软件工程
—设计、可靠性和管理

M.L.舒曼 著
朱兆堂 许惠良 译
黄嘉启 睢重星

上海翻译出版公司

软 件 工 程

——设计、可靠性和管理

M.L. 舒 曼 著

朱兆堂 许嘉良 黄嘉启 睢重星 译

上海翻译出版公司

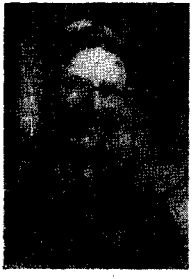
"内 容 简 介"

本书系统地介绍了软件工程中设计、测试、可靠性以及项目管理等一些领域。

全书共分六章，各章分别介绍了软件工程学的基本概念、软件的设计过程、软件的复杂性、软件测试的方法。并介绍用可靠性概念，推导了几种能预测软件的错误数、可靠性的模型。书末还列出了经常引用的概率论、可靠性理论的附录。

Martin L. Shooman
SOFTWARE ENGINEERING
Design, Reliability, and Management
McGraw-Hill Book Company
1983

软件工程
——设计、可靠性和管理
M. L. 舒 曼 著
朱兆堂 许惠良 黄嘉启 顾重屋 译
上海翻译出版公司
（上海复兴中路 597 号）
新华书店上海发行所发行 上海东方印刷厂印刷
开本 787×1092 1/16 印张 29.25 字数 744,000
1989 年 10 月第 1 版 1989 年 10 月第 1 次印刷
印数 1—3,200
ISBN7-80514-209-2/TP·28
定价：12.25 元



中 译 本 序

我衷心希望本书中译本的出版能使广大的中国读者了解到软件开发方面的一些新观念，懂得怎样在规定的预算范围内及时地开发出高质量的软件。当今世界上许多科学技术都与数字电子化技术的发展紧密地联系在一起。目前存贮器和微处理器的功能益趋增强，价格益趋降低；但程序却益趋庞大复杂。本书自1983年首版以来，书中的大部分设想已为许多软件开发者所采纳实现；但是，在怎样充分地利用现有的技术，怎样开发新方法方面，还需作出更大的努力。

近来，软件工具和计算机容错系统是人们十分感兴趣的两大领域。软件工具就是那些在软件开发过程中可以用来帮助确定软件规范、设计软件、程序编制、书写文档和测试软件的程序。容错技术也就是在计算机系统中利用某些冗余的硬件和软件来获得相当高的系统可靠性。感兴趣的读者可以查阅有关文献：

Proceedings Conference on Software Tools, April 1985,
IEEE Computer Society, Service Center, 445 Hoes Lane,
Piscataway, NJ 08854, U. S. A.

Probabilistic Reliability, 2nd. Edition, 1988, Krieger Publishing Co.,
Box 9542, Melbourne, FL 32902-9542, U. S. A, Appendix H.

M.L. 舒曼于
纽约长岛 一九八八年

作 者 简 介

M.L.舒曼是美国工业大学电机工程和计算机科学系教授，负责软件工程项目的研究。他毕业于麻省理工学院，在布鲁克林工业学院电机工程系获得博士学位，兼修数学、物理。舒曼博士是麻省理工学院的访问教授，他曾在外国和美国的许多大学里作过讲演，他曾担任贝尔实验室、美国国家宇航局、IBM公司以及美国陆军部门的顾问。

作者是美国电气和电子工程师学会(IEEE学会)理事，他曾五次获得IEEE学会可靠性分会和计算机分会的最佳技术论文奖。由于他在软件工程方面的卓越贡献，他获得了可靠性分会的嘉奖。在五次软件工程的国际会议上，他被推选为学术会议主席、常务主席。

舒曼博士还著有《概率可靠性：一种工程方法》一书(由McGraw Hill公司首版于1968年，Krieger公司于1988年第二版)，他还为其他八本书写了有关章节。在软件工程、可靠性和控制系统方面，他发表了八十多篇论文。他是三个美国荣誉科学研究学会的成员。舒曼博士被列入《美国名人大全》。

译　序

软件工程学是一门研究如何用工程化的方式有效地管理软件开发，以较低的成本按期开发出高质量软件的学科。软件工程方法学的推广应用必将进一步促进计算机的应用推广。本书比较全面地介绍了软件工程中设计、测试、可靠性以及项目管理等主要领域。

全书共分六章。第一章介绍软件工程学的基本概念。第二章介绍软件的设计过程，包括自顶而下、自底而上和结构化程序设计等多种方法。第三章介绍软件的复杂性模型。第四章介绍软件测试的彻底性概念、测试方法以及程序测试模型。第五章阐述了可靠性概念并推导出了几种能预测出软件的错误数、可靠性和可用性的模型。第六章着重介绍了软件系统项目的成本估计模型和项目管理方法。本书还列出了软件工程中经常引用的概率论、可靠性理论和图论基础知识的附录。

由于作者在计算机软件理论方面的精深造诣和在系统可靠性方面丰富的实践经验，目前，他已被公认为国际软件可靠性研究方面的先驱。本书被列为美国著名的麦格劳-希尔(McGraw-Hill)出版公司的计算机丛书之一。

