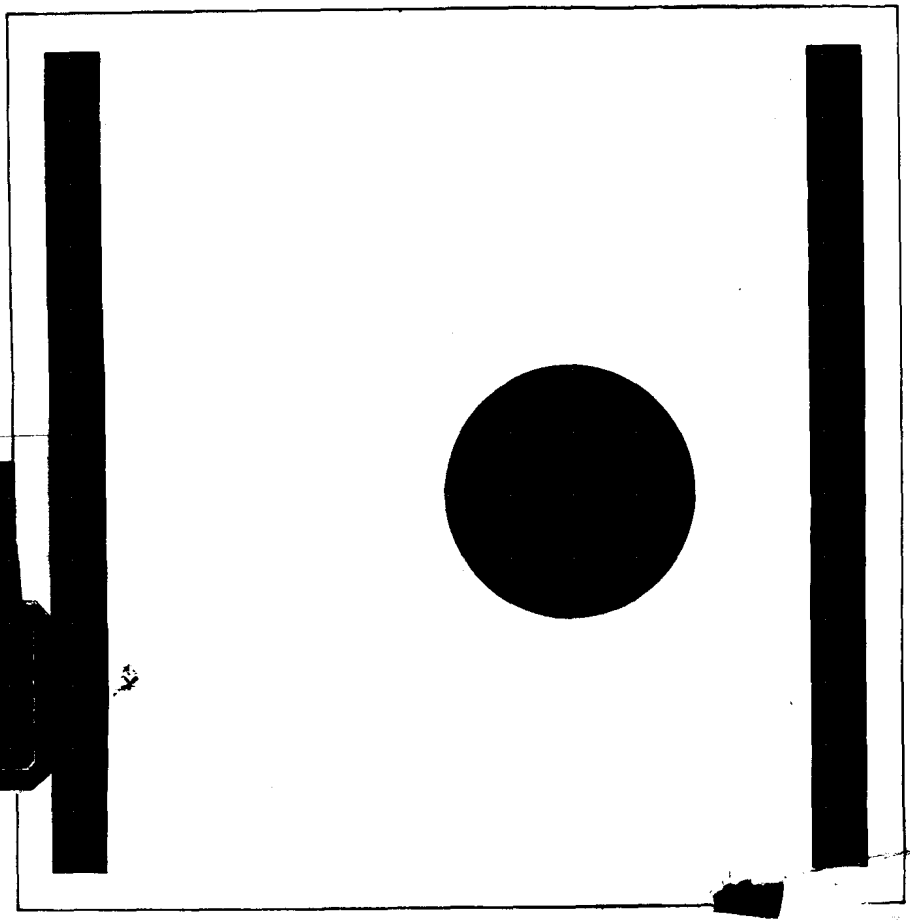
可靠性与质量管理丛书

计算机软件的可靠性

郦　萌　编著

KEKAOXING YU ZHILIANG GUANLI CONGSHU



计算机软件的可靠性

郭 菊 娟 编著

国防工业出版社

内 容 介 绍

本书是国内公开出版的第一本论述计算机软件可靠性方面的著作。介绍了软件的寿命周期(生存期)、失效及可靠性的概念，软件生产的阶段及各阶段的要求，为提高软件可靠性的软件设计技术、软件正确性的检验技术及维护技术，软件的容错设计技术，软件可靠性的模型及评估等方法。在计算机应用愈来愈广的今天，软件的编制及使用维护人员都应具有软件可靠性的基本知识，才能编出可靠的计算机软件，进行高可靠软件的生产及有效的维护使用。本书是软件开发、设计、使用人员的参考书，也可供大专院校师生作为软件教学的教科书或参考书。

可靠性与质量管理丛书
计算机软件的可靠性
郭 萌 编著

国防工业出版社出版发行
(北京市车公庄西郊老虎庙七号)
新华书店经售
国防工业出版社印刷厂印刷

850×1168 1/32 印张 6 000千字
1988年12月第一版 1988年12月第一次印刷 印数：0,001—3,280册

ISBN 7-118-00193-7/F·9 定价：3.00元

出版说明

“企业的技术开发工作要特别重视新产品试制、中间试验、生产性试验以及解决工业化生产中的质量、可靠性、经济性、成品率等一系列工艺和设备问题”(《中共中央关于科学技术体制改革的决定》1985年3月13日)。

