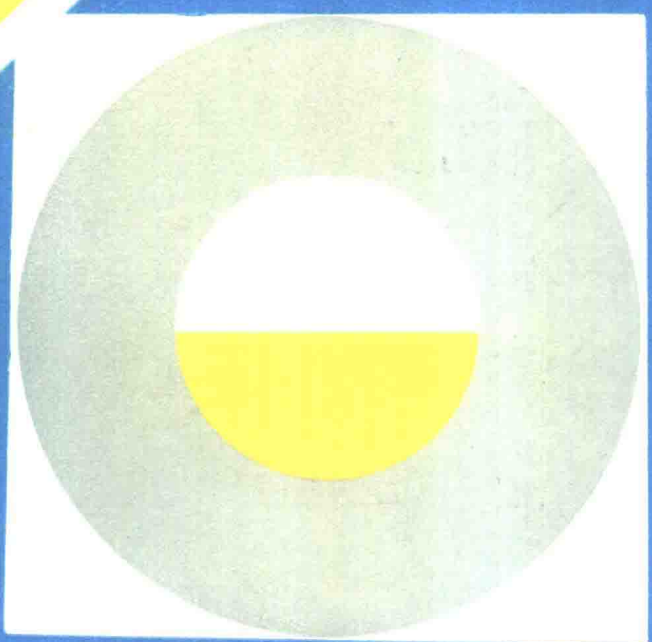
软件的可靠性与安全性

黄锡滋 编著



科学出版社

软件的可靠性与安全性

黄锡滋 编著

国家自然科学基金资助项目

科学出版社

1993

(京)新登字 092 号

内 容 简 介

软件可靠性与安全性是当代的科技前沿课题之一。本书融软件工程学与可靠性工程学于一体，系统地总结了国内外软件可靠性与安全性的主要研究成果，包括作者本人在这个领域内的一些工作。内容涉及软件可靠性的基本理论，可靠软件的设计，可靠软件的测试，软件的可靠性预计及可靠性增长分析，软件的可靠性分配，软件的可靠性与安全性分析，软件的质量保证。

本书适合于从事系统可靠性工程、质量管理、软件工程的高级和中级科技人员阅读，也可作高等院校中上述专业的学生和研究生的教材或教学参考书。本书对所有与软件开发、软件维护有关的科技工作者、管理人员都具有参考价值。

软件的可靠性与安全性

黄锡滋 编著

责任编辑 唐正必

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1993 年 10 月第 一 版 开本：787×1092 1/16

1993 年 10 月第一次印刷 印张：14

印数：1—2 000 字数：320 000

ISBN 7-03-003480-5/TN · 142

定价：13.20 元

前　言

80年代以来，我国国民经济和科学技术获得高速发展，在高科技领域，计算机的推广和应用尤其引人注目。各部门，各企事业单位，从国外引进和自行开发了大量的计算机软件，这些软件为促进我国的经济建设、国防建设和科技进步起了重大的作用。

然而世间万物，皆无完美无缺者，计算机软件也不例外。早在60年代末和70年代初，发达国家就饱受不可靠软件的折磨和伤害之苦。在这样的背景下，软件可靠性的研究应运而生。我国是一个发展中国家，近年来对软件的开发和研究取得了很大的成绩，然而对软件的可靠性和安全性的研究尚未给予应有的重视。例如，我国现役的军事装备中有许多是软、硬件构成的复合系统或子系统，这些装备在投入使用前大多经过了可靠性评定。但是，在可靠性评定中并未包括对软件可靠性的定量评定，显然，这种评定的结论只能具有相对的可信度。又如，那时我国高等院校理工类专业的毕业生，他们在高校学习期间，根本就没听说过何谓软件可靠性。他们在社会实践中开发高可靠产品时，首先遇到的便是软件可靠性问题。这种现状必须改变。1986年，我在电子科技大学（当时的成都电讯工程学院）为研究生开设了软件可靠性课程，随后又在课程中增添了与软件可靠性密切相关的软件安全性方面的内容。手上的讲稿逐年增厚，于是又萌发了编著本书的想法。时值我承担了国家自然科学基金软件可靠性方面的研究项目，这个想法得到了国家自然科学基金委员会电子学与信息系统学科张志健主任和科学出版社的大力支持，于是这本书才得以呈献给读者。

本书基本上包括了目前国际上软件可靠性及安全性领域的主要成果，其中也包括了作者本人在国家自然科学基金委员会和电子科技大学支持下所做的一些工作成果。内容涉及软件可靠性的理论及各个工作环节，也有为掌握软件可靠性所必需的、最低限度的软件工程的相关内容。本书适宜于从事系统可靠性工程、软件工程的中级和高级科技人员阅读，也可作为上述专业的教学参考书。

熟悉可靠性工程的读者读完本书后将会发现，和硬件可靠性工程学相比，若干环节的内容仍显得单薄。其实这一点恰恰是软件可靠性学科和硬件可靠性工程学差距的反映。我们相信，经过国内外可靠性工程界、软件工程界的共同努力，一个能与硬件可靠性工程学媲美的软件可靠性工程体系，将在21世纪初期建立。