因为本书涉及的面较广，每一章中还有不少的专题。全书按模块化方式编写，每章都是独立的，读者完全可以根据自己的需要选择必要的章节阅读。本书可作为计算机专业和软件专业高年级和研究生的教材，讲授半年，也可以结合学生的软件项目实践进行教授。

本书第一章及第六章由朱兆堂、杨立钧译，第二章由雎重星译，第三章和第四章由黄嘉启译，第五章由许惠美译，全书由朱兆堂和崔诚负责校对。朱三元同志仔细地审阅了译稿。何钟儒、陈慧等同志为本书的编辑和稿件整理做了大量的工作。在此我们表示衷心的感谢。我们还要感谢上海计算机开发公司和上海凯斯特软件公司对本书出版的帮助和支持。本书译者们为了让书与读者早日见面，以促进国内软件开发生产方式的改进，鼓励广大读者应用软件工程方法学的信心和兴趣，也在经济上为支持本书的出版作出了力所能及的贡献。

限于译者水平，译文有不妥之处恳请读者不吝指正。

朱兆堂

一九八八年

第一章 引 言

1.1 问题所在

推动二十世纪发展的某些最根本的动力，从技术上来说是起源于运输技术、通讯技术、能源技术以及生产制造技术。这些领域的进展与汽车、飞机、无线电、电视、雷达、核能、太阳能、生产组装线和自动制造业的发明及其广泛使用是紧密相联的。另外与上述各种巨大发明相互作用、并扩大了其影响的两大发展，那就是电子技术和计算机技术。电子和计算机又是不可分割地联系在一起的，因为我们现代化的计算机都是电子化的数字装置。

我们可以将一个计算机系统分成输入、输出和存贮系统(阅读器、打印机、终端、磁盘以及磁带存贮装置)，中央处理单元(思维判断及计算部分)和软件(程序、输入、输出数据和操作指令)。为了帮助我们进行现代化的计算机软件系统的计划、设计、研制和维护等工作，本书将研究这方面的工程技术、方法和理论问题。

本章的目的旨在让读者了解复杂系统中的软件设计问题，程序错误的普遍性和影响，以及人们在面临这些问题时，如何用经济的方法来进行软件设计。当然，经验丰富的专业人员熟知这些问题，他们可掠过这一章，从第二章开始读起。

1.1.1 看来功能无穷的计算机

人们在评述计算机项目时，从未听到过这样的话："我想这个项目我们没有能力上"和"我不能肯定此项目是否值得上"。看起来这似乎令人感到奇怪，因为每个工程师和科学家都知道，每种先进技术都有它的局限性，经济方面也有一定的限制，这些会限制着我们的作为。因此，我们的技术发展只能分阶段进行。但当谈到计算机时，我们所有的人，甚至连计算机行业里的专家们也会为其所显示出来的功能而吃惊。现在我们使用计算机所解决的问题正是25年前在惊险的连环画和科学幻想杂志中所描绘的问题。

在大多数情况下，我们是卓有成效的。计算机已经带来了超越人们想象力的技术飞跃。但人们不常谈论到的是，在某些场合下，我们以极大热情所阐述的那些系统的复杂性已造成了巨额开支，而且系统的不可靠性将严重到超过系统效益的程度。有时，我们往往会不明智地用成本虽低但错误百出的"计算机功能"来代替"人的功能"。实际上，更慎重地选择目标，或者对项目进行更深入透彻的研究、规划和工程设计，会使项目实施时的经费省得多。最后，即使不能完全达到原来的期望，至少能获得其中的大部分。诸如此类的例子不必由计算机专家来提供，有关这方面的报道已比比皆是：

1. 最近报纸上有篇文章报道说："由于一台跟踪飞往纽约和费城机场的计算机局部失灵，使得至少100架降落的班机晚点两个多小时……故障可能是由昨天新装了程序而引起的"。从一位空中交通管理员的正式报道来看，情况更为严重："在8点到4点的那一班中，我的雷达(屏幕)上两次全部空白。至于有多少飞机在我的控制之下，其飞行的位置、高度和去向都无从所知，因为屏幕上什么都没有，只是一片空白"。

2. 一本杂志报道了11个计算机错误的例子。其中最异乎寻常的是有关芝加哥一家大旅馆的事例。这家旅馆定制了一批信件，将它们分送给昔日光顾的旅客，以促进今后的旅馆业务。糟透的是，负责打印的计算机送错了一份邮寄名单，这样，成百个芝加哥的家庭主妇收到并拆开了来自这家旅馆的感谢信，信中感谢她们的丈夫曾光顾这家旅馆。瞬眼间，这家旅馆的电话多得应接不暇，愤怒的丈夫们要求旅馆对他们的妻子作出解释，因为很多妻子甚至威胁着要离婚。

