软件维护

——问题与解答

〔美〕 詹姆斯·马丁
卡尔马·麦克克劳埃 著

机械工业出版社

软件维护

——问题与解答

詹姆斯·马丁
〔美〕 著
卡尔马·麦克克劳埃

谢莎莉　文胜利　薛　非　译

柯秉衡　校订

机械工业出版社

本书较全面系统地介绍了软件维护技术，共分为六篇二十三章。第一篇导论，介绍软件维护技术的概念和现状。第二篇维护设计，介绍程序可维护性的度量标准和建立方法、数据库的重要性、数据的规范化、稳定数据结构的建立、文档和源代码风格。第三篇介绍方法论的革命，包括结构程序设计方法、第四代语言的使用、用户驱动的计算技术、原型设计和信息工程。第四篇软件包，介绍预先编制的应用软件包和可维护性软件合同。第五篇介绍执行维护功能，包括执行程序维护、纠错、维护工具、维护功能的管理和维护小组。第六篇展望未来，介绍未来维护的计划和战略计划与转移。

本书对从事软件设计、开发、维护的科技人员，程序员和计算中心的各级管理人员以及有关专业师生是一本有价值的参考书。

Software Maintenance

The Problem and Its Solutions

James Martin and Carma McClure

Prentice-Hall, Inc., 1983

• • •

软 件 维 护

——问题与解答

〔美〕 詹姆斯·马丁
卡尔马·麦克克劳埃 著

谢莎莉 文胜利 薛 非 译

柯秉衡 校订

•

责任编辑：邱锦来 责任校对：刘绍曾

封面设计：姚 毅 版式设计：张世琴

责任印制：张俊民

•

机械工业出版社出版（北京阜成门外百万庄南街一号）

（北京市书刊出版业营业许可证出字第117号）

中国农业机械出版社印刷厂印刷

新华书店北京发行所发行·新华书店经售

•

开本 787×1092 1/16 · 印张 16 3/4 · 字数409千字

1990年5月北京第一版 · 1990年5月北京第一次印刷

印数 0,001—2,850 · 定价：14.00元

•

ISBN 7-111-01948-2/TP·107

译者的话

在软件生存期间，维护阶段（即使用阶段）占67%的时间，是软件创造价值的唯一阶段。为了尽量发挥其应有功能和延长使用寿命，并使之创造更多的价值，软件维护是极其重要的。可是，人们的注意力常常集中在计算机本身，硬件需要维护，这是众所周知的事。对于软件维护的重要性直到70年代中期才为人们所认识。软件虽然不会用旧，也不会"磨损"，但同样需要维护。当软件投入运行后，为要解决发生的各种故障（错误），增强其功能，使之适应新的环境等活动，软件维护的工作量是相当大的。对一个软件开发机构说来，它为了维护已推广的软件，就再也没有力量开发新的软件了，这使一些软件开发机构伤透了脑筋。既然软件维护的工作量这么大，那么，什么是软件维护，软件维护的方法以及如何减少维护的工作量等问题就提出来了。

鉴于世界上关于这方面的研究较晚，有关软件维护的文献较少，提出的技术手段或方法还不太多，而本书属于目前软件维护技术方面较全面、较系统的著作，所以我们翻译了此书。

全书分为六篇，共二十三章。介绍了有关软件维护和开发技术中的一些重要问题及其解决方法，对近几年来这一领域里的某些新趋势也作了简要阐述。书中系统地描述了软件维护的有关概念和现状、软件维护的目标和内容、软件维护的技术、软件维护的主要步骤和软件维护过程中必须注意的问题。

本书内容深入浅出，通俗易懂，理论联系实际，并在各章末尾推荐了有关文献，以供读者进一步研究参考之用。

本书由柯秉衡副教授校订。第一到第九章由谢莎莉同志翻译；第十到第二十三章由文胜利同志翻译（其中第十二章、第二十章和第二十一章由王世正、王国基同志翻译）；薛非同志翻译了原序，并对全书逐字逐句斟酌，负责全书的整理工作。

限于译者水平，错误和不当之处在所难免，恳请读者批评指正。

译　者

1988年 2 月

原　序

软件维护耗资极大，是软件生存期间花费最多的一项工作。然而，系统分析及设计方面的论著和教程比比皆是，软件维护这一十分重要的问题却几乎被人们忽略了。人们对如何应付已经很严重的维护问题茫然不知所措。

事实上我们有很多事情可以做。广为采用本书中所讨论的技术，将使现今大多数单位的软件维护费用削减到只有原来的几分之一。

为了解决软件危机，软件的开发、使用及维护必须简化和自动化。软件技术以往大都集中在软件生存期的开发阶段，即程序设计方法和工具。随着60年代初期高级程序设计语言的开发，软件方面最主要的进展是软件工程的提出，特别是结构技术的问世。遗憾的是，我们低估了改变程序的需要及其困难。

设计的系统不具备可维护性，其代价是相当高的，而这常常只是指在明处的维护费用。还有一种潜在的费用更高。这就是系统变得如此脆弱，以致程序员及管理员不愿意改动它，因为任何一点改变，都会带来意想不到的后果，这种后果常引起系统其他部分出现问题，使用户烦恼，并耗费软件人员的宝贵时间。