北方交通大学叶杭教授、航空航天部611所王维翰高级工程师对本书的内容提出了许多宝贵的意见，谨在此表示感谢。

目　　录

第一章 绪 论

1.1 软件可靠性的重要性

可靠性工程学的诞生和发展，从 1957 年 7 月美国国防部电子设备顾问团（AGREE）的研究报告发表时算起，迄今已有三十多年．在这三十多年中，可靠性工程学建立了一套完整的理论体系，开发了许多适合工程需要的应用技术和管理方法，为当代高新技术的发展提供了有力的保证．同时，可靠性工程学的发展已跨越了国界，成为人类共同享有的财富．

我国可靠性工程学的研究和应用开始于 70 年代初，在 80 年代获得了很大的发展．现在，在电子、航空、航天、核能等许多领域中，可靠性工程日趋普及．然而无需讳言，在可靠性工程学这个领域内，我国和发达国家相比，仍然存在着很大的差距．这个差距首先表现在为数众多的产品的可靠性水平低于国际水平．对于这一点，我国可靠性工程界已清醒地认识到，并正在竭尽全力缩小这一差距．但是，另一方面的差距则往往为人们所忽视，那就是在新技术开发上的差距．有关这一问题的深入讨论，不是本书的主旨．但是，需要着重指出的是，我国对软件可靠性研究的忽视，则是这种差距的一个重要例证．

计算机的出现和应用是 20 世纪的一项杰出的科学成就．现在，计算机技术已渗透到人类社会的各个领域，并正在推动人类向信息时代迈进．软件是计算机的神经中枢．如果没有软件的指挥、控制和协调，计算机与常规的电子设备就没有什么差别．没有可靠的软件，即使是最先进的计算机，也无法保证正常发挥其功能，由此可见软件可靠性如此重要．因此我们在致力于改善硬件可靠性的同时，必须认真研究解决软件的可靠性问题．下面我们来介绍一下不可靠软件对人类社会造成的危害．

（1）导致系统设备失效，危及人身设备安全．在本世纪 40 年代计算机诞生后的很长一段时间内，国外对于软件的可靠性问题只有极少数人具有朦胧的警觉．到了 60 年代，随着计算机性能的改善，应用领域的日益扩大，以及对系统软件和应用软件需求的激增，人们才仓促地进入了设计复杂软件及软件系统的新阶段，但时过不久，人们就惊讶地发现，软件已不再是一个唯命是从的仆人，它已经转变为一个有潜在破坏力的对手．

飞向宇宙空间，探索宇宙空间的奥秘，是全人类共同的理想和追求，现代航天技术则是人类实现这个伟大理想的技术工具．各种航天器耗资巨大，而且，由于宇宙飞船和航天飞机要搭载宇航员在茫茫无边的宇宙空间遨游，任何微小的故障都可能导致严重的损失，甚至酿成重大的悲剧．因此，对于宇宙飞船和航天飞机所使用的计算机软件，人们都竭尽全力进行设计和测试．虽然如此，人们在这个重要领域仍饱尝了辛酸．60 年代中期，美国的首次金星探测计划，就因为在用 FORTRAN 语言编写的 DO 语句中漏掉了一个逗号而惨遭失败．在随后的阿波罗计划的实施中，阿波罗 8 号宇宙飞船的仓载计算机因为软件错误而丢失了部分寄存的数据．在阿波罗 14 号宇宙飞船 10 天的航行过程中又发现了 18 个软件错误．1973 年，美国空军杂志在一篇题为“计算机——明天空军的关键”的

文章中对此有这样一段评论："为阿波罗登月飞行而制定的软件开发计划，在当代可称耗资最大，且安排又最为精心。这项工程吸引了许多全国最卓越的程序设计师，组成了两支相互竞争的设计队伍。专家们对软件的检查倾注了全部专业知识和技能，同时将总计高达 66 亿美元的巨额资金用于阿波罗计划的软件开发。然而，阿波罗计划实施中的每一个重要的故障，从误报警直至真正的灾难，几乎都是计算机软件直接造成的。"进入 80 年代后，情况仍然没有根本的改变，原计划于 1981 年 10 月首次发射的航天飞机，在发射前 20 分钟突然发现了一个软件错误，致使发射被迫推迟。

在类似的大型工程项目中，不可靠软件所造成的严重后果已经显而易见。在人们的社会生活中，不可靠软件所扮演的角色，颇似惯于恶作剧的小精灵，被它作弄的后果虽算不上十分严重，却足以使当事人心烦意乱、无可奈何。例如，一个程序员将公司的 IBM 1401 系统的工资管理程序转换到 IBM360 系统使用后，发现自己工资条上的工资减少为 0.00 美元。又如有一个用计算机管理的冷冻仓库，由于数据库中的一个错误，使得 15 吨的冻肉处于常温状态而腐败变质，等等。