3. 最近报上登载了一篇题为"当计算机出故障时，谁来承担这笔费用？"的特写文章，文章引证了由美国律师协会的计算机法律处提出的三种假设、但并非虚构的情况：

a)一位空中交通管理员靠计算机控制台来的信息进行导航管理，结果将两架波音747引向交叉路口，飞机相撞、起火，全部机毁人亡。

b)一台用以监视病人手术后痊愈情况的小型医用计算机失灵，没能提醒医务人员病人正在中风，病人因此而死亡。

c)一家公司发现他的计算机篡改了公司非常有价值且十分敏感的信息资料，却无法纠正这些错误。由此造成的损失严重地影响了这家公司的市场。

虽然上述的许多问题并非现实，其目的在于告诫读者，编制一套程序，也就是用计算机软件来指示计算机硬件进行必要的计算，如果软件不善，则后果将不堪设想。为了改进目前的状况，我们必须进一步懂得软件开发过程，学会如何估计并综合考虑生产软件所需的时间、人力和经费。一个主要的目标，是懂得如何进行初期估计以及后来对质量、可靠性和软件产品成本的度量问题。

最热衷于搞计算机系统的部门中，有相当一批军事机构，它们忙于处理益趋扩大并且复杂的技术问题。Barry Boehm在他编辑的图例中对这种趋势作了图解说明，见图1.1。

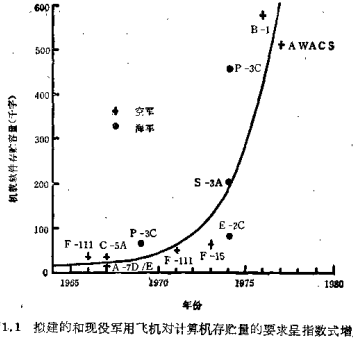


图1.1 拟建的和现役军用飞机对计算机存贮量的要求呈指数式增加

从1970年到1978年的整个阶段，机载计算机的存贮量飞速增长，已成为年份的指数函数。

在1.1.2节中，我们将评述数字电子技术的迅速发展所带来对器件和设备的成本、体积、重

量及对电源要求的迅速下降。然而，问题的关键是这些发展能否跟得上在图1.1中所要求的(计算机存贮量)增长趋势。

有两种不同的曲线拟合程序可用来拟合图1.1所示的数据。表1.1给出了原始数据和用来拟合的乘方模型或指数模型。我们可以从中预测到，如用拟合较好的指数曲线对后几年进行外推，到1985年最大的机载计算机存贮量将超过400万字。但这种预言的具体实现，还得与1.1.2节中提到的集成电路及磁泡存贮器联系起来。1.2节将探讨怎样经济地记录这样大的程序的可行性。

表1.1 对图1.1Boehm曲线的数学拟合

年	距离1960年	图1.1中存贮字数	乘方曲线* $M=199.3(Y-1960)^{1.83}$	指数曲线+ $M=4080e^{0.28(Y-1960)}$
1965	5	25,000	13,705	16,588
1967	7	30,000	22,188	29,070
1969	9	35,000	64,251	50,945
1971	11	55,000	108,884	89,281
1973	13	180,000	168,915	156,465
1975	15	290,000	246,052	274,203
1977	17	600,000	341,909	480,540
1980	20	……	524,129	4,474,263
1985	25	……	946,432	1,114,846

注释：M=存贮字数，Y=日历年。

* 表示采用德克萨斯仪器公司SR-52统计库"乘方曲线拟合"程序ST1-09。(注：修正系数=0.84)

+表示采用"指数曲线拟合"程序ST1-10(德克萨斯仪器公司SR-52)。(注：修正系数=0.93)。有关本程序的深入讨论和曲线拟合精度可见问题1.3。

1.1.2 数字电子技术的发展

二十世纪的技术开创了一个美妙的电子世界。在第二次世界大战之前和第二次世界大战中，技术方面曾取得了卓越进展的真空管，但战后就立即被晶体管所取代[1]。50年代初，当晶体管盛行的时候，所有电子线路的大小、成本及功耗都有了明显地缩小。晶体管故障率的减少，对冷却要求成倍地下降，大大提高了计算机的可靠性[2]。近年来，集成电路和小型计算机的革命使计算机的成本、大小、故障率以及对电源的要求就更低了[3]，并且为这些计算机设备开辟了全新的市场。

图1.2和图1.3定量地描述了集成电路的进展及其对计算机硬件的影响。图1.2说明了每美元所对应的计算能力(每秒处理的字位)按年度的增长情况。图1.3说明了每位二进制数存贮器价格的逐年递减情况。这些曲线将在问题1.5中再次被用来说明计算成本的变化。

1) Braun and MacDonald(1978)。

2) Thornton(1970),p.20—21。

3) Schnable et al.(1978),p.6—13。

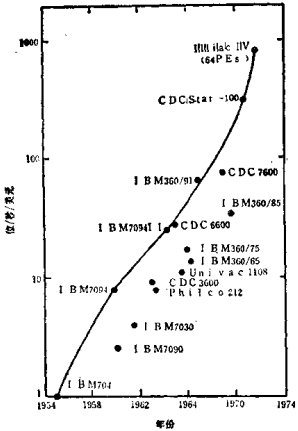


图1.2 通用计算机硬件成本效益曲线
(Turn,1974,p.55)