产品的可靠性是产品质量的重要组成部分。质量与可靠性是国内及国外市场竞争中能否胜利的决定因素。质量及可靠性要从产品的设计、试制抓起，把质量与可靠性设计到产品中去，贯彻在从试制、试验、生产到使用的各个阶段中。提高产品的质量和可靠性将给研制生产单位带来巨大的经济效益，也带来巨大的社会效益。鉴于国内还没有全面地从各个专题方面介绍可靠性及质量管理方面的著作，本社特委托何国伟同志组织国内有关方面的专家，编著了这套《可靠性与质量管理丛书》，并由何国伟同志担任本丛书的主编。本丛书的目的在于培养可靠性工程和全面质量管理的专业人材，以保障和提高产品质量。

本丛书适用于大专以上的工程技术人员和管理人员，主要用于在职教育，也可供大专院校教学参考。

目　录

第一章 引 言

软件可靠性在可靠性研究领域中还是一个新的课题。这是因为可靠性是一门专门研究产品“失效”规律的学问，“软件是否也会发生硬件那样的失效？”，“对软件来说，失效究竟是一个什么含义？”等等，这些问题在软件作为一个独立产品出现后的很长一段时间里理解仍是不一致的。此外，计算机问世的头若干年中，硬件投资占着整个系统的主要部分，当时硬件可靠性不高，计算机工程师总是为硬件的频繁失效所“折磨”，根本无暇顾及软件的可靠性。事实上，当时的软件，功能上远不如现在那样丰富，结构上也不像现在那样复杂。因此软件可靠性的问题并没有引起人们的注意。

随着社会信息化的进展，处在其核心地位的计算机系统的可靠性变得越来越重要。首先，微电子科学和技术的进展出现了大量轻、薄、小而省能的硬件产品，使得计算机结构愈来愈灵活和精巧，其应用范围也由此而更广泛。尤其令人注目的是硬件的可靠性有了惊人的提高，这就使软件质量和可靠性的问题逐渐突出起来。

譬如，在工业生产的自动化方面，过去用分立元器件或中、小规模集成电路来构成一些专用装置（或系统）。一个装置的功能是十分有限的，因为这类装置大多只是为某一特定的对象而设计的，一旦设计、试制完成之后，通过具体对象的实际调试证明是可行的话，那末，只要元器件不发生失效，系统就能一直正常工作下去。可以说，整个系统的可靠性就取决于这些器件（硬件）的可靠性。由于微处理机的大量应市，使得上述专用系统越来越多地被体积小、功能强的单板机、微机系统所代替。很多智能仪表也都使用这些被称为电脑的微处理器来增强仪表的功能和提高

精度。这样一来，影响一个装置可靠性的，除了硬件之外，还有作为系统一部份的计算机本身系统软件（指令、监控、操作系统等）和为使该通用计算机完成特定使命而设计的接口硬件和应用软件。需要指出的是，前后两部分软件的设计人员之间是没有必然联系的。计算机设计人员在设计该机的当时，无法预见该机将在哪些领域使用。而应用软件的设计者往往只能通过一些手册来学习该机有关系统软件的知识，不可能透彻地领会机器设计者的匠心。这就增加了发生软件设计差错的机会。

任何软件，当程序不太复杂时，设计者或调试人员可以通过对程序的检查、验证来找到错误并加以纠正，或用数学手段来证明程序的正确。而在进行功能很多、程序十分复杂的软件设计时，要求不发生差错或把错误的数目限制在极低的限额以下，如果没有一整套提高软件可靠性的技术和工具的话，是很难做到的。