据报道，贝尔实验室设计的一个大型自动换接系统，其软件都是经过精心计算和调试的，然而在该系统所发生的故障中，经分析有 20% 是由于软件错误所造成的。美国空军的一个指挥和控制系统，软件指令总数为 4.5×10^5 条，在 1000 小时运行时间内，软件共发生过 25 次关键性失效。单位指令的失效率为 5×10^{-8}/h (CPU)，大约相当于或略低于电子元器件的失效率。然而，由于考虑到系统中指令数都大大超过了系统的元器件数(总指令数超过 100 万甚至 1000 万的大型软件系统在目前已非罕见)，因此，这种程度的失效率，显然是一个不容忽视的严重问题。

(2) 导致严重的经济损失。不可靠软件会导致系统失效并造成经济损失，这是明显的事实，无需赘述。然而不可靠软件在开发和维护过程中所造成的经济损失却鲜为人知，这种损失可以从两方面来估算：第一是由于开发的软件不符合可靠性和质量的基本要求，无法使用而造成的损失。第二是由于不可靠软件在交付使用后出现错误，不得不进行维护和改进而造成的损失。关于第一种类型的损失，在美国 ALABALA 大学及 MARSHALL 空间飞行中心的一次调查报告中是这样说的："在调查所涉及的 680 万美元的软件开发费用中，占总数 95% 的 650 万美元被浪费了，其中，47% 的软件已交付给用户，但从未使用过；有 20% 虽支付了开发费用，但无法交付用户使用；有 19% 因无法使用而放弃或重新进行设计；在其余的 5% 中，只有小于 2% 的费用所开发的软件交付使用后在真正发挥作用"。这个统计数据确实令人吃惊。

关于第二种类型的损失，需要从计算机的开发过程来考察，进入 80 年代后期，计算机系统的开发费用的变化出现两个显著的特点。第一个特点是，在系统的开发费用中，软件开发费用大大超过硬件的开发费用。这个现象的出现是由于技术和工艺的改进，使得计算机硬件费用显著下降，以及由于软件功能强化导致结构日趋复杂所致。图 1.1 是计算机系统开发费用中，硬件与软件所占比例变化的趋势图。

第二个特点是在软件生存期的费用中，维修费用上升并占有很大的比例。图 1.2 是软件生存期费用中各阶段费用所占比例的示意图。由图可见，软件的维修费用占了 50% 的份额。在软件的维护费用中，又以改正软件中的错误和改换版本的费用居多。改换版本虽然包括了扩充功能的因素，但也含有改正错误的因素。

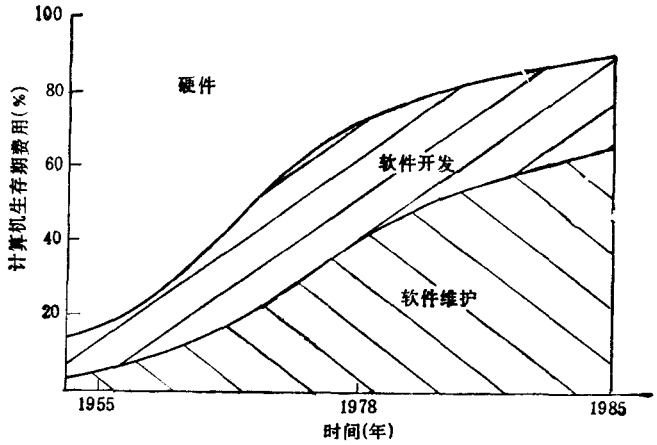


图 1.1　硬件与软件费用比例变化趋势示意图

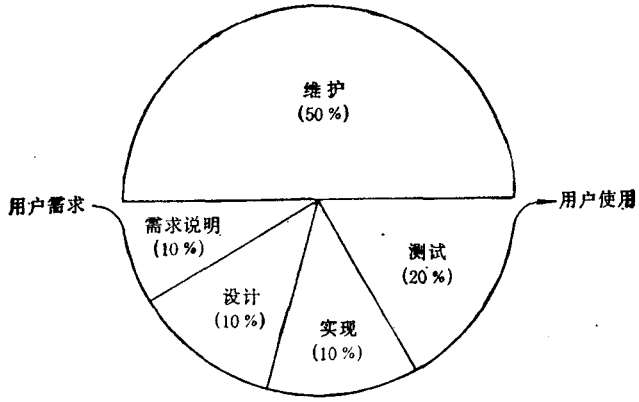


图 1.2　软件生存期费用比例示意图

如果人们在软件开发过程中对软件可靠性多给予几分关注，则上述两种类型费用的损失都可以减少．因此，软件项目开发的决策层必须意识到，改善软件可靠性虽然需要投入相当的经费，但这种早期的投入是明智的和必要的，因为它可以带来成倍的或数倍的经济效益．

