国防特色教材·航空宇航科学与技术

National Defense TEXTBOOK

软件可靠性工程

RUANJIAN KEKAOXING GONGCHENG

赵 靖 王延斌 曲立平 史长亭 编著

西北工业大学出版社
北京航空航天大学出版社 北京理工大学出版社
哈尔滨工业大学出版社 哈尔滨工程大学出版社

国防特色教材·航空宇航科学与技术

软件可靠性工程

赵　靖　王延斌　曲立平　史长亭　编著

西北工业大学出版社

北京航空航天大学出版社　北京理工大学出版社
哈尔滨工业大学出版社　哈尔滨工程大学出版社

前　言

随着软件产品与计算机一起渗透到各类军用、民用设备中，软件的质量问题也引起了人们越来越广泛的关注。软件可靠性指标是软件质量属性中最重要的内容之一，它定量地描述了软件开发和使用过程中出现的失效。软件可靠性工程的相关技术和理论贯穿了软件生命周期的每一个阶段，为保证软件可靠性和软件质量，在生命周期的每一个阶段，都可以采用相应的模型评估与预测软件的可靠性。可是，在复杂的软件系统中，可靠性经常又是最捉摸不定的。由于世界上的许多工程项目和系统都按软件进行开发，因此软件可靠性应达到相当的水平，既稳定又经济是至关重要的。软件失效可以是最大的新闻，因为它给人们带来很多麻烦，甚至危及人们的生命。

软件可靠性工程在软件工程领域逐渐成为一个重要的分支，成为一门崭新的技术。1988 年“软件可靠性工程”一词登上学术讲台。J. D. Musa 认为软件可靠性工程是“一门以减少基于软件的系统在运行中不满足用户要求的可能性为目标的应用科学”。他同时还说明软件可靠性评估是软件可靠性工程研究的内容之一。

本书是依据软件生命周期理论介绍了软件测试可靠性和运行可靠性。全书共分 8 章，内容广泛，从理论和实践两个方面系统地介绍了软件可靠性工程。第 1 章讨论了软件可靠性工程相关的定义、发展状况；第 2 章介绍了基于软件生命周期的软件测试；第 3 章介绍了软件可靠性与系统可靠性；第 4 章介绍了软件的运行剖面与测试剖面，对剖面差别进行了分析，并且对差别进行了建模，指出了影响软件可靠性评估精度的一个重要方面就是软件测试剖面与运行剖面的差别。第 1～4 章是软件可靠性工程的基础，同时也是理解后面各章节的基础。第 5 章介绍了基于软件生命周期的软件可靠性建模，以及生命周期各个阶段软件可靠性工程所采取的建模技术，同时为进一步提高软件可靠性的评估精度，将测试剖面与运行剖面的差别融入到软件可靠性增长模型中。第 6 章介绍了预测分析方法以及与这些方法相关联的软件比较标准。第 7 章介绍了实践经验，现场数据分析；第 8 章介绍了软件容错设计技术。

本书努力跟踪软件可靠性工程的新技术、新发展，力求反映软件可靠性工程

领域的最新成果、最新理论，以保持本书的先进性和实用性。

本书是面向本科生和研究生而设计的，同时适用于使用软件的高级管理者、软件开发人员、测试人员、质量管理人员，以及软件工程、可靠性分析、应用统计和其他想深入了解软件可靠性的工程技术人员等。

本书编写分工如下：第1,2章由哈尔滨工程大学计算机学院曲立平编写；第4,6章由哈尔滨工程大学计算机学院赵靖编写；第7,8章由哈尔滨工程大学计算机学院史长亭编写；第3,5章由哈尔滨工业大学机电学院王延斌编写。

在本书编写过程中，得到了国家自然基金项目(60873036,90718003)、国家863项目(2007AA01Z01)、国防基础项目和哈尔滨工程大学青年骨干教师计划项目的支持，在此一并表示感谢！

