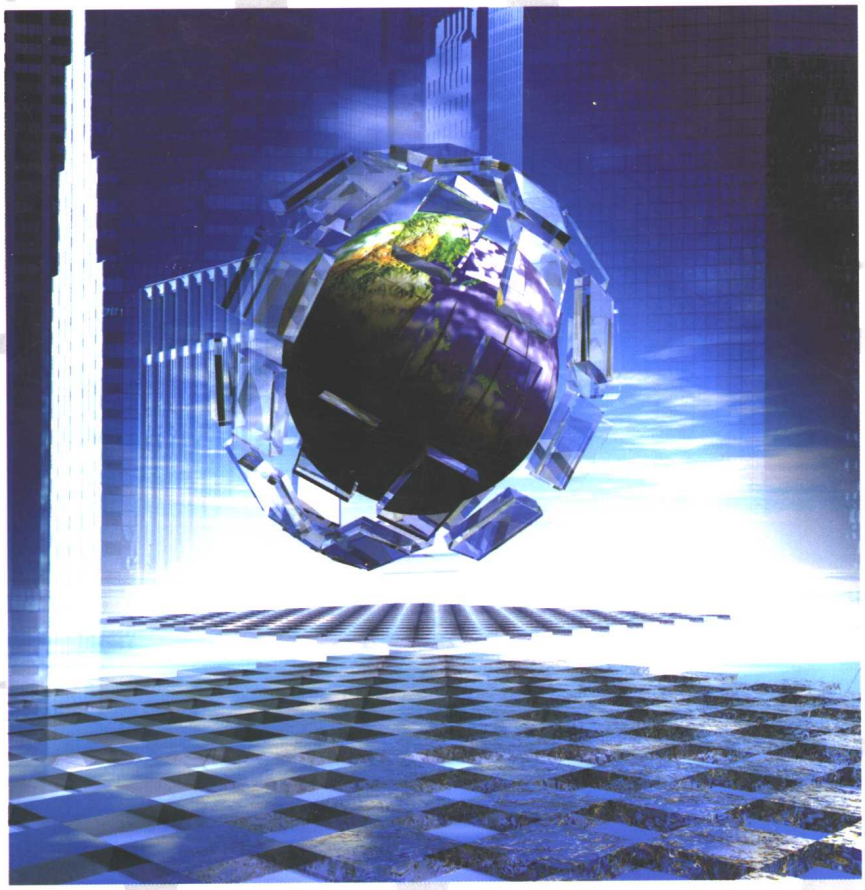
Software Maintenance: Concepts and Practice Second Edition

国外IT精品丛书



软件维护：概念与实践

（第二版）

☆ 全面一致的软件更改概念
☆ 控制和管理软件更改所需的理论基础
☆ 理解和运用当前维护手段及方法解决问题的框架

[美] Penny Grubb
Armstrong A Takang 著

韩 柯 孟海军 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Software Maintenance:
Concepts and Practice Second Edition

软件维护：概念与实践

（第二版）

〔美〕 Penny Grubb
Armstrong A Takang 著

韩 柯 孟海军 译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

软件维护是在软件产品交付之后进行的修改工作。软件维护的目的是修改缺陷、提高性能或其他属性，或使该软件产品适应经过修改后的环境。本书阐述了有关软件维护的基本概念、基本原理、基本方法，并全面论述了软件维护的影响因素和控制、管理方法。本书给出了大量的案例研究和思考题、练习题，有益于读者轻松掌握软件维护所涉及的内容。

本书不仅适用于高年级大学学生学习，也适用于各类软件工程人员阅读。

版权贸易合同登记号　图字：01-2004-0554

图书在版编目（CIP）数据

软件维护：概念与实践（第二版）/（美）格鲁布（Grubb, P.）等著；韩柯等译.—北京：电子工业出版社，2004.3

书名原文：Software Maintenance：Concepts and Practice Second Edition

ISBN 7-5053-9628-5

Ⅰ. 软…　Ⅱ. ①格…　②韩…　Ⅲ. 软件维护　Ⅳ. TP311.53

中国版本图书馆CIP数据核字（2004）第005120号

责任编辑：马振萍

印　　刷：北京天竺颖华印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路173信箱　邮编：100036