1.2　软件可靠性发展史

软件可靠性的发展是与软件工程的发展紧密相联的，因此需要根据软件工程的发展过程来了解软件可靠性的发展史．软件工程的发展约可分为下列四个阶段：

(1) 1950—1958 年．在这段时间里，计算机几乎完全用于科学和工程计算．计算机编程语言只有机器语言和稍晚些时候出现的简单的汇编语言．那时没有专职的程序员，程序是由应用计算机的科学家和工程师自行编制的，没有公认的规则可供遵循．他们集科学家(工程师)、程序员和用户的职能于一身，因此，编制程序成了物理学家、数学家、机械师及各专业工程师的一项高超的智力活动．经过这些业余程序员的精雕细刻，确实编制出了一些非常高明的佳品，但是这些程序一般仅供个人一次性使用，除非编者本人确有

需要才再次使用．在软件发展过程的这个原始阶段，完全没有软件可靠性的概念．美国著名的 AGREE 报告是公认的可靠性工程的奠基性文件，该报告集中了许多知名专家，经过反复探讨研究，历时 5 年，才于 1957 年正式发表．但是，正是由于上述原因，在这份重要的报告中，完全没有提及软件和软件可靠性的问题．

（2）1959—1967 年．在这段时间内，软件领域开始出现高级语言，应用也日趋广泛．操作系统开始应用于计算机，使得脱机操作得以实现．计算机的功能除单一的科学、工程计算之外又增添了一个重要的数据处理功能，从而使计算机的应用范围扩展至银行、商业、交通管理等领域．软件设计、编码、操作及系统分析等职能逐渐分离，并出现了相应的专业人员，用户也逐渐与设备和程序脱离．这个阶段在开发高级语言方面尤其受到了关注，FORTRAN，ALGOL 和 COBOL 语言相继出现，其目的在于减轻用户在编程上的困难．“虚拟机”的概念也随之形成．根据这种概念，计算机可以看作是被操作系统软件及其编程语言所定义的．

与这阶段计算机硬件和软件技术飞快发展形成强烈对比的是，软件可靠性问题仍然被人冷落，这种状况理所当然地很快就受到了来自软件的严重的报复．在那个时代 IBM 公司推出的 360 系列机是一项较为杰出的技术成就．它使用了诸如 FORTRAN，ALGOL，COBOL 等多种高级语言．然而，IBM360 系列的操作系统 OS360 的故障却连续不断地发生，最终使该系列机陷入困境．OS360 系统的开发依靠了一批优秀的专家，并投入了 5000—10000 人年工作量和每年平均 5000 万美元的费用，所以在遭受挫折后引起了有关方面非常强烈的反响．在计算机应用的其他领域，问题也大量地暴露出来．所以人们将这段时期称为软件危机时期．

（3）1968—1978 年．这个阶段以集成电路为主体的第三代电子计算机已经问世，各种小型机逐渐得到了广泛的应用．并开始配备上较为满意的系统软件，高级语言诸如 FORTRAN，ALGOL，COBOL 等也可直接在小型机上使用．

在硬件技术迅速发展的前提下，以及在软件危机的刺激和推动下，软件工程学得以建立和发展．软件的可靠性问题是软件工程学得以建立的一个强大推动力，也是软件工程界力图解决的一个重要目标．软件工程学的理论和技术为可靠软件的设计、测试和管理提供了指南和工具．但是，它没有能力解决在系统开发中，用户要求对软件可靠性作出定量评价的问题．于是，许多著名的软件工程专家开始致力于利用和改造硬件可靠性工程学的成果，使之移植到软件领域．由于他们的努力，迎来了软件可靠性工程学的开创时期．这个时期的特点是，以软件可靠性为主题的国际学术会议频频召开，吸引了各界人士的关注．软件可靠性的数学模型尤如雨后春笋般地大量涌现．著名的 JELINSKI-MORANDA 模型、SHOOMAN 模型、NELSEN 模型、MILLS-BASIN 模型都是在这个阶段推出的．此外，软件失效数据的积累和分析工作也有了初步发展．但是硬件技术的飞速发展，也给软件工程和软件可靠性带来了不容忽视的消极影响．小型机可以作为控制元件应用的优点，吸引了一大批电子工程师投入这个领域的开发，设计了大量的实时处理嵌入式软件．70 年代初期的小型机还缺乏合适的系统软件，不具备“虚拟机”的必要特征．但在这个领域却出现了众多的集工程开发、程序设计、使用于一身的非专业软件工作人员，造成了软件工程发展的后退趋势．

（4）1978 年至今．在这个阶段，由于大规模集成电路的出现，导致计算机向大型机

和微型机两个极端的方向发展，小型机逐渐被微型机所取代。这种两极发展的势头，对软件和软件工程学的发展产生了深刻的影响。在大型化的一端，各种类型的大型机都具有各种高级语言的编译程序。它的分时系统能力很强，一般可以配置上百个终端，同时还有批处理系统和实时系统的完善的通讯软件，可以沟通与一些卫星站之间的联系，并具有与其他计算机系统联网的能力。大型机系统的程序支持环境还能为其他应用软件的开发提供工具。庞大的应用软件为不同领域的用户提供了极大的方便。