由于水平所限，书中难免有不妥之处，敬请广大读者和专家批评指正。

编著者

2010年9月

目　录

第1章 导 论

1.1 可靠软件的需求

随着现代信息技术的发展，计算机已经渗透到国民经济和国防建设的各个部门，甚至人类活动的各个领域。计算机应用领域的扩展和功能需求的进一步完善，使得软件系统的规模日益增大，结构和功能日益复杂。例如，航天飞机的机载系统有近 50 万行代码，地面控制和处理系统有大约 35 万行代码，在对系统进行大规模削减之后，国际航天局仍然需要数百万行的软件代码来操纵导航、通信和实验用的大量硬件设备。在美国电信业中，电信线路的正常运转需要数百个软件系统的支持，其代码总量超过一亿行；在航空电子领域，几乎所有新的载荷仪器都有其含有丰富内嵌式软件的微处理器系统；在战机航电系统中，软件实现的功能随着每一代计算机而翻番。在我国的专项工程中，某些装备的机载软件已经超过 100 万条语句，下一代装备的软件规模还将成倍增加。

软件可靠性直接关系到计算机系统乃至更大的系统能否在给定时间内完成指定的任务。不可靠的软件引发的失效可能给软件的使用者或者软件开发人员带来灾难性的后果。在军事领域：①1996 年 6 月 4 日，欧洲航空航天局耗资 67 亿美元研制的阿丽亚娜 V 火箭，在首次飞行实验中，点火后 37 s 在空中爆炸。调查委员会调查分析后认为：事故是由惯性制导系统软件中一个故障引起的。这个软件故障不仅造成巨大的财产损失，而且使这一项目的进程大大延期，其损失难以估量。②在海湾战争中，一个软件故障扰乱了“爱国者”导弹的雷达跟踪系统，导致发射导弹时产生了 1/3 s 的时间误差，结果未能击中伊拉克发射的飞毛腿导弹，造成美军 28 人死亡，98 人受伤。在民用领域：①丹佛新国际机场投资 1.93 亿美元建立的自动化行李系统，由于其中的软件故障，该机场的开放时间推迟了半年以上。②20 世纪 90 年代中期，放射性治疗设备的软件错误导致美国的 Therac－25 放射治疗仪多次产生超剂量辐射，在加拿大造成多起癌症病人死亡。③1992 年 10 月 26 日，伦敦救护服务中心的计算机辅助发送系统出现崩溃，致使这个全世界最大、每天要接受 5 000 个待运救病人的救护服务机构全部瘫痪。

实际上，由于软件的不可靠导致系统失效，最终酿成重大损失的事例不胜枚举。因此，提高软件的可靠性已经成为软件开发人员迫切需要解决的问题。软件可靠性工程技术可以有效地对软件产品特性进行度量和预测，对软件开发过程的状态进行控制、设计并开发出高可靠性的软件。

1.2 基本定义

1983年，美国IEEE计算机学会给出的软件可靠性(reliability)定义如下：

(1)在规定的条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和系统使用的函数，也是软件中存在的错误的函数。

(2)在规定的时间周期内，在所述条件下程序执行所要求的功能的能力。

软件可靠性是软件质量的指标之一。全面衡量软件质量还需要其他一些用户关心的因素，如功能(functionality)、可用性(usibility)、性能(performance)、可服务性(serviceability)、能力(capability)、可安装性(installability)、可维护性(maintainability)和文档。

软件可靠性已被公认为是系统可依赖性的关键因素，因为它可以定量地衡量软件的失效性——可导致软件失去作用甚至危及整个系统。软件可靠性也可被认为是从软件质量方面满足用户需求的最重要的因素。

软件可靠性工程(Software Reliability Engineering，SRE)是一门以减少软件系统在运行中不满足用户要求的可能性为目标的应用科学。它是以软件的一系列重要特性为中心展开的，其研究内容包括：

(1)软件可靠性的分析：确定指标、设计开发过程、进行预计、分析失效严重性等。

(2)软件可靠性的测量：用失效数据和软件可靠性模型估计(或测量)软件运行的可靠性。

(3)软件可靠性的管理：利用可靠性测量和其他信息来控制和改进开发过程，并对采购或重用的软件进行管理。

(4)开发过程的改进：确定影响软件可靠性的因素，改进费用效益关系。

注意，软件可靠性定义的3个要素为：失效(failure)、时间(time)和操作环境(operational environment)。还应在软件可靠性度量(measurement)中注意评价(estimation)与预测(prediction)的区别。下面给出这些术语的详细定义。