北京市海淀区翠微东里甲2号　邮编：100036

经　　销：各地新华书店

开　　本：787×1092　1/16　印张：16　字数：310千字

印　　次：2004年3月第1次印刷

定　　价：26.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：010-68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

致　谢

献给我们的家人，特别是：

- George和Danny（感谢他们所做的校对工作）
- Ayem、Bessem和Nyente

我们要感谢全世界各种卫生医疗机构中的同事和朋友，他们的经历和问题形成了本书所使用的“ACME医疗诊所”案例研究。

我们还要感谢Steven Patt及其同事在文字编辑方面给我们的协助。

译 者 序

在软件生存周期的各个阶段中，需求获取、软件测试和软件维护是公认最容易被忽视的活动。软件设计和开发具有很高的团队和个人自由发挥空间，能够给人们带来更高的成就感，能够得到管理层的重视，也更能够吸引高素质的人才。但是需求、测试和维护则不同。许多机构在需求上的投入往往不足，需求没完成就常常很快转入设计开发阶段。而软件测试和维护工作的艰巨性和重要性常常被严重低估。软件测试和维护工作的开展常常非常被动，团队和个人都缺乏足够的热情，缺少必要的资源。

不过近年来，需求获取和软件测试开始受到国内公司的重视。因为软件项目失败或不够理想的很大一部分原因是需求和测试没做好。需求工作没有做好，会导致软件的有用性降低，而软件测试又是保证软件质量的重要活动。但是软件维护与需求获取和软件测试还有不同。尽管软件维护在软件总成本中占很高的比重，但是由于国内目前很多机构的软件开发能力还不够理想，文档、配置管理、风险分析、软件重用、逆向工程、软件度量等方面的活动都不很到位，软件维护往往不得不由系统的原始开发人员，也就是机构中的“精英”实施，维护活动也主要是纠正性维护。而这时这些人员往往还承担了新的开发任务，有巨大的工作压力。他们往往把兼职性的老系统维护看做是一种负担。很难设想在这样的情况下能够比较理想地实施必要的维护。

本书是一本有关软件维护的教材，全书结构组织和论述都很严谨，强调基本概念、基本原理、基本方法的阐述，全面论述了软件维护的影响因素和控制、管理方法，给出了比较多的案例研究和思考题、练习题，引用了大量参考文献，并对重要参考文献给出了简要提示。因此本书不仅适合大学高年级学生阅读，也非常适合各类软件工程人员阅读。

在翻译过程中，我们力求忠实原文。但由于译者的知识水平和实际工作经验有限，不当之处在所难免，恳请读者批评指正。参加本书翻译、审校和其他辅助工作的还有：黄慧菊、屈健、刘芙蓉、王威、李津津、原小玲、韩文臣、孟海军、张红旭、付程、杜旭涛、朱军等。

前　言

目的与目标

撰写本书的目的是，解释支撑软件更改方面的关键问题，并讨论这些问题怎样影响软件系统更改的实现。由于人们需要一本能帮助软件工程师直接处理他们在修改复杂软件系统时所面临的问题的教材，所以促成了本书的诞生。这类问题的影响可以从修改软件的成本得出。软件修改成本可以占到软件整个生存周期成本的70%[4、36、176]。软件维护被认为是软件工程的一个关键领域[9，163]。尽管人们对软件维护有如此重要的认识，但是很多主流软件工程课程都偏重讨论新软件系统的开发，没有讨论这些系统投入运行后围绕系统修改所产生的问题[70]。