其次，尽管在我国还没有正式承认软件的专利，但大家都承认软件是一种具有很高价值的脑力劳动产品。软件产品和其他产品一样可以单独销售。并已经形成了一些专门从事软件制造的产业。而软件出现之初的情况却不是这样。当时软件的设计没有明确的分工。设计、实现、调试、检查以及维护常常都由一个人来进行。有关软件的知识也是只在少数人的范围中零星地、“师徒相承”地传授。一旦某一软件的设计者离职或去世，他所制作的程序就再也无法维护了。这种局面显然是和计算机应该广泛地为人们服务的目标相背谬的。到近代，软件生产已逐渐走上工业化、社会化、商品化的途径。软件知识的传播，软件制作方式和软件技术发展大开生面。需要一系列关于软件性能的评价指标，用以衡量一个软件的好坏和协调软件买方（用户）和卖方(厂商)之间的利益。软件的可靠性指标是其中很重要的一个评价基准。

以上是从软件发展的两个方面来说明软件可靠性技术和可靠性评价研究的意义。事实上仅以 1980 年时的统计来看，在引起计算机不能正常运行的原因中，属于软件错误的占了相当大的比

重。因为在计算机中，软件和硬件是协同工作、共同来完成系统要求功能的。从发展趋势看，早期的批处理发展到联机系统，又发展到实时系统、分时系统，软件在充分调动计算机潜力和保持协调工作方面，承担越来越关键的作用。相应地说，系统研制成本中软件所占的比重也会越来越高。一个系统，为研制其软件而花费的代价，可以远大于购买硬件的花费。而且即使花了可观的代价，制作好的软件中仍然可能包含某些差错。有的虽然没有差错，但由于没有充分考虑扩充和变更的问题而存在着局限性。这样的软件在维护上的开销是很大的。对复杂程序进行维护又可能引入新的错误。这样一来，据近年统计，用于软件维护的花费平均达到系统总花费的70%左右。它既说明了软件可靠性本身的重要性，也说明在软件设计中可靠性作为一种“信念”必须贯穿从开发、设计、制造、试验、检查、维护直到废弃这样一个关于软件生命的全过程。在有的文献中，描述可靠性和软件生命各阶段的关系是：把“可靠性种子”播种到软件生产的各个阶段，通过精心培育以期收获可靠性的果实。更重要的是关于软件可靠性的信念必须在软件人材训练的最初阶段就应注意加以培植。特别要使未来的软件专家明白，高可靠软件的取得，只有依靠个人的勤奋和群体的协力，任何惰怠，粗枝大叶，刚愎自用都是和成功的愿望背道而驰的。

人们一般认为，硬件失效的机理比较清楚，而软件却不是这样。因此在讨论软件可靠性之前，分析对比一下两者的失效机理是非常有益的。

我们把失效机理中导致失效的最关键因素叫做致命因子，那末，我们先考察一下硬件失效中致命因子存在的形式和分类。

硬件失效模式甚多，但大体可纳为两类，寿命失效和随机失效。寿命失效机理主要是磨损和耗散。前者如各种运动机件、电气触头的机械摩擦、疲劳导致失效，后者如电池、真空管阴极则是由于化学能耗尽、涂层蒸发而失效。其致命因子是在每一个合格产品经过一段使用期后必然产生的。而有些产品的失效，如电

容器、变压器的绝缘击穿，半导体 pn 结的破坏等，如果设计是正确的、材料是合格的、制造工艺是理想的，且又在额定的环境中工作的话，理论上说，失效是永远不会发生的。这些产品之所以失效，原因有两种，一是产品出厂时就存有内在的致命因子。如绝缘材料的杂质，电极或导线上的金属毛刺，硅片上晶格错位以及封装漏泄等。有这些缺陷的产品在考虑了足够保险系数的环境下工作时，和完美的产品很难区别，因此它们能够通过常规检验被当作合格产品出厂。而当环境发生波动时，这些产品就会失效。第二种是合格产品由于受到超过其可能承受的应力（包括机械的、电气的在内的各种应力）而失效。这两种情况下的失效都是随机的。软件没有磨耗问题。其失效模式接近于上述存在有内在致命因子的产品的失效模式。