1.2.1 错误与失效

从软件可靠性定义中可以看出，软件可靠性的关键是如何能保证软件无故障运行，使其功能更好地满足用户需求。软件可靠性与软件中的缺陷直接相关。下面介绍软件可靠性工程中几个与缺陷有关的概念。

1. 软件错误(software error)

错误是指计算、观察或测量值或状态与真实的、规定的或理论上的正确值或状态之间不相符。软件错误是指在软件生存期内出现的不希望或不可接受的错误，它是在软件设计和开发

过程中引入的，其结果是导致软件缺陷的产生。软件错误的例子如下：

(1)启动错：程序不按要求启动。

(2)输入范围错：程序不能正常检测输入数据的范围。

(3)说明错：输入数据的约束范围说明模糊。

(4)算法错：计算给定数学方程的算法不能正常执行。

(5)边界错：数组索引越界。

软件错误是一种人为结果。它是由人的不正确或疏漏的行为造成的，是软件开发活动中不可避免的一种行为过失。软件错误相对于软件本身，是一种外部行为。在大多数情况下，软件错误可以被查出并排除，但是在某些情况下，仍然会有部分软件错误隐藏于软件内部。在非容错系统中，错误等于失效。

2. 软件缺陷(software defect)

软件缺陷是指存在于软件(包括说明文档、应用数据、程序代码等)中的不希望的或不可接受的偏差，例如，缺少一个逗号，多一条语句等。

软件缺陷是程序本身的特性，以静态形式存在于软件内部，是软件开发过程中人为错误造成的。当软件运行于某一特定条件时，软件缺陷将导致系统出现软件故障，即软件缺陷被激活。

在软件生命周期的各个阶段，特别是早期设计和编码阶段，设计者和编程人员的行动，如需求不完整、理解有歧义、没有完全实现需求或潜在需求、算法逻辑错、编程问题等，使软件在一定条件下不能或将不能完成规定功能，这样就不可避免地存在软件缺陷。软件一旦存在缺陷，它将潜伏在软件中，直到被发现并且被正确修改。反之，如果在一定环境下软件运行正确，软件缺陷将继续潜伏在软件中，除非环境发生变化。此外，软件中的缺陷不会因使用而损耗，所以软件缺陷是无损耗地潜伏在软件中的。

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。软件故障是一种动态行为，是软件缺陷被激活后的表现形式。

软件故障总是由软件错误引起的，但是软件错误不一定引起软件故障。当软件运行时出现软件故障，并且没有采取措施处理时，便产生软件失效。

4. 软件失效(software failure)

软件失效是软件运行时产生的一种不期望的或不可接受的外部行为结果，是系统运行行为对用户要求的偏离。软件失效的例子如下：

(1)某预定代理商发现数据库接收了一个查询，却未产生任何反应。

(2)某银行出纳要求计算机给出某一张支票账户的余额,计算机却显示"读错"以示读支票余额操作未成功。

(3)某飞机驾驶员发现自动飞行控制系统未能按指令保持某确定高度。

(4)某电话用户拿起电话拨号,但转接系统未按要求连接到正确线路。

所有的软件失效都是由于软件故障引起的。一次软件失效实质上就是引起系统失效的一个软件故障,但是软件故障不一定使软件失效。判断软件是否失效的依据有:系统死机、系统无法启动、不能输入/输出显示记录、计算数据有误、决策不合理以及其他削弱或使软件功能丧失的事件或状态。

用户可以根据软件失效对系统服务的影响来判定失效的严重程度级别,如:灾难性失效、重大失效、微小失效。这些严重程度级别的定义因系统而异,但在某一给定系统内应保持术语的一致性。

综上所述,软件错误是一种人为错误,在一个软件的开发过程中,这种主观的错误是无法避免的。一个软件错误必定产生一个或多个软件缺陷。软件缺陷往往十分隐蔽,不容易被发现和改正,换言之,真正影响软件可靠性的是那些在开发过程中没有被发现的软件缺陷。这些软件缺陷只有通过不断的测试和使用,才能以软件故障的形式表现出来。软件故障如果没有得到自动、有效、及时的处理,便不可避免地导致软件失效。一个软件中究竟存在多少软件错误和软件缺陷,这一点是无法确定的。无法确定总量,也就无法达到完全排错。所以说任何软件都是存在缺陷的,软件开发人员只能尽量地排错,使软件在顾客满意的情况下运行。