软件维护的一个基本问题是，当用户要对系统作一点改动时，常常产生一些预见不到的副作用。排除一个故障很可能导致出现另一个新的故障。

虽然一些最不美妙的维护记录令人望之却步，但我们也应该看到某些系统甚至包括一些复杂系统的维护得到了控制。为了控制维护，需要采取各种不同的开发和维护措施，其中大部分措施在系统开始设计时就要实行。

要使维护得到控制，我们认为，对于本书中所讨论的维护中的问题，数据处理组织所有成员——程序员、分析员、系统设计员和管理员都应该弄清楚并应用各种不同的方法和工具加以解决。了解所有这些方法和工具并在建立任何一个系统时相应运用，这对一个系统开发员来说是十分重要的。

因此，本书是一个十分令人鼓舞、内容相当丰富的论著。它几乎涉及到数据处理的所有方面，从编码技术到战略计划，从程序员到计算中心领导人，应有尽有！

目　录

第一篇 导 论

第一章 维护混乱的现状

引论

我们正在加速进入一个计算机时代。从商业刊物到时装杂志，有关计算机应用的文章比比皆是。到80年代末期，随着计算机在诸如电子资金汇兑、电力设备维护、最新式武器系统、自动工程设计以及机器人生产线等许多方面应用的急剧增加，行政人员、秘书及消费者都将把使用计算机作为他们日常活动的一部分。

然而，当最高级事务管理人员要查询他们认为已在计算机中的信息时，却往往得不到。当董事们想改变某一过程时，他们得到的答复却是不能改变，因为计算机不能做出这种改变。政府规定有个什么改动，一些大保险公司不得不借助手工运算去处理索赔案子。问题在哪里呢？

问题在于已经建立的计算机程序很难维护。

什么是维护

在计算机程序交付买主或用户使用以后，对其所作的修改即所谓“维护”。进行维护的原因多种多样：

1．纠正错误和设计中的缺陷

2．改进设计

3．变换程序，以适用于各种不同的硬件、软件、系统特性、通信设备等

4．处理程序之间的接口

5．改动文件或数据库

6．扩充功能或对于不同应用做相应变动

程序维护不同于硬件维护。计算机的硬件维护包括替换损坏部件、纠正缺陷、加强设计以及润滑和清洗机械零件等，这些都不会影响计算机设计的功能。因此，对用户来说计算机没有什么变化。

程序维护不仅仅是纠正缺陷和增强设计，还包括增强程序的功能。用户不断要求调整程序的功能。大多数维护工作都是出于这些要求而不是由于可靠性问题而进行的[1]（见图1-1）。

工业上使用的软件系统要经常加以修改，以适应不断变化的数据，满足用户不断变化的需求。甚至一个完全可靠、完全满足用户要求、结构也很好的系统在维护阶段也常常要作修改。除非未来的软件系统设计得更容易修改而又不影响其质量，否则这些系统的维护仍将耗费大量的时间和资金。

同硬件维护类比，把“维护”一词用于扩充程序功能似乎不大合适，也许用另外一个词更确切些。然而，现在“维护”这个词已经广为使用，意指以各种形式加强软件。当我们谈到“技术性变化”或是扩大应用时，设计易于维护的软件指的是同一个问题。

无法满足应用上的需求

计算机已大幅度地降价。显然，随着各行各业的使用，批量生产还会使价格继续下降。什么能够减慢计算机更新浪潮的势头呢？当然是设计不良的软件。

“软件”已经成为实现计算机化的一个决定性因素。许多单位的计算机化能否成功取决于软件的支持。程序供求之间的差距正在迅速扩大。目前大多数公司都积压了3～4年的需要进行程序设计的计算机应用项目。

另外，还有一种“不可见积压课题”没有正式列入迫在眉睫的应用项目之列。斯隆学院情报系统研究中心 (Sloan School Center for Information Systems Research) 的调查发现，所调查的主要企业的“不可见积压课题”平均为申报的164%。这次调查所得出的积压课题总数相当于全部已装机使用的应用软件的179%[2]。

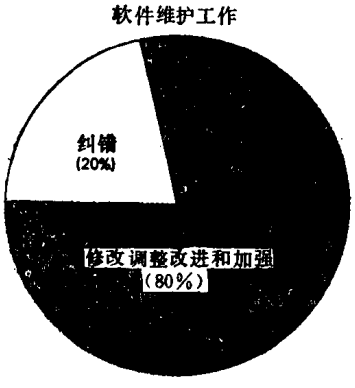

图1-1 大多数单位的主要维护工作都花在加强和改进软件系统上

程序设计费用在计算机成本中所占比例日益增大。大多数软件无论其工作量大小，都会碰到许多问题，甚至失败。程序设计工作的完成时间往往比原计划的时间要长，其实际费用也往往比原计划的要高。如图1-2所示，实际费用超过预算的300%，实际进度超过估计的200%的情况屡见不鲜，已不再是什么例外了[3]。经理人员们对不能把正规的管理方法应用于数据处理而感到为难。用户对于应用软件难以改变，又达不到预期运行效果而感到失望，怨声怨气。软件专业人员对于为什么一个项目成功了，而另一个却失败了，感到迷惑不解。