在微型化的一端，由于机型小巧和环境适应性的改善，微型机现已进入办公室、教室、家庭和生产现场。由于性能的提高，微型机字长已达 16 位、32 位，内存可达 4 兆字节，还可配置几百兆字节的外存。因此，在 80 年代，微型机在生产、武器装备的自动控制和社会生活中，扮演了极为重要的角色。

在上述的各种场合，软件可靠性的重要作用是无容置疑的。软件失效所造成的影响，可波及社会生活的各个角落。因此，对软件可靠性的研究在这一阶段仍然保持了旺盛的势头：软件的设计、测试技术都有所提高，各种软件可靠性模型相继推出，重点为模型的验证和试用；软件可靠性管理技术的开发已列入日程；软件可靠性标准化工作已经起步。国际电工委员会的 TC56 技术委员会在 1985 年成立了软件可靠性工作组，并制定了软件可靠性和维修性管理规范(草案)。在 80 年代中期，软件安全性问题受到了特别的重视，硬件可靠性和安全性分析中所采用的故障树分析法、故障模式效应分析法和潜藏回路分析法，已在软件安全性分析中使用，并取得了令人鼓舞的成果。

在软件可靠性的研究和应用取得稳步进展的同时，小巧而价廉的微机得到更加广泛的使用，因而使在前一阶段软件工程发展所出现的负效应更趋严重。这是因为数十万甚至上百万的非软件专业人员在系统软件较贫乏的微机上，从事各项应用的开发。他们虽然拥有电工、电子、机械等专业知识，但是对软件工程和软件可靠性不甚了解，这种局面的后果是不难想象的。

综观软件可靠性 20 多年的发展过程，进展是明显的。然而就其发展水平与硬件可靠性工程相比，还有很大的差距，而且远不能满足计算机科学发展的需要。我们必须坚持不懈的从事开发和研究，促使它更快地发展。

参 考 文 献

[1] Myers, G. J., Software Reliability: Principles and Practices, John Wiley & Sons, New York, 1976.

[2] Koptz, H., Software Reliability, Macmillan, London, 1979.

[3] Lipow, M., *IEEE Trans.*, R-28, p. 178, 1979.

[4] Glass, R. L., Software Reliability Guidebook, Prentice-Hall, Englewood Cliff, 1979.

[5] Marco, A., and Buxton, J., The Craft of Sofrware Engineering, Addison Wesley, Workengham, 1987.

[6] Uber, J. G., Government Report N89-21754, USA, 1989.

第二章　软件质量及可靠性的基本概念

2.1　软件及软件工程

2.1.1　软件的定义

在当代社会生活中，硬件、软件这两个词汇屡屡出现，致使其内涵远远超出了计算机科学的范畴．为了明确本书的研究对象和研究范围，必须对这两个名词的含义加以说明．在本书中，软件一词是计算机软件的简称，硬件一词是指计算机系统中的主机及外围设备，有时则泛指所有具有物理构形的设备及元器件．例如，一个计算机控制系统，包括受控设备、传感器、电缆及计算机，这时计算机仅是系统中的一个硬件．

根据国标 GB/T-11457 的定义，软件是“计算机程序、过程、规则及与这些程序、过程、规则有关的文档，以及从属于计算机系统运行的数据”．国军标 GJB-437 则定义软件是“计算机的各种程序，相应的数据和文档的总称，并包括固件中的程序和数据”．这两个定义的内涵是一致的，但在 GJB-437 中提到了固件中的程序和数据，因而定义更加全面．有人常常把计算机用的磁盘、磁带称为“软件”，这种称呼比较含混．应该把软件和存储软件的物理介质区别开，存储程序和数据的有形物，如硬盘、软盘、卡片等，属于硬件范畴．

在软件的定义中还涉及到计算机程序、数据、文档、固件等术语，对这些术语我们还需要作进一步的解释．

计算机程序：指一系列可以被计算机设备所接受的指令或语句，这些指令或语句可以使计算机执行一种或多种运算．

数据：数据是用能被计算机设备接收、翻译和处理的结构形式所表示的事实、概念或规则，它们可以存储于计算机内，也可以用计算机可读的形式存在于设备之外．

文档：指技术资料，这些资料记载着计算机程序的需求、设计、实现和其他细节，也包括用可供人们阅读的形式制备的程序列表及打印输出，这些资料还为计算机程序的使用及维护提供指南．

固件：指由“程序、数据及具有记忆力功能的特殊硬件共同构成的功能实体，这些程序和数据在正常运行的条件下是固定的和无法改变的”．

按照软件的性质和功能，可将软件区分为如下四种类型：

（1）支持软件．支持软件是指用于帮助和支持开发的软件，它包括编译程序、汇编程序、连接和装配程序、库管理程序及相应的文档，还包括调试软件、模拟软件、数据析取和归约软件、开发工作管理软件、配置管理软件、设计工具软件等．

（2）应用软件．应用软件是指用于解决各个专用领域的特殊问题的软件．应用软件种类繁多，凡是以科学计算、工程设计、过程控制、事务管理为目的而设计的软件都属于这个类型．

（3）系统软件．系统软件是指管理计算机系统资源的软件，在软件开发期和程序运行期使用的操作系统和数据库管理系统都属于这一类型．

(4) 测试和维护软件。测试和维护软件是指用于故障诊断、错误隔离、系统调试及检验设备和系统可靠性的软件.