1.2.2 时间

可靠性量化通常用相关时间来定义。这里的时间有 3 种,分别是日历时间、时钟时间和执行时间。

日历时间是指日常生活中使用的日、周、月、年等计时单元。时钟时间是指从程序运行开始,到运行结束所用的时、分、秒,其中包括等待时间和其他辅助时间,但是不包括计算机停机占用的时间。执行时间是指计算机在执行程序时,实际占用的中央处理单元 CPU 的时间,所以执行时间又称 CPU 时间。如果计算机被程序连续占用,并且该程序占用 CPU 的一个时间段,则执行时间与时钟时间成正比。

日历时间、时钟时间和执行时间的区别可以用下面的例子说明。

假设某软件系统 1 周之内运行了 42 h,CPU 的工作时间是运行时间的 2/3,则日历时间是 1 周,时钟时间是 42 h,CPU 时间是 28 h。

对于软件可靠性测量和建模而言,执行时间比日历时间更准确。然而,为了便于用户理解,可靠性量化必须最终与日历时间相联系。当经理、工程师和其他用户将结果与不同系统作比较时,尤其如此。因此,需要在日历时间与执行时间之间进行转换。如果不容易得到执行时

间，可以用其他时间近似，如时钟时间、加权时钟时间、主要运行时间或其他应用程序易得的计时单位，如处理或测试箱等。

时间基准确定后，失效可以用3种方式表示：累积失效函数(the cumulative failure function)、失效强度函数(the failure intensity function)和平均失效时间函数(the meantime to failure function)。累积失效函数，也称为均值函数，表示与每一时间点相关的平均累积失效。失效强度函数，也称为失效率函数(failure rate function)或失效发生率(rate of occurrence of failures)函数，表示累积失效函数的变化率。平均失效时间(meantime to failure，MTTF)表示观察到下次失效的期望时间。显然，上述3种度量标准密切相关且可相互转化。例如，失效强度是由累积失效函数相对于时间推导所得到的瞬时值。如果可靠性函数呈指数分布，那么其平均失效时间为失效强度函数的导数。

与时间有关的另一个量为平均修复时间(meantime to repair，MTTR)，它表示在观察到失效后，修复系统所需要的时间。对硬件部件来说，平均修复时间是一个很容易估计的评价标准，因为典型的硬件修复过程是通过诊断确定系统出错的部件，然后用同样的或等效的部件来替换它，这一标准过程使对平均修复时间的估计十分精确。然而对于软件，情况就不同了。软件失效的暂时恢复可以是简单的重装软件或重运行软件直到失效再次出现，而永久性的恢复需要调试、修改、确认、重装该软件。因此，软件的平均修复时间因系统而异且因时间而异。

在系统的MTTF和MTTR测定后，其可用性(availability)即可求得。可用性是指在需要时系统可用的概率。通常，它可按如下公式计算：

$$\text{可用性}=\frac{\text{MTTF}}{\text{MTTF}+\text{MTTR}}$$

1.2.3 操作环境

在不同的操作环境下，软件的可靠性是不同的。严格地说，定义软件可靠性所要求的操作环境，主要指对输入数据的要求和计算机当时的状态(软件环境)，而其他一切支持系统及因素都假定对该软件是理想的，即不会影响软件的正常运行。明确规定的操作环境可以有效地判断软件失效的责任在用户方还是在研制方。

软件操作环境通常以操作概图(profile)来描述，它指明软件操作所需的环境。操作概图是软件可执行的操作及其发生概率组成的集合。操作指一组包含相同处理的运算。典型的操作概图如图1.1所示。

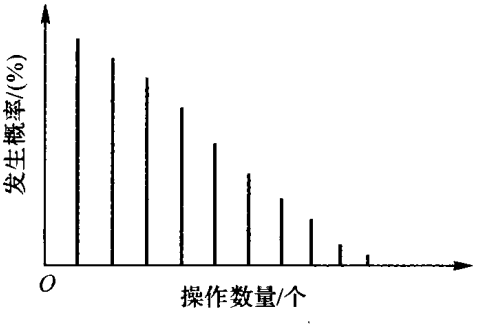

图1.1 操作概图

通常，软件操作的数量可能极大。当不可能确定所有操作及其概率时，常使用基于输入状态(系统状

态)的分组或划分成域的操作。当无法得到操作概图或只能得到近似图时,可以利用可靠性增长测试时得到的代码覆盖数据来评价软件可靠性。

1.2.4 失效数据采集

软件失效数据是进行软件可靠性分析和预测的基础,在软件可靠性研究中占据重要地位。在实际的开发和运行环境中,通常采集两类失效数据:失效计数数据(failure-count data)和失效间隔时间数据(time-between-failures data)。