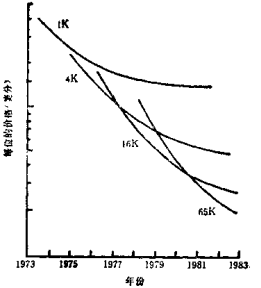


图1.3 存贮器价格逐年递降曲线,不同的曲线对应于不同的组装密度,从一条曲线到另一条曲线的转移表示了组装密度的跃迁。
(Noyce, 1977 年 Scientific American)

在讨论并明确了程序规模和存贮器成本在效果上反作用的问题之后,下一节中我们将转入对人的作用的讨论。

1.1.3 程序人员是科学家,是工程师,还是艺术家?

软件行业的症状之一是人们对于程序人员应起的作用看法很混乱,程序人员到底是教师,还是学生,是专业人员,还是经理,都说不清。并且他们对于系统分析员,软件工程师,软件经理或其他人员所起的作用也分不清。当然,在大型、复杂的开发项目中,这类问题是无法避免的,这正是怎样定义各个设计部门之间的相互衔接,以及在整个设计过程中如何把它们组合起来的关键问题。为了弄清这些问题,让我们来看一下科学家和工程师譬如在电气或机械系统中所起的作用。

首先,大家都同意,科学家的作用是属于研究和高级开发性质的,他们发明创造的新技术、新方法可以用来帮助其他人以更好的方式或方法来从事开发工作。有时,他们被派去从事一些开发项目,往往是以顾问的身份来传授他们的高超技术。另一方面,我们有设计者,他们的

主要作用是进行设计(首先以模型方式先构成设计)，证明设计是可行的，然后将这个初步设计转换为成品，这个成品可以批量生产，能符合环境规范，并且在制造工艺中使用标准部件。

大师的创造性工作在于怎样构思出一种可行的设计，证明这种设计符合规范，并作出初步的成本估计。这些工作通常由工程师来承担，他们必须一直留在项目组内直至模型被构成并经试验，然后将其移交给设计者(或制造工程师)为止。当然，在工程与设计之间的分界线不是很明显的。

现在让我们把软件领域内的业务状况与上述情况作一对照。多年来，程序人员常认为他们自己就是艺术家，某些人到现在还这样认为。艺术家这个称号对于少数在工艺技术上达到非凡程度的专家来说可能是适当的。然而，象这样的人目前在软件领域中实在太少了。如果我们想要一种由这类艺术家生成的软件，我们只向他(或她)提供任务的说明书以及所有的必要信息，然后悄然离去。两年后再回来，向他们买这件完成了的杰作。

也许只有象具有米开朗琪罗[1]这样才能的人，才能胜任这样的工作。可惜，连当代最杰出的程序员也无法和他相比，在技能、才干和持久能力方面，要好几个人加起来才能与米开朗琪罗相比拟。况且，罗马西斯庭教堂的壁画是花了五年的时间才完成的。如果我们听(或见)过有关米开朗琪罗生平电影的话，可能会记得教皇朱丽叶斯二世还经常会漫步到教堂里，仰望站在脚手架上的米开朗琪罗问道："做完了没有？"

程序设计领域的另一端，我们有计算机科学。在它的指导下，大量的软件人员正在研究哪一种理论更适合于计算机行业的要求。这方面的大部分内容都是为了使得程序设计以及计算机的实施结果更加公式化、概念化，使之容易为人们所理解。然而，这些概念却不能用来帮助我们从事实际的软件生产。绝大部分的任务仍将由软件工程师和软件设计人员来承担。困难的是能真正履行这两种功能的人员实在太少，甚至大学里也在为开设软件工程这门课程大伤脑筋。

1.1.4 缺乏程序工程和设计技术

大部分工程方面的设计技术不是合成技术，而是迭代技术。先粗略地进行初步设计，然后再对它加以分析，看它是否符合规范。如果不符合规范，必须对其修改。要得到一个满意的设计，上述过程需重复多次。因此，对设计进行分析、再将分析结果与要求的规范进行比较的能力，在设计过程中是必不可少的。缺乏足够的分析技术，这正是软件工程最致命的弱点。

我们可以看一下与之相似的其他工程技术。例如，在机械工程中，有着大量基于自由体图的静态、动态分析，热力学理论以及应力应变计算技术。我们可以用运动学的语言来讨论程序的静态分析和动态分析。静态分析是指当控制信息通过并离开所研究的单个程序模块时，其输入量、起始条件及软件是怎样把程序变量从初始状态变换到最终状态的。现在的许多技术可用来帮助进行这种分析。例如，我们可以研究程序在输入和存贮时的数据结构，并人为地用程序来处理一组数据，再检查每条指令在数据变换中所起的作用。

动态分析比静态分析困难得多。通过动态分析，我们可以对一个程序在新输入景驱动下的数据变换的全过程进行研究，例如怎样穿越一条路径，怎样把控制信息从一个程序模块传递到另一个模块的。目前，这种工具十分缺乏，使设计过程更加困难，并妨碍了我们在解决问题过程中，明确地说明或者文件化地记录数据的全部演变过程。