2.1.2 软件工程

软件工程学是在软件危机的背景下诞生的。1968 年在联邦德国 Garmusch 召开的计算机软件技术、管理和维护讨论会上，首次以软件工程一词来标志会议的主题。此后，这一术语很快为科技界所接受.

22 年来，对软件工程一词，有若干种定义方法。

1972 年，Bauer 定义软件工程是“建立和运用完善的工程原则，以获得经济上划算的、又能可靠地工作的软件”。1983 年，IEEE 计算机学会定义软件工程是“用于软件开发、运行、维护和报废的一套方法”。1985 年，Fairley 定义软件工程是“开发和维护软件产品的一套系统的技术和管理规则，能使软件产品的生产和维修以预定的经费按预定的时间完成，其首要目的在于改善软件产品的质量”。1987 年，Macro 定义软件工程是“建立和运用完善的工程和管理原则，开发适用的工具和方法，在已知的及合理的资源条件下，获得满足规定要求的高质量的产品”。1989 年，我国国标 GB/T-11457 采用 IEEE 计算机学会的解释方法作为软件工程的定义.这些定义尽管表达方式不同，但是其内涵是相似的。从这些定义可以看出软件工程的首要目的是获得质量满意的可靠的软件.其次，合理的经费开支和准时交付使用也是软件工程的重要目标，因此可以把软件工程的目标归结为“质量、费用和时间”.

软件工程学的建立，使程序设计由个人创作转变为工程开发，为提高软件产品的质量和可靠性作出了重大贡献。但是，软件工程仍有待继续发展，在软件工程现有的框架内，对软件可靠性的研究还十分局限。需要了解软件工程当前所面临的问题的读者，可从参考文献[4]中找到答案。

2.1.3 产品生存期

任何一项产品，从构思开始，经过设计、制造、使用直至废弃为止的整个期间，构成了产品的生存期。“产品”一词具有广泛的含义，它既用来表示由硬-软件综合系统构成的大型工程项目，也用来表示设备，乃至单个的元器件。产品生存期概念和人的生存期的概念在形式上颇为相似。人生的各个阶段，由于生理及心理特征不同，采用的保健措施是不同的。产品在生存期的各个阶段，也需要采取不同的技术、管理措施。按照产品开发的客观规律，正确区分生存期的各个阶段，明确各个阶段的任务和要求，并对产品不同的阶段实施规范化的管理，是软件工程的一个基本要求，也是获得高质量、高可靠性产品的保证.在硬-软件复合系统中，软件是系统的一个组成部分，软件的开发必须与系统的开发过程相适应，软件的功能及可靠性必须服从系统的功能和可靠性的要求。

国际电工委员会文件 IEC-300 将产品的生存期划分为五个阶段：

(1) 构思和定义阶段 (concept and definition phase)。在这个阶段，使用方和承制方对产品的需求形成共识，就产品的基本特征和性能作出双方认可的定义，并用产品说明书的形式予以规定。

(2) 设计和开发阶段 (design and development phase)。这个阶段的任务，是按照

产品说明书设计制造出符合规定功能要求的原型产品，对原型产品进行实验室模拟试验或现场试验，拟定详细的制造说明书、运行和维护说明书.

(3) 制造和安装阶段 (manufacturing and installation phase). 这个阶段的任务是正式制造出符合规定要求的产品. 对于大型和复杂产品，由于使用者没有能力完成安装任务，所以安装成了制造过程的延伸，在产品移交给用户之前，应该进行交收测试.

(4) 运行和维修阶段 (operation and maintenance phase). 运行期是产品按照预定的功能为用户服务的时期，也是产品的可用寿命期. 为了使产品的功能获得最大的效益，需要对产品的工作状态进行监测，并且采取必要的维修措施. 维修措施包括预防性维修和事故后的修理. 这个阶段有时还需要对产品作一些改进，以提高产品的性能.

(5) 废弃期 (disposal phase). 当产品的可用寿命期已经结束，或用户对产品的需求发生变化使产品无法满足时，产品将被废弃或拆卸，则产品的生存期至此宣告结束. 有时出于经济上的考虑，在技术条件可能的情况下，对已经废弃的产品再次进行大规模的改造，以适应新任务的需要. 这种状况应视为另一个新产品生存期的开始，而不是原产品生存期的延续.

除 IEC-300 之外，涉及产品生存期划分的，还有美国军标 MIL-STD-785B 和我国的国标 GB-6993. 美国军标 MIL-STD-785B "系统和设备研制、生产的可靠性大纲" 将正式使用前的产品获得阶段分为：初步设计构思阶段，论证和证实阶段，大规模工程研制阶段，生产阶段等四个阶段. 我国国标 GB-6993 参照 MIL-STD-785B，将产品获得阶段分为：任务确定阶段，方案论证与审定阶段，技术设计阶段，样机研制阶段，试生产阶段和正式投产阶段.

