软件中存在缺陷，就其原因、部位和危险程度进行研究，决定纠正缺陷、排除软件故障的办法，包括修改程序代码、数据或软件文档等。软件故障排除属于软件维护的范畴。

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。

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。研究软件故障诊断具有重要意义。本文研究了软件和硬件的区别，分析软件故障产生的原因和软件失效机理。当前硬件系统故障诊断的研究和实践取得较大的进展，本文在介绍现有硬件系统故障诊断技术的基础上，对软件故障诊断进行了初步探讨。

在今后的工作中，我们将进一步深入研究软件故障的特点和各种硬件系统故障诊断技术，分析硬件系统故障诊断案例，研究软件故障诊断技术，并且应用于软件故障诊断实践。 ◇

参考文献

[1] Weyuker E J. Testing component-based software: A cautionary tale. IEEE Software, 1998, 15(5): 54-59.

[2] IEEE. IEEE Std 610.12-1990, IEEE standard glossary of software engineering terminology. December, 1990.

[3] 蔡开元. 软件可靠性工程基础. 北京: 清华大学出版社, 1995.9.

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

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

[6] 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, 17: 69-71,91.

[7] 郑人杰. 计算机软件测试技术. 北京: 清华大学出版社, 1992.

[8] 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.

[9] 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.

[10] 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.

[11] 蔡开元. 软件可靠性工程基础. 北京: 清华大学出版社, 1995.

[12] Weyuker E J. Evaluation techniques for improving the quality of very large software systems in a cost-effective way. The Journal of Systems and Software, 1999, 47: 97-103.