在讨论可靠性时，“失效”与“未失效”之间的界限必须是十分明确的。对于某些产品，其状态的劣化即使是连续的，也一定要订一个适当的功能标准来确认某产品是否失效。产品丧失规定的功能就认为失效。近年来，不少文章提出“允许功能降级”的可靠性评价模型或称多阶段失效模型的研究。这些模型实质上可以理解为是用一个产品去同时或逐一承担功能要求不同的多个系统的任务，当它丧失一方面功能时，对于有这方面功能要求或功能要求高于这个标准的系统来说，这个产品已经失效，而对其余系统来说则是仍然有效的。对于计算机这样一个功能十分丰富的系统来说，失效的多义性是一个必然出现而又比较麻烦的问题。

另外，失效往往又是多层次的。大到一个系统，它是由分系统、模块、单元逐层构成。如果失效只是产生在某一层次，即仅由于某一层次功能的丧失而导致系统功能丧失，那末，只要及时采取修复、更换等手段是可以使系统维持原有功能的。小到一个元件，也可以将其分解为更小的各个部分，如一个晶体管可以分为引脚、管座、芯片等。由于失效的多层次性，泛泛地谈“失效”就会引起误解。本书中，把带机理性的，构成致命因子的，层次

较低的失效不论硬件软件均称为“故障”(Fault)。能够感知的，中间层次的称为“错误”(Error)。在软件领域中，以往多把设计中的失误称为“差错”(Bug)。通常“错误”是指结果，而“差错”是指起因，是合乎一般人理解习惯的。但本书为了统一起见，把软件设计中的差错也称为软件故障。通常低一层次的失效（错误、故障）就是高一层次失效的起因。高一层次的失效（或出错）则是低一层次故障（或出错）的结果。

硬件设计中无疑也会有差错或失误，但很少人把设计中的差错看成是一个硬件故障，而认为硬件设计制造中的差错都应该在投入运行之前的调试阶段发现和改正。硬件的可靠性、计算机容错技术都没有考虑设计差错的因素。软件则不同。多数软件，特别是功能多、通用性强的软件，要实现在所有可能的使用环境下来调试是不可能的。越是大型的软件，因设计差错导致失效的情况就越多。这种软件故障是在软件投入运行之时就存在的。

在软件的调试阶段，可以发现一些设计差错并加以改正。因此，在这个阶段软件失效率是递减的。当软件中所包含的设计差错的数目已经变得相当小时，往往在一般的调试环境下很难发现这些差错。这样的软件往往就被交给用户使用了。在实际使用环境中，在系统处于某些特定的状态和某些特定的输入相组合，形成一个特定的环境。这时，这些残存的差错就会暴露出来，造成错误。这时往往担心修改会引入更多的差错，而不轻易进行修改。由于这种特定环境的出现有一定偶然性，所以投入运用后软件的失效率是很低的，可以视为常数。软件在调试阶段和运用后一段时期内，其失效率变化趋势和硬件是相似的。也就如通常所说“浴盆曲线”的第一和第二段所描述的那样。“浴盆曲线”的第三阶段，从硬件来说，主要反映在机件的磨耗、疲劳、腐蚀、耗尽等等。软件显然不存在这类致命因子。但应当指出软件致命因子中除了设计差错之外，以下几方面的因素还是应当加以考虑的。

第一，在新的同类软件生产出来之后，经用户使用和比较，

证明原来的软件相形见拙，而且随着新软件被更多人所熟悉、接受，旧软件已渐渐不再为用户所欢迎时；

第二，用户普遍提出了新的要求，原有软件已明显地无法满足时；

第三，由于硬件技术发展，出现具有新功能的硬件，而原有软件已难以与之匹配时。

新的软件总会不断地被生产出来，用户的新要求也总在不断被提出，硬件技术和工艺的发展也无止境，因此，即使没有设计差错或差错很少的软件，到了这种时候，也会经常出现满足不了用户要求的局面。在有的文献中，把处于这种状态的软件称为陈旧了的软件。不断进行版本的改进可以推迟陈旧，但改版有一定的限度，往往由于费用的关系，不得不整个地废弃原有的陈旧了的软件系统。在软件领域中，历来对可靠性有是否“达到规定功能”和是否“满足用户要求”两种定义。前者是静止的，功能一经规定，不再变化，是具体的。后者中的“用户要求”一词不仅笼统，而且意味着将随时代的变化而变化。显然在后一种定义中，用户要求不能得到满足的事件也被作为一种失效。这时的失效率是随时间递增的，和浴盆曲线的第三阶段相吻合。