2.1.4 软件生存期

软件生存期是指从软件产品的构思开始，直至废弃为止的整个时间区间. 软件生存期由若干个阶段构成. 在划分这些阶段时，必须考虑两方面的因素. 首先，由于软件通常是系统中的一个组成部分，这就要求软件生存期的划分必须与产品生存期的划分兼容。其次，软件与硬件性质不同，在开发过程中各有特点，软件生存期的划分应能正确地反映这些特点. 下面是国际电工委员会 (IEC) 对软件生存期阶段的划分.

(1) 构思计划阶段 (conceptual planning phase). 构思计划阶段是指软件生存期的开始阶段. 在这个阶段中，经过使用方和承制方的共同协商，就拟议中软件的功能、使用状况和经费达成共识，并记载于产品开发建议书或研制合同中.

(2) 需求定义阶段 (requirements definition phase). 在这个阶段中，产品建议书被扩充为"产品需求说明". 产品需求说明包括对软件的主要任务、次要任务及相关任务的说明，还包括对软件主要功能、次要功能的说明，以及软件和系统其它部分连接关系的说明. 需求说明是软件设计必须遵循的准绳，因此要求内容明确无误，切忌多义和含混.

(3) 产品定义阶段 (product definition phase). 在这个阶段中，要充分地利用各种技术信息、工程原则和设计者的创造性，按照需求说明，对软件的结构、接口和设计语言作出明确的定义. 这阶段还要进行设计预评和需求评审.

(4) 概要设计阶段 (top-level design phase). 这个阶段的任务是，按照需求说明及产品定义，设计系统和软件的结构，设计软件与外部设备的接口，设计软件和系统其它部

分的接口，设计人机接口，设计数据结构．这阶段还要进行概要设计评审．

(5) 详细设计阶段 (detailed design phase)．这个阶段的任务是对需求说明进行更深入的分析，对系统和软件的结构设计、接口设计和数据设计加以充实和完善．这阶段还要进行关键设计评审(又名详细设计评审)，以确认设计方案满足需求说明的要求．

(6) 实现阶段 (implementation phase)．这个阶段的任务是，按照设计方案和软件的结构层次，逐步编制出软件各单元的程序，进行单元调试和测试，实现各单元代码的逐级综合，逐步形成一个完整的计算机程序，在这阶段要进行程序的综合测试和功能测试(又称初步交收测试)．

(7) 测试和综合阶段 (testing and integration phase)．这个阶段要按规定的环境条件，对系统的硬件和软件进行综合测试．软件和系统的综合是按一定顺序逐步实现的，综合测试以后要对系统及软件的性能作出评估．

(8) 认证、安装和接受阶段 (qualification, installation and acceptance phase)．这个阶段包括软件的安装、测试，以及向用户移交的全过程．这个阶段中，软件必须通过严格的正式测试，以确保产品满足用户的要求．

(9) 维护和扩充阶段 (maintenance[1] and enhancement phase)．进入这个阶段，产品已经完成运行前的准备或已实际投入运行．为考察产品的功能是否满足设计要求，以及用户对产品的整体状况是否满意，产品的运行应置于受控状态．产品运行中还会发现一些始料不及的问题，用户的需求也会有局部的改变，这时往往需要对原程序作局部的修改．

(10) 废弃期 (disposal phase)．产品使用一段时间后，将到达预期有效寿命的终点而被停止使用．有时由于外部条件的变化，使产品不再具有使用价值而提前停止使用．产品生存期到此结束．

软件开发期的划分与系统开发期的划分，虽然形式不一致，其实二者不存在根本的分歧．软件开发期的各阶段与系统开发期的各阶段，存在着如表 2.1 所示的从属和对应的关系．

表 2.1　产品生存期和软件生存期的关系

产品生存期	软件生存期
构思和定义阶段	构思和计划阶段 需求定义阶段 产品定义阶段
设计和开发阶段	概要设计阶段 详细设计阶段 实现阶段
制造和安装阶段	测试和综合阶段 认证、安装和接收阶段
运行和维修阶段	维护和扩充阶段
报废期	废弃期

1) maintenance 一词，可靠性工程界译为维修，软件工程界译为维护，本书采用软件工程的译名．

我国对软件生存期的划分没有统一规定，国军标 GJB-437“军用软件开发规范”中提到，软件生存期包括系统分析与软件定义、软件需求分析、软件设计、软件实现、软件测试、软件维护等阶段，但是没有作进一步的解释。