目前许多国家和地区的稳定的经济正受到生产率滞后的威胁。实现计算机化也许是降低生产和办公费用的最佳途径。计算机辅助设计及辅助制造正在加速采用。文书工作程序也需要做大的改变。在程序员和分析员日益紧缺的同时，这些必要的改变要求迅速增加程序的生产。《华尔街》日报(Wall Street Journal) 某日头版文章中谈到：“石油和软件是经济发展的两个主要障碍”[4]。

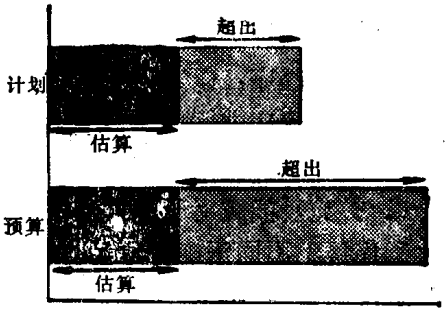

图1-2 软件项目中实际进度超过估计200%，实际费用超过预算300%的实例已经司空见惯而不例外

软件技术的发展并没有跟上硬件技术的发展。一些买了最新型计算机的计算站进行程序设计时，用的还是已经用了20年的语言，其方法则与工程训练方法没有什么联系了。

缩短软件供需之间的差距

缩短程序供需之间的差距对于未来计算机化是十分重要的。要解决软件危机，开发、使用与维护程序的工作必须加以简化和自动化。

软件技术以往大都集中在软件生命周期的开发阶段——简单地说，就是程序设计方法和工具。随着60年代初期高级程序设计语言的开发，软件方面最主要的成就是软件工程的提出，特别是结构技术的问世。

同60、70年代一样，80年代的软件技术也大多集中在软件开发技术方面，而往往忽视终端用户、管理及维护方面的问题[5]。然而，再注意一下你就会发现，后一方面比前一方面的

技术问题更多。使用户不满意的不仅仅是因为系统故障和失灵，还因为文档短缺、训练不适当以及程序无法适应用户不断变化的要求。

过去软件的缺陷以及当前成本发展趋势可以说明：把注意力集中在技术和开发方面的软件方法的有效性是值得怀疑的。这些现象固然是值得重视的，但并没有直接指出软件危机的主要原因。对现行软件系统的维护消耗了许多本来可以用于新的开发项目的宝贵资源，这就是等待编程的应用项目积压越来越多的主要原因。从工作量和费用角度来看，维护工作主导着整个软件生命周期（见图1-3）。

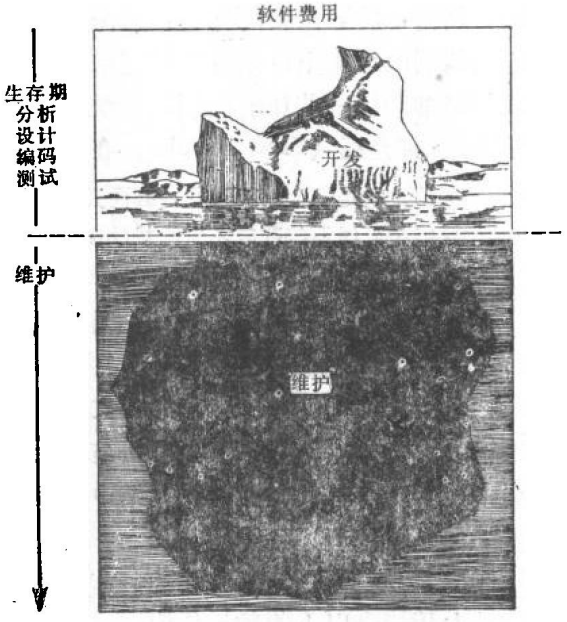

图1-3 软件维护决定着整个软件生命期。在许多单位，软件维护工作经费消耗了整个生命周期费用的3/4，占用了一半以上的数据处理人力资源

我们过去严重低估了改变程序的需要及其进行改变的困难。

时机代价

设计系统不考虑维护，其代价将是非常高的，这常常还只是指的维护的表面费用。还有一种潜在的花费常常更高。由于系统如此易于损坏，以致数据处理管理人员不愿去改动它。任何一种改变常常会引起无法预料的后果，导致别的地方出现问题，令用户烦恼，消耗宝贵的人力资源。

因此，企业经理人员被告知："你不能做这种事情，计算机处理不了。"甚至微小的改变也不行。经理们得不到他们作决策所需的信息，过程改进也不能进行，为顾客服务的更好方式也只好免了。企业本身本来就在迅速变化之中，但数据处理部门却步履维艰。精干的经理们也感到束手无策了。他们常常想进行一些变革，可是做起来困难越来越多，就好象在慢慢凝固的胶水中游泳一样。