1. 失效计数数据

失效计数数据跟踪单位时间内检测到的失效次数,常见于工程报告和安装说明中。某些软件可靠性模型的公式中直接采用该数据。

表 1.1 给出了一个典型的失效计数数据集。采集到失效数据后,需要一个特定的过程对相应的软件错误进行修改并确认。

表 1.1 失效计数数据

时间/h	周期内失效数/次	累计失效数/次
8	4	4
16	4	8
24	3	11
32	5	16
40	3	19
48	2	21
56	1	22
64	1	23
72	1	24

软件问题报告机制既可以是手工的也可以是自动的,问题报告可以以易于提取的方式存储在计算机数据库系统中或手工文件系统中。时间单位应在 CPU 速度、人员占用时间、测试强度等方面保持一致。通常无法得到用于检查一致性的有关信息,所能得到的仅是每一时间间隔内的失效计数。而在每一时间间隔内的测试强度可能不一致。例如,某间隔内的测试成员可能为其他间隔内测试成员的 2 倍。解决这一问题需要大量附加工作,如与测试者交流信息,甚至还要重新检查有关的过去时间记录。总的来说,较长时间的测试可以消除短时间间隔内产生的偏差,所以形成失效计数的时间越长,结果就越稳定。

另外,由于错误修复通常不是即时的,因此可能由一个相同错误导致多个失效。其结果是难以确定哪次失效与哪个单一错误有关。这给问题报告的及时性和精确性增添了不确定因素。

问题报告中还存在一些其他问题。例如，如何确定某一问题是否是软件失效——因为对于某些组织而言，问题报告可能不仅仅用来反映软件失效，还用于反映软件中任何形式的异常。另外，问题报告中所反映的时间可能不是失效检测的时间，而是填报告的时间。至于报告率(每周、每月或每年报告失效数)，可视用户的总体测量目标而定。建议采用与目标一致的最小时间单元内所报告的失效次数。如果结果不够好，将时间间隔组合到下一级。例如，日到周，周到季等。前面提到的稳定结果有助于模型的建立。

2. 失效间隔时间数据

典型的失效间隔数据详见表1.2。此类数据可以通过启动自动在线过程监测并记录失效次数而直接得到，也可以通过间接地人工跟踪失效记录，细察失效的后序时间来得到。如果可能，采集时最好记录程序的执行时间，而不是时钟时间，因为执行时间是处理器在执行软件指令时所花费的实际时间，它可以给出软件的更真实的描述。使用执行时间建模，较之于单纯使用时钟时间更有优越性。但是，因为要有一个操作系统的监视器，所以要因数据更难得到而多花一些代价。

表1.2 失效间隔时间数据

失效数/次	失效间隔/h	失效时间/h	失效数/次	失效间隔/h	失效时间/h
1	0.5	0.5	13	0.9	26.1
2	1.2	1.7	14	1.7	27.8
3	2.8	4.5	15	1.4	29.2
4	2.7	7.2	16	2.7	31.9
5	2.8	10.0	17	3.2	35.1
6	3.0	13.0	18	2.5	37.6
7	1.8	14.8	19	2.0	39.6
8	0.9	15.7	20	4.5	44.1
9	1.4	17.1	21	3.5	47.6
10	3.5	20.6	22	5.2	52.8
11	3.4	24.0	23	7.2	60.0
12	1.2	25.2	24	10.7	70.1

另一种得到失效间隔时间数据的方法是通过时间单元平均利用率算法来调整时钟时间。如果无法直接得到失效间隔时间数据，而仅有每一时间单元发生失效的次数数据，那么失效间隔时间可通过下面将要论及的方法间接得到。