1) 十五世纪意大利著名雕刻家、画家和建筑师。——译注

1.1.5 根深蒂深的计算机硬件的经营管理思想

假定大部分高级经理都是50多岁，他们先前接受教育时只有20多岁，从事工业必有30多年。这意味着他们所受到的学校教育是在50年代结束的，那还在数字计算机普遍使用之前。如果在他们晋升到现有职务的过程中，曾参加过某个计算机的应用项目，他们必然尝到过一些程序设计和软件开发的滋味，也懂得一些具体细节。很可能他们在计算机方面的大部分经验，还是近些年来才有的（在他们已获得了相当高的职位之后），且是从系统硬件中得到的。一些中层经理的情况也同样如此。然而，他们中很多人可能已参加过与软件有关的项目，因此对软件问题富有亲身体会。总之，要管理和处理软件问题，懂得业务是第一步，但经营管理的工具和经验也是必不可少的。

多年来，在项目审查中，经常出现的现象是，硬件人员常常能提供像进度表、系统方块图和有待解决的具体问题清单之类的资料。这种定量资料对于高级经营管理部门来说是容易理解的，并且对于判断这个项目开展得是否顺利，是否符合进度或是否需要适当的监督都很有利。我们可将这种情况与以往软件人员介绍项目时的情景进行对照，"靠我瞎！我们有优秀的程序员！程序能编得又好又及时！"。过去常常会发生这种情况，所有的程序模块编写后都是由各自测试的，但这些模块[1]集成和后阶段排错这一冗长而又困难的任务只完成了一部分。往往会引起数周或数月的拖延，严重的进度迟误和超支常会造成内外交困的局面[2]。

目前，软件（形势）情况已有明显的好转。现代化的设计技术可进行自顶向下的模块设计，这种设计便于解释，而且容易集成。各种设计表示法如HIPO图，伪代码和结构流程图都可以用高级管理部门所能理解、吸收的形式表达出来。然而以往的问题、错误和拖延常常会妨碍管理部门对新改进的领会和对项目的支持。实际上，让上层管理部门（和许多专业人员）意识到重大的变化已经发生，并使他们接受最终得到他们的支持是一个长期的问题。这一点将在第六章中予以讨论。

从长远来看，大型项目一般都有时间拖延和超支的情况（如果管理部门在原估计中包括了"安全系数"，其中有些超支就可被掩盖）。对硬件来说，只要使用一些现有的管理工具和经营管理经验，在项目进行过程中就可预测出上述的超支。但对软件来说，我们还刚着手开发并应用这方面的工具、技术和分析能力。

既然我们已揭开了问题的实质（复杂性所引起的灾难；现有的硬件成本低，提供方便；但缺乏适当的软件设计工具；管理人员和设计人员幼稚无知），那么就来衡量一下问题的严重性。

1.2 问题的严重性

1.2.1 软件代价高

计算机行业已被我们普遍认为是国民经济的一个重大组成部分。但有多少人会意识到在1980年的政府财政年度中，计算机系统方面竟耗费了大约570亿美元[3]！更令人震惊的是其

1）这里所用的术语——程序模块（module）是用来表示能执行某种特定功能或子功能程序中的一个较大的独立部分。

2）"当项目价值达数百万美元，最理想的进度也得三年或更长一些，这类项目的典型超支将达200～300%，已达到无法容忍的地步"。（Putnam和Wolverton，1977，p.i）

3）为了便于比较，可以汽车行业为例，汽车的年销售量900万辆，每辆汽车平均价格为8,000美元，年销售额仅720亿美元。

中320亿美元(占总数的56%)是用于计算机软件方面的。人们逐渐开始意识到在开发一个新型计算机系统或修改一个现有系统的过程中,最大部分的资金是用在系统软件开发方面。

如果技术进展能使硬件成本继续降低(修正了通货膨胀系数之后),设计者就能在越来越广泛的范围内使用功能越来越强的计算机。虽然日趋复杂化的系统会引起成本上升,但元器件价格的不断下降能与之抵消,这就使得新型计算机设备的硬件总价仍保持相对稳定。然而由于多种原因软件成本还将不断提高,如:(1)每一项新应用就意味着要有新程序;(2)新计算机代替了现有的旧计算机后,常常意味着要有新的软件,或者对现有的软件进行修改;(3)编制程序是一项劳动力高度密集的产业,受到通货膨胀率的严重影响。虽可采取一些新技术来提高软件的生产率,但是这些收益未必就能抵消今后工作量的大幅度增加。

通过对软件费用逐年的研究估计方法的研究,我们就能对软件费用进行深入的分析。在图1.4中,可以看到一条有关美国空军计算机系统的软、硬件费用趋势的分界曲线(1972年以后的曲线是外推得出的),表1.2是我们利用了这些数据以及1980年的联邦预算资料而编成的。美国空军自动数据处理方面的费用占空军预算的9%,而费用中的80%是软件(摘自Barry Boehm为首的兰德公司资料,美国空军,1972年),表1.2中整个军事部门中的软件费用就是按照这个百分比假定的。所对应的其他两类百分比是作者估计的。非政府经济部门的软件费用只占预算的0.5%,与此相比,军事部门的软件费用占年度预算的7%。很清楚,这就是军事部门为什么急于要促使软件开支下降的原因。