既然软件失效和硬件失效在规律上有一定的相似之处，硬件可靠性分析的理论和方法大体上也可以用于分析和研究软件可靠性。也可以参考硬件中提高可靠性的手段来提高软件的可靠性。但是，由于硬件和软件两者关于什么是“一件产品”有根本不同的两种理解，所以在软件可靠性研究中，以下几点务请注意。

第一，通常所说的硬件产品指经设计制造出的一件具有一定使用价值的物品。按照同一设计、同一工艺可以制造出许多件有相同使用价值的产品。而软件的一个版本只是一件产品，版本的复制并不产生新的产品。

第二，系统中一个硬件单元的失效，可以用另一个相同单元来替换，使系统恢复正常。而软件系统中，如果一个程序模块中包含着设计差错，从而导致系统失效，显然是不能用相同的程序

模块来替代使系统恢复正常的。

第三，在利用冗余方法来提高软件可靠性时也同样要考虑这个问题。在软件中，冗余不仅不是相同的程序或程序段在同一空间运行多次或在不同空间同时作多重运行，也不应是在系统分析、设计、实现和检验阶段中相互有牵连的那些程序作上述运行。而必须是一些完全相互独立制作的程序，这和硬件的情况不同。只要满足这一要求，也就可以利用冗余方法来提高软件的可靠性。以这种方法构成的系统称为容错软件。在现代容错（Fault-Tolerant）计算技术的研究中，还有一类是利用软件的手段来提高整体可靠性的技术。如通过专门的软件来诊断、屏蔽硬件的失效等等。为了避免混淆，本书中把这类技术称为软件容错，指其系用软件来消除因硬件失效引起的错误。所谓容错数据结构、容错数据库等，本质上说都属这一类。

最后，是关于软件可靠性的定量分析问题。软件差错产生是随机性的这点，只须定性的论述便可以理解。因为人为地制造错误、或者连经常用到的软件知识也没有掌握的设计者应该是极少数。产生差错的原因，主观上说，可能是由于设计人对偶尔出现的细节问题不明了。特别是对于从自然语言出发，可能产生理解上二义性的地方（例如计算机理解的“0”与设计者理解的“0”可能不一致）。经验丰富的人，不明了的部分少一些，平均差错也会少一些。另外客观环境对软件差错也有相当大的影响。影响设计人思考的因素很多，它包括个人心情好坏、健康状况、疲劳程度等等随机出现的因素。因客观原因引起的差错，包括通常所说的“疏忽”，对于人来说，在一定程度上是不可避免的。人的思维可以设想为是由人脑这台计算机的紧张工作来进行的。这在机理上和机器的失效是可比拟的。不同在于我们需要估计的不是人脑本身的可靠性，而是人脑所设计的产品——软件的可靠性。人脑的失灵（或疏忽）和软件失效两者虽有联系，但不相同。疏忽可以造成设计差错，有差错的软件只有在一定的输入环境才会形成软件运行的错误。这里介入了一个软件的输入环境。反过

来，只有当软件的运行结果出现错误时，人们才感知软件设计的差错和分析出疏忽之所在。因此，软件可靠性的评估，除了可以参考一般模型之外，还包括有自身的内容。

本书是按《可靠性与质量管理丛书》的要求来编写的，因此不过多介绍一般可靠性的基础知识。而它既然以软件可靠性为题，也就不必重复通常的软件的设计、调试的知识。本书围绕软件可靠性的概念、技术、工具、容错软件和软件可靠性评估等问题作一概略介绍，其目的在于以下两方面：