2.2 软件的质量

2.2.1 产品的质量

什么是产品的质量？从不同的角度可以赋予质量一词不同的含义。关于质量的较为通用的定义是美国著名质量管理专家 Juran 提出的，他认为，“产品的质量就是产品的适用性”，而产品的适用性则是“该产品在使用时能成功地满足用户需要的程度”。这个定义的重要意义在于，明确地提出了产品质量的评价，应以是否满足用户的需要为前提，而不是以制造厂家的见解和宣传为依据。产品的适用性受到时间、地点、使用对象、社会环境、市场竞争等诸多因素的影响，因此，用户对产品的质量要求也在不断的变化和发展。

产品满足用户需要的程度，体现在产品所具备的特性能否满足用户的要求。这些特性是内在特性、外观特性、经济特性和服务状况。

内在特性包括产品的性能、寿命、可靠性、安全性等。

外观特性包括产品的形状、色泽、气味、包装等。

经济特性包括产品的成本、价格、维修费用等。

服务状况包括服务态度及销售、运输、维修、备件、供应等环节。

Juran 还认为，社会上所有的机构，不论是企业、商店、学校、医院、剧场或政府机关，他们的目的都是为社会提供“产品”或提供“服务”。对这些产品或服务的基本要求就是要“适用”，适用性恰当地表达了“质量”的含义。

我国国标 GB-6583.1 将质量定义为“产品、过程或服务满足规定或潜在要求（或需要）的特征和特性的总和”。不难看出，这个定义与 Juran 的定义基本上是一致的，都是以满足用户的要求为最终目标，也就是说，产品必须是为用户的需要而生产，产品的质量优劣，也应由用户来评价。中国质量管理协会对质量的适用性的含义作了更具体的解释，明确地提出了适用性包括的因素有性能、寿命、可靠性、安全性和经济性等。

性能：指产品为满足使用目的所具备的技术特性。如电视机的灵敏度和分辨率，汽车的功率、速度、爬坡能力和载重量，手表的准确性和防水、防震、防磁能力，食品的营养成分和品味等。

寿命：指产品能够正常使用的期限。如灯泡的使用小时数、开关的使用次数、轮胎的行驶里程等。

可靠性：指产品在规定的时间内和规定的条件下，完成规定工作任务的能力。它是产品在投入使用过程中表现出来的，满足人们需要的程度。如电子元器件的失效率、无故障工作时间、火箭的发射成功率等。

安全性：指产品在流通、操作、使用中保证安全，不致发生危及人身安全的事故的能力。如显像管的防爆性能、电器产品外罩的绝缘性能等。

经济性：指产品从设计制造到废弃的整个产品生存期的费用大小。具体表现为设计成本、制造成本、使用成本等。

产品质量就是上述五方面质量特性综合反映的结果。就一个具体的产品而言，在诸多质量特性中有关键的、主要的特性，也有非关键的、次要的特性，必须具体分析，区别对待，以满足用户的特殊需要。

2.2.2 软件的质量要素

软件是一种具有特殊属性的产品，因此产品质量的定义完全适用于软件领域，对软件的质量保证工作具有重要的指导意义，尤其是面向用户的"适用性"的观点，应该成为指导软件开发的座右铭。但是，由于软件的特殊性质，与硬件相比，适用性的含义应该有所不同，需要根据软件的特点作出更明确的解释。软件工程界习惯用质量要素（quality factor）来概括适用性的内涵。

1977 年 McCall 率先提出了软件质量要素包含的内容，此后，Glass 于 1979 年，Cho Chin-Kui 于 1980 年，Reiffer 于 1982 年相继提出了各自的见解，他们的观点既有共同之处，也存在着差别。Macro 在 1987 年对 Glass, Cho Chin-Kui 和 Reiffer 的意见作了比较和分析，发现三者共同认可的要素有 4 个，两人共同认可的要素有 4 个，认识不一致的要素有 6 个。比较结果见表 2.2。

表 2.2 软件质量要素比较分析

一致程度	质 量 要 素
完全一致	可靠性、效率、可维护性、可移植性
部分一致	一致性、可理解性、人类工程(可使用性)、可修改性
不一致	简明性、结构性、健壮性、可测试性、正确性、完整性

在各家的观点中，影响最大的是 McCall 的见解。McCall 认为，软件的质量由 11 个要素构成。这些要素是正确性、可靠性、效率、完整性、可使用性、可维修性、可测试性、灵活性、可移植性、重复使用性及连接性。对这些要素，McCall 提出了自己的解释。

正确性（correctness）：指程序满足需求说明及用户目标的程度。

可靠性（reliability）：指程序按要求的精度完成预期功能的程度。

效率（efficience）：指程序完成其功能所需的资源及代码的数量。

完整性（integrity, security）：指对未经许可的人员接近软件或数据加以控制的程度。

可使用性（usability）：指熟悉程序操作、为程序准备输入数据和翻译程序输出所需付出的努力。

可维护性（maintainability）：指确定可运行程序中的错误所需要付出的努力。

可测试性（tastability）：指保证程序执行其预定的功能，在测试时所需要付出的努力。

灵活性（flexibility）：指修改可运行程序所需要付出的努力。

可移植性（portability）：指移植程序至另一个硬件配置或软件系统环境，需要付出的努力。