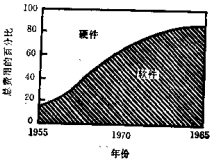


图1.4 硬件和软件费用的趋势(美国空军,1972)

表1.2 美国自动数据处理和软件费用的估计

经济部门	预算或国民生产总值(单位:10亿美元)[1]	计算机总费用占预算的百分比	软件费用占计算机总费用的百分比	软件总费用的估计值(单位:10亿美元)	硬件总费用的估计值(单位:10亿美元)
国防部门预算	125.8	9	80	9.1	2.3
美国政府其他部门的预算	405.8	5	50	10.1	10.1
美国国民生产总值	2565	1	50	32.0	12.8
总　　计				12.8	25.2

1) "Carter Faces Problems in Achieving His[1980] Budget Goals," Wall Street Journal, Tuesday, Jan. 23, 1979, pp4,5。

其他类似的经费估计值也证实了表1.2的计算：

1. 根据1974年美国有450,000名程序员和他们的平均年薪(包括管理费)35,000美元的资料，Boehm估计出该年的软件产值是160亿美元[1]。

2. 使用1975年的数据，进行类似表1.2的计算，可得出该年的软件产值是190亿美元[2]。

3. Arthur D. Little在《数据自动化》(Datamation)杂志上发表了一篇综述报道，文章说计算机总销售额已超过300亿美元，并且一次性出售的软件销售额达30～40亿美元，其中还不包括军用软件、公司提供的软件和用户自编的软件[3]。

4. 有关计算机软、硬件费用方面大量详细的资料可参阅Phister的著作(1979年)。

软件方面增长的另一项估计值是从计算机制造商提供的软件数量(机器指令数)中获得的。可注意到1954—1967年间软件是呈指数状增长的(见表1.3)。至1970年可外推到570万字的存贮量，但是将外推到1985年的330亿字是不合理的。要控制巨额的软件成本，有赖于生产率的极大提高(见本章结尾处的习题1.2到1.10)。

表1.3 用指数曲线来拟合有关软件量增长的McClure数据*

年	年-1950	存贮字数	
		McCLure数据(机器指令数)	指数曲线+$M=951e^{0.42(Y-1949)}$
1954	4	5,000	5,414
1956	6	20,000	12,919
1959	9	35,000	47,638
1961	11	100,000	113,657
1964	14	350,000	418,983
1966	16	1,000,000	993,838
1967	17	2,000,000	1,544,532
1970	20	.....	5,693,740
1980	30	.....	440,522,966
1985	35		3,268,223,800

* 这里的机器语言编码数是用来支持该年描出的标准的计算机产品的，它是由 McClure in Naue 等准备的 (1976)。

+ 采用"指数曲线拟合"程序ST1-10(德克萨斯仪器公司SR-52)(注：修正系数=0.926)见问题1.3。

1.2.2 软件错误的影响

系统运行过程中所暴露出来的软件错误[4]会造成两个方面的损失：(1)使系统受到了损害；(2)随之而来的纠正工作。举一个系统出错的例子，一个银行的电子存款-转帐系统中，假定有位客户在开列新帐户的当天又撤销了它(这种情况极其偶然)。如果系统只是成功地开列了帐户，却没有及时将其撤销。那么很明显，就会产生多付款的现象。当一位银行经理遇到这

1) 1975(p.4)。

2) Shooman(1978)。

3) Rothenbueschen(1978),p.85—110。

4) 第五章中，软件错误(Software error)被定义为由于内部错误所引起系统外部操作上的问题，俗语所说的差错(bug)只是用在不需要精确定义的场合。

种错误情况时，可以有下列四种不同的选择：

1. 这种事情难得发生，或许有人用普通的会计方法就能把错误鉴别出来；再说人们利用这种错误来窃取资金的概率极小，因此，错误就被遗忘。

2. 由于纠正错误要付出较高的代价，发生的概率又小，因此不再修改软件，只是通知全体有关人员注意这种异常现象的发生。

3. 在适当的时候，譬如说，数月内按原计划准备发行新软件时再一起予以纠正。

4. 立即指定一名软件设计人员进行诊断，重新设计，测试并纠正错误。

显而易见，方法2或方法3对这种场合比较合适（也适合于大部分场合），方法1和方法4就很不合适，而且是不经济的。预料中的损失将与由于错误可能引起的资金失窃[1]和监视错误所需的时间有关。使用第六章的技术，就可以将这个问题的代价估计出来。

讨论代价问题之前，首先应该弄清软件开发过程由哪些阶段组成。所谓生存周期是由如表1.4所示的那些阶段组成的，此表对于任何的大项目都是适用的。就软件来说，还需要有一系列的注释。首先，表1.4的设计过程中包括有软件设计和程序编写（编码）。一般来说，软件并