计算机提供了大大提高企业效率的希望，但是这种希望的实现，取决于能否采用可维护性的最佳技术。

连锁反应

程序维护的一个基本问题是：进行一次修改常常会产生难以预料的副作用，排除一个错误很可能引出一个新的错误来。

一个地方的改变常常影响整个系统，这种影响不是显而易见的。人们试图用最小的工作量进行一个局部变动，但往往引起别处一连串的反应。除非该系统的文档很完整，否则，这些副作用直到它们在运行中造成了麻烦时才可能被察觉。

由于维护人员或修改人员往往不是写源代码的本人，情况就变得更加糟糕。修改可能影响到多个编码员的工作。

由于副作用，维护比其它编程工作更加需要测试程序的每行代码。进行一次修改，可能需要运行一整套检查事例，以确定其它部分工作是否仍然正常。这种检查副作用的"回归测试"可能是昂贵的。昂贵的程度取决于该系统结构的复杂程度。大型复杂系统中连锁反应的

影响可能非常严重。

维护修改很可能使程序结构变坏，使程序变得更为复杂，并使下次维护修改更为困难。维护常常是在危机情况下进行的，这时需要的是快速“打补钉”，而不是精心重新构造。补钉越积越多，就在补钉上再打补钉。

随着纠错，将引出新的错误，越来越多的时间花在解决这些二次问题上，而不是在改正产生原始问题的结构上。系统变得越来越紊乱，有些复杂系统甚至达到了无法维护的地步。每一次纠错都产生一些新的问题。系统变得如此不稳定，以致无法进一步扩充。

弗莱德·布鲁克 (Fred Brooks) 在总结建立OS/360的管理经验时写道：

系统程序的建立是一个熵降过程，因而天生地处于亚稳状态。程序维护是一个熵增过程。即使用最巧妙的方法也只能是延缓系统变成不堪维护的废物的衰退进程[6]。

莱赫曼和勃莱迪 (Lehman and Belady) 研究过一个大型操作系统的相继出版的历史记录[7]。操作系统越来越大，模块数随着版次的增大呈线性增加，而受到维护修改影响的模块数随着版次的增大呈指数增加。

在整个数据处理过程中，程序和文件的数量不断增加。维护费用可能变为数据处理预算中越来越大的部分。除非采取强硬措施，否则将无法控制这种局面。

图 1-4 表明在有代表性的计算站中，维护占数据处理预算的比重越来越大的情况。在许多数据处理单位，程序维护活动几乎占软件生命周期全部费用的3/4[8]，并占用一半以上的数据处理人力资源[1]。最早使用联机和交互式系统的那些单位，常常把80%的时间花费在维护上。由于使用了本书所讨论的某些技术，在一些单位，这个数字已降低到大约20%。

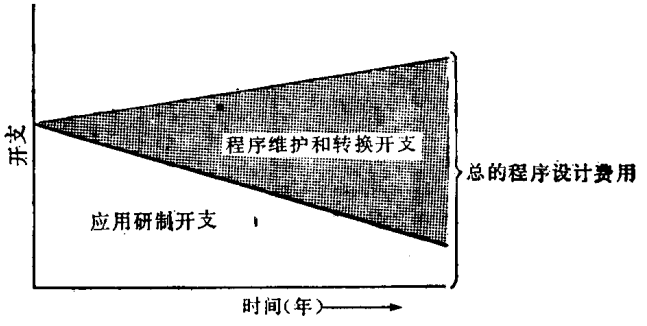

图1-4 常常由于修改现行程序和文件的费用增加而延迟了发展新应用软件。某些公司花费了程序设计预算的80%以上来维持现状，只剩下不到20%的预算用于发展

有些单位甚至出现100%的工作量都花在维护现行程序上的状况，使得新的应用程序无法编写。有一个大的政府机构，当它要重新设计其数据结构时，冻结了所有的应用开发工作达一年半之久。这种状况令终端用户难以容忍，他们常常想法绕过数据处理部门。

文件问题

许多数据处理单位已经在磁带或磁盘上建立了大量的文件库。当业务发生变化时，常采用某种方法修改程序，这种方法会引起记录结构的变化。遗憾的是，还有另外的程序也使用这些记录，这些程序意外地不得不因此而修改。在一个老计算中心中有许多其它程序使用该文件，也不得不都进行修改。

文件环境下的任何一点似乎微小的变化都将引起其它地方不得不进行修改的连锁反应。这种大变化是很费钱的，原来的程序员又正在做其它工作，但修改又非他莫属。有时修改是很困难的，因为应用程序没有足够的文档。

随着时间的推移，这个问题变得更为严重，因为建立起来的程序越来越多，每修改一个文件要修改更多的程序。

系统分析员及数据处理员常常认为：现行运转良好的程序可以不管了。然而事实上，他们建立或使用的数据，其它应用也需要，而且其需要形式总是略有不同。如增加新的数据项类型，用前几个记录中的数据项类型建立新的记录类型，要求数据必须以不同的方法索引，改进数据的具体格式，归并不同应用的数据等等。

一个老的文件环境就象一碗面条，每拉出一根面条就会牵动碗中其它所有面条。时间越长情况就越糟。因为面条根数增加了，它们交织得更复杂。维护文件系统的难度随已产出的应用程序数呈几何级数增加。