3. 两类输入数据间的转换

许多可靠性建模程序都具有根据失效计数数据或失效间隔时间数据估计模型参数的能力，而统计建模技术可兼施于两种数据。如果程序只提供了一种类型的数据，那么可能有必要

变换成另一种数据。

如果程序提供了失效间隔时间数据，而期望输入数据是失效计数数据，那么可以先将失效间隔时间数据变换成累计次数数据，然后计算在规定时间内发生失效的累计次数。将表 1.2 中的失效时间间隔数据转化成表 1.1 中的失效次数数据的过程十分直观。

如果程序提供了失效计数数据，而期望输入数据为失效间隔时间数据，那么可以通过以下两种方法完成转化：

第一种方法是在规定时间间隔内随机分配失效。随机性对某些模型不会造成 15% 以上的估计差错。

第二种方法是在时间长度内均匀分配失效。该种方法最易实现。例如，假设时间间隔为 3 h，在此期间出现 3 次失效。可以认为失效间隔时间为 1 h。图 1.2 给出了 3 种在 3 个 1 h 期间分配 3 个失效的可能方法。在图 1.2(a)中每次失效发生在 1 h 期间的开始，在图 1.2(b)中每次失效发生在 1 h 期间的中间，在图 1.2(c)中每次失效发生在 1 h 期间的结尾。

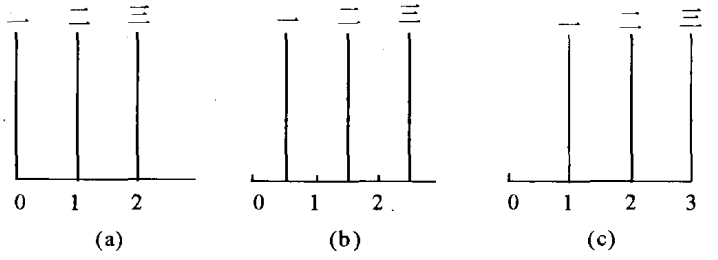

图 1.2 根据一致分布分配失效的方法

应该注意，除非十分必要(如，模型需要失效间隔时间数据)，通常不提倡从失效计数数据到失效间隔时间数据的转化，因为这种转化会引起模型操作的噪声。

一些软件可靠性工具具有进行数据转换的能力。

1.2.5 评价与预测

软件可靠性测试包括两方面：可靠性评价和可靠性预测。

可靠性评价是对系统测试和系统运行期间得到的失效数据应用统计推理，从而断定系统当前的软件可靠性。它是对从过去到当前所得到的可靠性的度量。其主要目的是评价当前软件的可靠性，并确定一个可靠性模型是否为回溯的正确依据。

可靠性预测是利用已知的任何软件度量与规程确定未来软件的可靠性。在软件开发的不同阶段，可靠性预测可采用不同的预测技术。

(1)在已知失效数据时，如软件在系统测试与操作期，可用评价技术来定参和验证软件可靠性模型，从而完成对未来可靠性的预测。

(2)在不知失效数据时,如软件在设计与代码生成阶段,软件开发过程中得到的规程和所得产品的特性可被用来确定系统的软件可靠性。

第一种预测技术又称可靠性预测(reliability prediction),第二种预测技术又称早期预测(early prediction)。表1.3概括比较了这两种预测技术。

表1.3 早期预测和可靠性预测比较

项目 \ 技术	早期预测	可靠性预测
何时使用	预测前期	测试运行期
测试目的	初始可靠性和失效率	连续可靠性评价
依赖失效数据	不依赖	依赖
需要输入数据	过程和产品评价	失效数据
输出数据	错误密度,总错误数	全部可靠性相关测度
性质	静态,单时测度	动态,连续适应
可靠性预测	直接换算	直接可测
时间性	早期	晚期
精确度	不确定	更好