表1.4　大型系统开发过程中各项目阶段的一览表

1. 概念化阶段
2. 需求分析
3. 规范
4. 初步设计
5. 验证和测试设计
6. 重新设计
7. 样机制造
8. 组装和系统综合测试
9. 验收测试（设计验证）
10. 生产（如需要数个系统）
11. 现场（运行）测试和排错
12. 现场维护
13. 添加特性的设计和安装
14. 系统报废（系统消亡）或整个系统重新设计

没有一个相当于样机（如线路试验板，比例模型等）生产的阶段。然而，第六章里，我们将讨论一些管理技术，其中包括两步法的设计过程，设计过程中的第一阶段可考虑作为样机阶段。其次，在表中的第六到第八阶段之间的某处，即在系统综合集成之前，对每一种独立编制的软件功能或进程（称为程序单元或模块）进行模块测试。最后的生产阶段不过是计算机程序、磁盘和只读存贮器[2]（ROM）的复制。现在我们将讨论整个开发周期的总工作量的分配方法。

如果观察一下Boehm编制的表1.5，我们可以看到程序设计项目中40%的工作量去用来进

1）错误所引起的危害不一定就涉及到失窃事件，例如，系统结果上注销了这位客户（或不同的客户）在银行中的另一个帐户，要纠正这样的错误会使用户十分头痛。

2）有人把存贮在只读存贮器里的软件称为固件（firmware）。

表1.5 软件工作量按各种活动的分布情况

	分析和设计*	编码和审查†	测试和集成*
命令控制型 (SACE, NTDS)	35%	17%	48%
命令控制型 (TRW)	46%	20%	34%
宇航型 (Gemini, Saturn)	34%	20%	46%
管理型通用程序设计 (OS/360)	33%	17%	50%
科学型 (TRW)	44%	26%	30%
商务型 (Raytheon)	44%	28%	28%

选自Boehm(1975), p.7.

* 未把文件成本(需再增加10%)估计在内。

† 审查是指模块在上机测试以前,通过阅读程序和研究(桌上检验)来发现错误。

行软件测试以发现错误,并修改软件(即再设计)以消除已发现的错误。如果我们能确保绝对正确地进行程序编码,那么,就可节约一笔相当可观的原来用于消除错误的开支。现代化的程序设计技术(结构化程序设计,自顶向下设计法)可使出错率大为降低,但还会有些错误,目前我们还无法把测试工作量降低到零。事实上,我们不应该过于信任一种方法(这只是冒险)而减少测试工作量,除非我们对低含错(low-bug-content)技术很有经验或者有能力测且含错量,以便判断软件的质量。在开发周期中定量估计软件质量,对于评价项目的进展情况是很重要的。因此,第五章所述的可靠性和含错量的测定,也可以作为销售管理部门定量地确定何时已经过了充分的测试,产品可准备发行的量度。

至今我们主要提到的还只是开发过程中的错误,一些代价更大的错误往往要到现场才能察觉出来。图1.5有趣地表示了在软件开发各个阶段中计算机错误排除所需的相对成本。随着项目的进展,排错代价越来越高,这有多方面的原因:

1. 测试日趋复杂,成本更高。
2. 修改的文件和文本需分发的面越广,成本越高。
3. 问题和修改信息传递的面越广,涉及的人员越多。
4. 多次反复地进行以前的测试(回归测试),成本更高。
5. 一旦系统运行后,原来的研制小组即告解散,需要重新指定新的小组。

程序运行开始前,还需要准备一笔资金用于对程序进行额外的检验和测试,这样可以大大减少计算机运行后可能产生的那些昂贵的错误量。因此从长远来看,其收益是可观的。1.2.3节将对运行阶段中纠正错误所需的巨大代价进行讨论。

1.2.3 软件修改和软件维护问题

开始研究软件工程领域中的数据时,就可以发现数据量越大,软件研制成本就越大,软件维护成本则更大。由于维护成本的概念常常容易混淆,有必要先对它进行仔细的定义。维护成

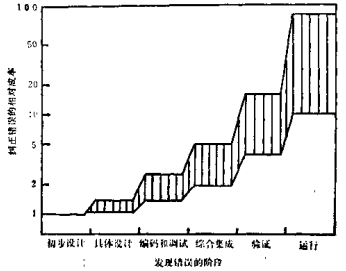


图1.5 修复错误所需的相对成本与各阶段的关系曲线,数据资料来自IBM-SDD, TRW, GTE和贝尔实验室程序。上、下两曲线之间的阴影区域表示了95%的置信区间(Boehm,1976b)

本是指修复那些已正式发行的以及正在运行的软件错误所需的代价总和。此外,几乎每个使用了若干年后的大型软件系统都有这样或那样的修改。我们常把那些与修改有关的工作称为再设计(增强,增加特点),这部分工作量十分巨大,它往往接近、甚至超过软件的初期成本。第五章引证的部分资料表明,在数据处理方面,预算的30%到80%往往是用在再设计和软件维护上的。许多情况下,因为再设计和软件维护的费用混杂在一起,这两方面工作很难分开。