第一，在向信息社会迈进中，随着OA(办公自动化)、DA(设计自动化)、CAD(计算机辅助设计或诊断)、CAM(计算机辅助生产)、CARP(计算机辅助可靠性程序)等的广泛应用。传统技术领域中大量引入计算机技术，专业人员必将接触、使用、维护甚至设计软件。了解软件可靠性的知识，能使他们对保证软件质量从一开始就加以重视，形成良好的工作制度、采取正确的工作方法，以取得事半功倍的效果。

第二，对于软件专业人员来说，则希望强调软件的产业化生产和质量管理，注重协作精神，更多地听取用户的要求和注意了解其他技术领域的进展，使软件产业更加成熟和兴旺起来。

由于软件可靠性课题内容较新，有一些观点还在争鸣之中。本书对有关内容大多采用兼收并蓄的方针，以便读者了解这些观点和意见。有的内容限于篇幅只能作概要介绍。但本书各章均列出详细的文献目录，以便要求继续深入探索的同行作更进一步的研究。挂一漏万以及谬误之处自当不免，望读者指正。

第二章　软件生存期、寿命、失效及可靠性

2.1 概　述

一切有生命的东西都有一个“寿命”，即它们从出生到死亡为止所持续的时间。一个国家的人民健康水平往往就用国民的平均寿命作为主要的衡量指标之一。寿命的概念也延伸到对无生命产品的质量估价。例如，常常提到的机械产品、电子产品的寿命等，就是指产品从出厂直到丧失其使用价值为止的持续时间。而软件本身是一个直接思维产品（或称精神产品），它不存在物理上的磨损、化学上的耗尽和生理上的衰竭。因此软件“寿命”的含义，从本质上讲和生物及硬件产品不同。

从软件工程角度来说，软件寿命也可以理解为软件的寿命周期或生存期（*Life Cycle*）。这个生存期包括：要求/规格说明（*Requirements/Specifications*）、设计（*Design*）、实现（*Implementation*）、检验（*Checkout*）和维护（*Maintenence*）等五个阶段，如图 2-1 所示。

前四个阶段也称为软件的开发期。软件的维护则是在使用期进行的。

早期的一些文献中，曾把软件生存期划分为三个阶段，就是：“基本思想和要求”、“研制”及“运行和维护”。目前也有将其划分为“要求分析”、“拟定说明”、“设计”、“编码”、“调试”、“运行和维护”等六个阶段。不同的划分和命名并不影响我们对软件寿命周期或生存期的理解。总之，从软件任务的提出一直到它完成使命，因陈旧而被废弃为止，代表着一个软件历世的全过程。

但是，从用户的角度来说，更关心的是软件在交付使用后的

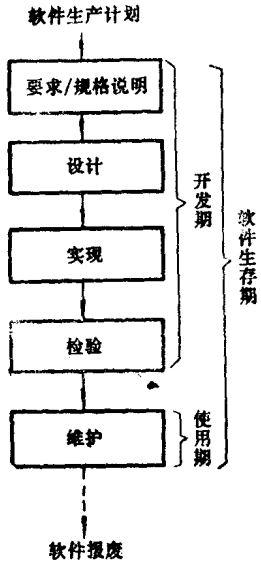


图2-1 软件生存期的五个阶段

情况如何。也就是需要一个从可靠性角度来理解的“寿命”。说得更明确一些，就是希望有一个类似于硬件中平均失效间隔时间（MTBF）这样的指标来表明在规定的要求和条件下，能在多大的程度上依赖这个软件来完成任务。不过软件的失效和输入实例（也称输入环境）有关，如一个存在缺陷的编译程序当用于学生做简单练习时，MTBF可能很长。而做一个大课题时，则MTBF可能由于连续出错而减小。所以即使是一个大的MTBF只能解释为对可靠性的一个可能的样本数据，而不能作为一个依据。

为了清楚起见，有必要对以上所述的寿命概念作一区分。在本书中，我们把从软件研制角度来理解的寿命概念称为软件生存期或软件寿命，它由图2-1所示的五个阶段组成。而把使用期软件能正常工作的持续时间称为软件使用寿命。作这种区分后，显然有助于理解以下几个问题。

（1） 软件生存期包括前后二个时期：开发期（要求/规格