正如我们下文要讨论的，数据库技术正是为解决上述问题而发明的。在数据库管理得很好的计算站中在相当大的程度上取得了成功，而在数据库管理得不好的计算站中问题变得更加糟糕。

对于某些单位来说，维护混乱已变得不可捉摸。想一下20年以后应用及系统开发愈来愈多时的情景真令人担心。微生物在适当的条件下，会按指数率倍增，正象今天的计算机一样。但是，如果把它们放在实验室一只密封的盘子里，它们终将淹没在自己的粪便之中。一个数据处理部门领导人曾用此来比喻自己的维护问题。他说，新的发展将被陈旧的COBOL和PL/1系统的“粪便”所窒息。如果不在维护及开发环境中广泛采用和吸收新技术的话，程序设计人员终将被淹没在自己编写的程序的维护之中。

最近一项DOD研究表明，空军飞行控制系统软件研制费用是每一条指令75美元，而维护费用每一条指令高达4000美元[9]。

弗莱德·布鲁克讽刺地引用克·斯·莱维斯（C.S.Lewis）的话[6]：

> 这就是历史的关键。花费了巨大的精力建立了文明社会，设计了绝好的机构。可是每次都出点什么错。致命的缺陷总是把自私和无情的人引向顶峰，然后全都陷入悲惨和毁灭。事实上，是机器出了毛病。它在起动时似乎还好好的，也跑了几步，后来就坏了[10]。

取得控制

虽然最坏的维护记录如此令人望之却步，但我们也应该看到某些系统甚至一些复杂系统的维护得到了控制。为了控制维护，需要采取各种设计及管理措施，其中大多数措施在系统初始设计时就应予以实行。

问题主要在于尽管维护费用高且有破坏作用，但在许多单位中，仍然没有正确地对待维护。在新系统最初构思时就没有考虑如何使系统设计得易于维护。习惯上总是将维护与新的开发分开个别处理。搞维护的人通常是一些不同于搞开发项目的而且经验较差的人，这两组人员很少互相联系。在搞数据处理的人们中间，看不起维护工作，维护被看成是一个很少有机会使用新技术的、没有挑战性的环境。人们认为用现代软件技术进行维护是不适宜或不需要的。

程序设计的巨额投资

政府及各公司所依靠的计算机程序代表了亿万美元的人工工作量。这些系统不能轻易被新技术所取代，因为投资太大了。由于还能适当地工作并且也非完全不能为用户所接受，从投资效益上来说，现在还不到替换的时候。此外，为了避免中断用户服务，任何一种替换都必须仔细地计划，使得旧系统逐步地进化成新的系统。现在有一些较新的软件技术能够用来建立起较好的系统，但这些先进技术将需要较长一段时间才能普及。70年代早期就已产生的提高程序设计效率的辅助工具，例如结构技术、测试数据发生器、数据库设计技术及数据词

典等在许多计算站中仍然没有得到广泛应用。

在今后许多年，现有程序仍旧是一个维护负担，在程序设计的投资仍然继续增长的情况下，我们不应该在今后为失去控制的维护付出日益增长的费用了。

日益紧迫的压力

如果一个数据处理经理人员不采取坚决的措施去控制维护，它将成为一个更严重的问题，除非细致地将维护纳入控制之下，否则维护会变得更糟。计算机的成本将继续下降，用户对如何使用计算机的知识也在不断增加。开发新的应用软件的要求将以更快的速度增加，同样，修改老的应用软件的压力也在增大。人们要求在价钱更便宜的计算机上开发新的应用软件，同时要求维护老的应用软件，数据处理经理人员被夹在中间，压力越来越大（见图1-5）。

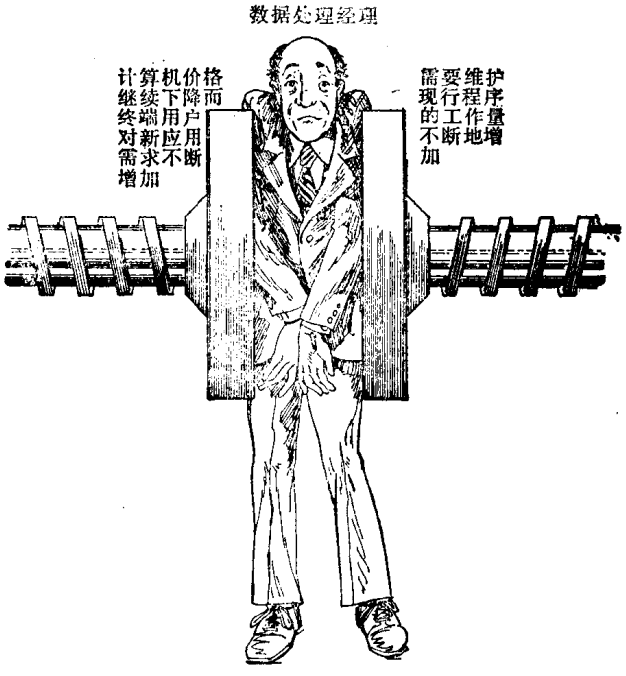

图1-5　数据处理经理在夹缝中

为了使维护得到控制，我们认为所有数据处理人员——程序员、分析员、系统设计员和经理人员都应该理解和运用本书中所讨论的各种解决问题的方法。

维护的解决方法

本书讨论了解决维护问题的多种方法。希望系统开发人员弄清所有这些维护方法并在建立系统时适当地加以应用。

必须在系统最初计划时就考虑到设计易于维护的问题。设计系统和编制程序时可以采用许多措施使之日后具有可维护性。对维护设计不好的系统，日后也不好改进。相当多的单位在设计时没有充分考虑维护问题。