从表1.3可以看出,为了进行可靠性预测,考查某些软件产品的实际失效数据是十分必要的。这些失效数据可以是系统在测试早期记录的数据(如失效计数或失效间隔时间),该数据用于校准统计评价模型,使之适于应用。校准后的模型可用来测量和预测软件产品的可靠性。早期预测无须真实数据,而用其他参数来刻画软件产品及其开发、测试和维护过程。早期预测技术通常用来求静态度量标准,如固有错误密度、期望错误总数和初始错误率。通常在系统进入测试期之前执行一次。早期预测对软件管理中的耗费评价(cost estimation)、资源计划(resource planning)、方案确证(schedule validation)和质量预报(quality forecasting)都十分重要。一方面,可靠性预测需要执行软件,而早期预测则可在执行软件之前进行。换言之,在确定软件可靠性方面早期预测的及时性较优。然而,作为连续适应过程的可靠性预测,因为评价模型根据失效数据精确度量软件可靠性,所以在对软件可靠性的度量精度方面较优。另一方面,早期预测模型不考虑可靠性的增长,也不精确建立软件可靠性模型。从这些模型出发很难建立作为依赖时间的度量标准的可靠性和作为静态测量标准的错误密度之间的关系。

当前大多数软件可靠性模型属于评价类。相关文献中也给出了几种早期预测模型,本书第5章给出了现有的评价模型。

1.2.6 其他术语

其他软件可靠性工程术语在本书的各章第一次出现时将详细介绍。以下文献也给出了标准词汇表和辅助信息:

(1) Software Engineering Standards Subcommittee of the Technical Committee on Software Engineering of the IEEE Computer Society. IEEE Standard Dictionary of Measures to Produce Reliable Software (IEEE Std 982.1 - 1988). IEEE Computer Society, 1988:1 - 40.

(2) Software Engineering Technical Subcommittee of the IEEE Computer Society. IEEE guide for the use of IEEE standard dictionary of measures to produce reliable software. (IEEE Std 982.2 - 1988). IEEE Computer Society, 1988:1 - 139.

(3) Standards Coordinating Committee of the Computer Society of the IEEE. IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12 - 1990). IEEE Computer Society, 1990:1 - 83.

1.2.7 实例

下面给出软件可靠性对系统可靠性影响的一个例子。

例 1.1 某军用分布式处理系统需要 100 h 的 MTTF 和 0.99 的可用率。图 1.3 给出了系统的总体结构，系统包括 3 个子系统(SYS1，SYS2 和 SYS3)、1 个局域网和 1 个 10 kW 的发电机 GEN。为使系统运行，所有的部分都得正常工作。在系统测试早期，可根据军用可靠性手册 Mil - 217 来预测硬件可靠性，且给出每个组件的可靠性。在图 1.3 的每个组件块上有两个数字，上面的数字代表该组件的 MTTF，下面的数字代表它的 MTTR，单位为 h。例如，SYS1 的 MTTF 为 280 h，MTTR 为 0.53 h，而 SYS2 和 SYS3 的 MTTF 为 387 h，MTTR 为 0.5 h。注意到在虚线框中的 SYS2 被构造成一个三模块冗余系统，只要两个以上的模块能正常工作，该子系统就能正常工作。由于具有了容错能力，其 MTTF 改进为 5.01×10^4 h，而 MTTR 变为 0.25 h。

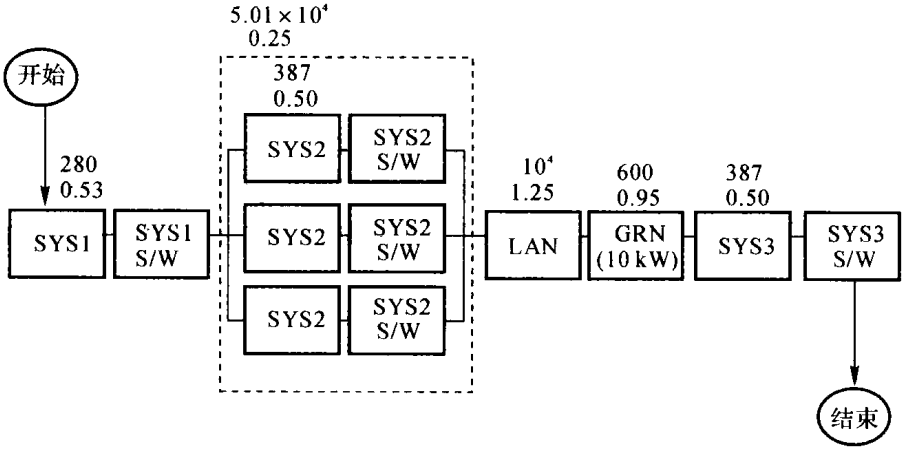

图 1.3 预测系统可靠性实例