从经济观点来讲,较聪明的做法是再多付10%的费用,用来开发可靠的软件并建立相应的文档资料。长远来看,这样做可减少以后的维护业务,使将来的再设计更加方便,收益较大。问题在于软件的开发和维护往往分属于不同的部门,预算也是分开的。因此,明明开发部门通过谈判只要增加10%的费用就能使今后的维护预算降低20%,常常也很难做到。所以在初期规划系统时,怎样分配整个生存周期中各阶段的费用是非常重要的。在硬件设计中,我们通常是采用标准化的零部件。1.2.4节将考虑软件设计可否标准化的问题。

1.2.4 我们能软件标准化吗?

目前软件行业中存在着一种"再发明轮子"的迹象(译注:轮子是人类历史上一项了不起的发明,既然发明了,人们就不必一次次地再去发明,只要懂得怎样使用它)。亦即,人们仍继续不停地设计许多程序或者一遍又一遍地设计程序的大部分。软件标准化的一个明显的例子如检索和分类,这个领域早为人们熟知,这些功能已成为Fortran 77的内部操作提供给了用户。在数学软件方面,也有许多执行矩阵运算、统计分析和绘图功能的程序。然而,就是这样一些程序,人们还在进行无价值的重复设计。也正是这批做学软件的开拓者,他们支持了"合格软件"(Certified Software)的事业。

合格软件应该是由若干标准化机构同意批准的,类似于Underwriters实验室测试并批准某些电子产品一样。这种软件将符合某些确定的标准,描述清楚,周期性地(或偶尔)进行维护,有定义严格的但又灵活的接口。总的来说,合格软件可被包含在其他项目里边,也可以作为一

项单独的产品独立使用。下列参考资料介绍了合格软件领域里的一些进展情况：

1. 用Fortran和PL/1语言的IBM科学子程序包[1]。

2. 合格软件会议[2]。

3. 英国标准软件项目[3]。

软件标准化工作进展缓慢的原因有几个，最重要的原因是要完成标准化并要克服标准化工作中的吹毛求疵、派别偏见都需作巨大的努力。Rosen著作[4]中介绍了一些Fortran、Algol、Cobol和PL/1语言标准化的早期历史。本书第二章讨论了最近在Fortran语言和美国国防部的Ada语言标准化方面所开展的工作。1979年IEEE计算机学会办公室和美国国家标准学会首先标准化了Pascal语言。我们期待着软件合格化方面新的、重大的成果即将到来。

1.2.5 本世纪最后廿年内的计算机行业

现在我们将像权威那样来预测本世纪最后廿年内计算机行业的发展趋势以及对软件开发过程的影响。计算机行业的发展不是革命式的，而是进化式的。因此，下面的许多评论只是从已经讨论过的内容中逐条引伸出来的。然而，把它们收集起来归纳成一张表，可以更好地说明它们之间的联系，为读者描绘出一幅综合的前景。这段时间内我们的期望是：

1. 计算机中央处理单元的价格继续下降。

2. 计算机存贮器的价格继续下降。

3. 单片集成电路微处理器的功能继续增强。

4. 外部设备如卡片阅读器、磁盘、打印机和磁带机的价格也将降低，但比中央处理器慢些[5]。

5. 八十年代，计算机行业将赶上汽车行业，成为美国经济最大的组成部分[6]。

6. 从八十到九十年代，将使用一些更新的程序设计方法，并促使越来越多的程序员运用新方法，尽力降低软件的成本。

7. 增强对计算机动态操作的理解，发展更好的程序结构理论。

8. 开发更多、更好的软件工程设计技术。

9. 软件设计工具将作为正规软件的一部分，由计算机制造商随计算机一起提供给用户。

10. 作为产品管理和推广应用的一项措施，军事、工业和标准部门将迅速进入合格软件领域，从而使分摊到每个用户身上的价格大为降低。

11. 微处理器价格低、体积小、功耗少。微处理器必将使那些带计算机的、生产量大的产品的种类大大增加（目前属这类产品的只有袖珍计算器和数字电子表）。

---

1) IBM360系统上用Fortran和PL/1语言的科学子程序包(360A-CM-074)。

2) Cowell(1972)。

3) Numerical Algorithms Group, Ltd. NAG Central Office, 13 Banbury Road, Oxford, Ox26NN, United Kingdom(一个英国大学间的合作项目)。

4) 见Rosen著作(1967, pt.2, p.29—34)，同见"Cobol 1961"。

5) 磁盘子系统将迅速降价，预计600兆字节单元的价格，将从1981年的39,000美元下降到1989年的3,000美元(见Isaacson, 1979)。

6) 如果我们研究Pbister数据，可以发现轿车的年销售量自1945年后基本上一直呈直线式上升，至1975年为240亿美元。而计算机的销售量和年收入自1955年后一直呈指数式上升，1965年为50亿美元，1975年为300亿美元(见Pbister著作，1974，图1.1.5和图1.20.2)。