我们需要把像操作系统这样复杂的软件的建立和商业数据处理软件的开发分开。减少后者的复杂性是比较容易的，并可以采取许多强有力的措施使未来的维护成本尽可能降低。

下面是其中的一些处理方法：

1．一开始就要以将维护减到最少的指导思想来进行系统设计和程序设计．采取那些利于可维护性的管理方法．

2．最大限度减小系统及程序结构的复杂性．

3．选择和采用易于维护的结构技术．

4．通过提高软件的可靠性来减少对纠正性维护的需求．通过加强计划和控制用户扩充软件来减少修改性软件维护的需求。

5．预期今后将转向哪些新技术或软件，做出计划，力求使重写程序的需要最小．

6．保证文档及结构图表完整清楚，便于以后的维护人员使用．

上面列举的方法适用于所有软件开发。下面列举的是特别适用于商业数据处理的方法：

1．使用最适宜于将今后维护减到最少的数据库软件．设计并管理好数据库环境，从而建立尽可能稳定的数据库．

2．采用能够使修改容易进行并迅速完成的第四代语言．例如报表生成器、应用程序生成器，高级数据库语言以及第四代程序设计语言．在有可能的地方使用非过程语言．

3．运用能够简化数据库过程建立的结构技术．

4．训练用户学会编制他们自己的报表和使用数据管理软件，保证用户能够对他们用这些工具建立起来的应用软件进行增强性维护．

5．选择应用软件包时，要十分注意其可维护性．

6．使用外部软件并把维护的责任交给供应者．

7．鼓励程序开发人员、维护人员及用户之间互相沟通．

8．使用有助于维护的工具(例如原型生成工具、结构分析器、数据词典)．

9．确认那些维护花费很大的老系统，用更新的技术重新开发（这些新技术包括数据库、第四代语言、更好的结构、软件包等等)．

在以后诸章中，这些基本方法将扩展为把维护的工作量和费用降到最小的一系列方法。这些解决方法既有长期的，也有短期的。我们将从管理和技术两个角度来讨论维护问题。还将提出改进新系统、现行系统、数据库系统及软件包的可维护性的一些方法。

小结

处理维护问题最有效的方法就是一开始就把可维护性设计作为系统的一部分。我们将讨论这样做的各种不同方法。然而维护人员常常碰到的是那些很久以前写的没有设计维护的程序。因此，本书将从下面两个方面来讨论：

1．维护现有程序，这些程序常常设计得很差，补钉不完善，文档短缺——不幸的是，大多数维护人员不得不来做这项工作。本书的第一篇集中讨论这项工作，其它章节也会涉及到。

2．设计易于维护的程序和系统。第二篇讨论进行这项工作的传统方法。第三篇讨论那些在软件或设计基本原理方面有重大变化的方法。第四篇讨论买进软件包问题。第六篇讨论如何制定远景计划以控制维护费用。

在写本书时，全世界每年在软件维护方面的花费超过300亿美元。如果凡是适合的地方都采用本书提出的技术的话，则这300亿美元中至少一半是可以节省的。

参考书目

1. B. P Lientz. E B. Swanson. and G. E. Tompkins, "Characteristics of Application Software Maintenance. *Communications of the ACM*, Vol. 21, No. 6 (June 1978): 466-471.
2. R. B. Rosenberger. "The Information Center," SHARE Proceedings No. 56, Session M372, March 1981.
3. L. H. Putman, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem," *IEEE Trans. on Software Engineering*, Vol. SE-4, No. 4 (July 1978): 335-3.61
4. W. M. Bulkeley, "Computer Makers Feel Key to Sales Edge Lies in Better Programming," *Wall Street Journal*, Sept. 29, 1980, pp. 1, 18.
5. P. Isaacson and E. Juliussen, "Guest Editorial: Window on the 80's," *Computer*, Vol. 13, No. 1 (January 1980): 4-6.
6. F. Brooks, *The Mythical Man-Month* (Reading, MA: Addison-Wesley Publishing Co., Inc., 1975).
7. M. Lehman and L. Belady, "Programming System Dynamics," ACM SIGOPS Third Symposium on Operating System Principles, October 1971.
8. "Software Prototyping," *DCAS Newsletter*, Vol. 3, No. 2 (June 1981): 1.
9. B. deRose and T. Nyman, "The Software Life Cycle—A Management and Technological Challenge in the Department of Defense," *IEEE Trans. on Software Engineering*, Vol. SE-4, No. 4 (July 1978): 309-318.
10. C. S. Lewis, *Mere Christianity* (New York: Macmillan Publishing Co., Inc., 1960), p. 54.

第二章　软件维护技术的现状

福尔曼制造公司

总部设在芝加哥的福尔曼公司是一家具有50年历史的制造公司，它每年的销售额达6000万美元。5 年前，福尔曼被一群有进取精神的年青经理人员买下，他们计划在 10 年之内使福尔曼的收入翻两番。他们要通过兼并一些小公司来实现这一雄心勃勃的目标。

两年前，福尔曼选中了科尔勃塑料制品公司作为它的第一个子公司。作为决策兼并科尔勃公司可能性的一部分，福尔曼的管理部门同本公司数据处理部门进行磋商。福尔曼有一个为全公司服务的中央数据处理部门，它包括一个数据处理经理人员、三个程序员/分析员、三个计算机操作员、一个穿孔员，中心内配有IBM4331计算机。福尔曼的大部分计算机化系统是批处理系统，例如工资管理系统、会计系统、材料需求计划系统及各种工程程序。

数据处理部门报告说，收并科尔勃需要立刻对现有程序做重大的修改。因为科尔勃使用九位数字的零件编号，而福尔曼的程序目前只允许使用六位数字的零件编号。福尔曼的程序必须修改以适应更大的零件编号。

福尔曼经营的业务易受存货的影响。因此，准确盘存是十分重要的。能够处理科尔勃的零件编号对于控制存货并确保新的子公司有利可图至关重要。虽然许多程序都受到零件编号

变更的影响，但数据处理部门确信这是一种简单变化，三个月之内就能容易地实现。

由于得到来自数据处理部门的保证和对科尔勃乐观的财政估计，福尔曼兼并了这家公司作为它的一个新子公司。遗憾的是，并非一切都是按计划进展的，一年后，数据处理部门仍在为实现原想三个月内完成的零件编号转换计划而奋斗着，所有新的数据处理开发都搁置了。大部分数据处理资源都花在编号转换上，管理人员不仅搞不清到底出了什么差错，而且连货存也控制不住。结果这一年科尔勃没有赚到钱，福尔曼的收入也大大减少。

福尔曼的管理部门决定放弃其野心勃勃的收并计划。受挫的责任在于福尔曼的软件无法帮助公司实现其长远规划。福尔曼的管理人员对此的抱怨已经不是第一次了。他们的软件已经多次被证明太缺乏应变能力以致无法满足公司变化着的需要。

中西部电话公司

TAS是一个已有20年之久的软件系统，由美国中西部电话公司用于一个城市主要城区的电话赋码。TAS是一个非常庞大的系统，由600多个程序将近20万行代码组成。TAS源程序主要是用汇编语言写的，但新增加的程序都是尽可能用 PL/1 语言写的。作为以新技术更新TAS的计划的一部分，去年完成了向IBM数据库管理系统——即IMS系统的转换。

运行和维护TAS的年预算约为500万美元，其中绝大部分用于维护。公司雇用了 50 名程序员和分析员对TAS进行连续不断的软件开发和要求性维护工作。此外，还雇用了20名电话服务支援人员，跟踪检查由于无效电话号码造成的TAS程序中断。每一个月大约有10000次程序中断，但只有很小一部分确实是由程序错误造成的。

近来TAS程序中断次数明显减少，未解决的程序问题积压也明显减少。1981年全年内，到每月月底尚未解决的已知错误不超过15个。大家认为TAS系统终于稳定下来了。然而，更准确的调查表明的原因却大相径庭而令人惊讶。这就是用户对TAS不能提供可靠的服务感到不满，他们回避TAS系统，回到了人工电话赋码。IMS 数据库渐趋过时，错码及电话中断的次数开始增加。

销售调查研究所

SMCS在销售调查研究所被看成是一个高质量的模型软件系统。SMCS 是一个用于向制造厂家报告杂货产品销售趋势的图形系统。它是在70年代中期采用结构技术开发的有 2 万行程序的 ANS COBOL 系统。SMCS不仅是按进度和预算进行开发的，而且已经被证明是一个非常可靠的系统。在 5 年使用期间SMCS只出过少数错误。

基于它的运行历史，可以设想SMCS 是可维护性很高的系统。当研究所的数据处理人员及程序员被问及SMCS 的质量时，他们会说这是一个结构很好且易于维护的系统。然而进一步的了解表明，维护成本低的原因并不在于此。研究所中一般的COBOL 程序员认为应用逻辑太复杂，难以理解。因此，SMCS的质量是通过打消用户的修改和扩充要求来保持的。

软件失败

福尔曼公司、中西部电话公司以及销售调查研究所的遭遇是很普遍的，它表明了提供工业适用软件的困难程度。但是，它们不是软件失败的例子，而是软件维护不可捉摸的例子。对许多单位来说，最严重的、花费最多的软件问题发生在维护阶段。

实践证明，福尔曼公司的程序太难修改，迫使公司管理部门改变它们的整个公司计划；中西部电话公司的程序错误百出，很不可靠，迫使用户又退回到人工系统；SMCS 系统太复杂以致难以理解，使程序员不愿做任何修改。

正象福尔曼公司的情况那样，由于影响到整个单位的管理和用户，维护问题所产生的后果波及之广，常常大大超过只是与数据处理生产率有关的范围。

大多数维护工作的根源是什么？实施维护时的主要困难是什么？使得一个软件系统的维护容易或困难的原因是什么？

最大限度减轻维护负担的第一步是弄清楚维护功能都包括哪些活动，哪些因素导致维护软件系统的困难和高成本。弄清了这些问题。才能选择更有效的技术和工具来控制和减轻维护负担。本章的目的就是要通过定义“维护”这一术语和描述当前维护活动及其存在问题使大家清楚这些问题。

维护工作的类型

斯旺逊（Swanson）把维护的起因分成三种基本类型[6]：

1．故障；

2．环境变化；

3．用户和数据处理人员的要求。

故障归因于程序错误，诸如无效输出结果、缺乏数据编辑校验、性能差、违反程序设计标准等等。

环境变化对于一个单位一段时间内使用的系统来说是常有的事。一般有两种类型的变化：一种是数据环境的变化，例如一个事务处理代码的改变，重新构造一个数据库等；一种是处理环境的变化，例如安装了新的硬件或新的操作系统。

用户和维护人员本身也是维护的一个原因。他们要求修改软件系统，以提高操作效率，增加新的功能，改变现有功能，提高可维护性等。

斯旺逊把由这些基本原因引起的维护活动归为以下三类[6]：

1．纠错性维护；

2．适应性维护；

3．完善性维护。

以上三种类型正在成为维护基本类型的标准分类方案。表2-1举出了一些例子。如表2-2所示，罗特（Reutter）提出了另一个类似的分类方案，但他把斯旺逊的三个类型细分为七个类型[2]。罗特分类重要的一点是增加了一类支援性维护。而斯旺逊把支援活动包含在每一个基本类型中。罗特把支援部分分出来，以强调用户和维护人员之间信息交流以及系统支援计划的重要性。

表2-1 维护活动的例子

纠错性维护
1．重置一开关以排除故障；
2．测试各种可能的条件，以排除故障；
3．处理文件中最后一个记录，以排除故障。
适应性维护
1．现行应用系统转到数据库管理系统(DBMS)的实现；
2．将指定代码从三字符修改成四字符；
3．调整系统，以减少响应时间；
4．把MRP系统从批处理转换成联机操作；
5．调整两个程序，使它们采用相同的记录结构；

（续）

6. 修改某一个程序，使之用于一个不同的终端；
完善性维护
1. 修改工资管理程序，以体现新的结算方法；
2. 在销售分析系统中增加一个新的报表
3. 改进终端对话，增强对用户的友好性；
4. 在一个报表中增加一栏；
5. 调整一个打印银行报表的程序，使之能在一种新设计的表纸上进行打印；
6. 改进图形输出；
7. 增加一条联机求助(HELP)命令；
8. 改善查询处理能力，使之能检查更多的数据类型；
9. 增加审查员所需的设备。

表2-2 维护工作的类型

斯旺逊的分类	罗特的分类
纠错性维护 ·识别并纠正软件错误，改正性能缺点，排除实施中的故障	紧急维修 ·当需要立即修复以继续用户服务时进行。 纠正编码 ·正确反映规范要求或正确利用系统资源。
适应性维护 ·使软件适应数据要求的变化或处理环境的变化	更新 ·使软件适应处理要求的变化 条件变化 ·适应由于管理环境或其它环境超出组织控制所造成的业务条件的变化。 增长 ·适应数据要求的变化或新程序、新用户的增加。
完善性维护 ·增强性能，提高投资效益，提高处理效率，改进可维护性。	加强 ·响应用户对系统进行改动或添加的要求。 支援 ·说明系统能力，计划未来支持，测试性能。

维护调查

对于影响应用软件维护的有关基本论点及问题我们已经提出了许多看法。例如，维护是一件人人都不喜欢但又避免不了的讨厌事情；维护工作得不到重视和承认；新的软件技术不能应用于维护。

这些观点是否反映了维护的真实情况呢？为搞清软件维护的状况，最近做了几项调查和研究[9-12]。我们将利用这些调查中的发现来剖析工业当前所面临的维护问题。

像维护人员的自信心、维护人员的调动、系统可靠性差以及不适当的文档等问题常常被认为是维护的主要问题。在1981年国家计算机会议上，扎宾（Chapin）把维护不受重视、文档差、维护人员缺乏经验作为维护的严重问题列举出来[10]；罗特认为对维护功能重要性缺乏认识以及维护管理不善是维护问题的症结所在[2]；莱昂斯（Lyons）指出软件不具备被改变而不破坏其结构与质量的性能是维护问题的主要原因[4]。总的说来，人们都把不断的用户要求列为首要的维护问题，软件不灵活、脆弱问题被列在第二位。

有关减轻维护负担的方法要包括管理及技术两个方面。

系统和程序设计资源中用于维护的比例多大？

大量的系统和程序设计资源要用来维护软件系统，工商界已开始对此感到不安了。人们