我们提供包含以下内容的教材：

- 全面一致的软件更改概念。
- 影响、控制和管理对不断改造的软件的软件更改系统所需技能的理论基础。
- 理解和运用当前维护手段及方法来解决问题的框架。

本书不是一本菜谱，没有处理软件维护问题的现成规则。在一种情况下是恰当有效的解决方案，对于另外一种环境中的同样问题可能会完全不适用。因此，软件工程师必须对软件维护有深刻的理解，这一点很重要原因有以下多方面。首先，常识告诉我们，找出问题的解决方案在很大程度上取决于对问题的理解。第二，对支撑软件维护问题的认识，会有助于形成能够用于指导恰当支持工具开发的合适框架。这种框架还使研究人员能够确定潜在的研究问题，并对研究发现进行比较。

本书的读者对象

本书的读者对象是对开发和维护软件系统感兴趣的学生、学术研究人员和实际工作者。

本书既可以用做参考书，也可以用做软件维护、软件改进和高级软件工程一般理论的课程教材。本书还可以用做希望从事软件维护研究工作的人员的入门教材。

作为本科生课程教材，本书可使学生提高对软件维护问题的认识，例如开发适合软件系统进化趋势的程序的需求。这不仅为掌握学科知识打下基础，也为在商业社会中发展做准备。很多大学毕业生进入软件业的第一份工作就是维护现有系统，而不是开发新的系统[187、282]。此外，本书还可以作为其他本科生软件工程和程序设计课程的补充材料。

对于软件专业人员，本书定义了一些常用术语。这些定义很重要，因为人们大量地使用术语和行话[211]。此外，本书提供的案例研究和现实生活中的例子会有助于开展在职培训，或开设有关软件维护的进修课程。

本书的结构与组成

本书分成五个部分。

第一部分是软件维护的背景。这部分将介绍在其中实施维护时所使用的基本概念和框架。通过研究软件更改的基本知识引入深层理论，但也介绍了目前阶段现实世界中需要考虑的问题。第一部分将介绍如何建立软件开发和维护生存周期模型。

本书第二部分将讨论软件维护中要开展的活动，从理解要更改的系统开始，一直到实施具体的更改和测试修改后的系统，以及伴随这个过程的管理问题和决策问题。

第三部分将介绍有关整个过程、软件组件和软件维护的测量和评估手段，讨论如何实施跟踪、如何进行客观评估。

本书的前三部分介绍什么是软件维护和如何实施软件维护，这些内容构成系统可维护性的基础。

第四部分将研究如何运用以上内容构建更好的系统。

第五部分将讨论软件维护的研究领域和未来发展。

每一部分的开始都提供一些讨论要点，目的是引起读者对基本问题进行思考。

全书的练习既有有关本书内容细节的简单问题，也有复杂的角色扮演项目，要求读者把自己放在特定的软件维护环境中全面考虑具体问题。

全书将使用大的和小的案例，把所讨论的材料与软件维护现场会出现的问题联系起来。

目　录

第一部分　软件维护的背景环境

第二部分 维护期间的活动

第三部分 不断跟踪维护过程

第四部分 构建更好的系统

第五部分　未来展望

第一部分　软件维护的背景环境

概述

这一部分将讨论具体背景环境下的软件维护，目的是为查看软件维护过程中出现的问题的详细内容做准备，并提供支持读者构建更好软件系统的理论基础。

与所有学科一样，深刻理解基础理论以及这些理论所适用的背景环境，对于理解实际问题并运用理论解决问题是非常重要的。

在这一部分中，我们将介绍软件维护的基本概念和软件维护实施的整体框架，包括软件更改的基本概念和一些限制与约束，最后讨论维护过程模型理论。

基本概念

这一部分的基本概念研究说明，软件维护在软件工程中的地位和作用，重点突出了软件维护作为一种独立学科的特点，同时也讨论了为什么需要软件维护，以及如何有效地实施软件维护。

维护框架

这一部分的软件维护实施框架研究，将定律坚实地扎根于现实世界中，提出在实施软件维护期间必须考虑的各种不同要素。

软件更改的基本原则

这一部分的软件更改理论研究，使读者能够更深刻地理解软件系统可以实施和不可以实施的更改类型，采用更结构化和有效的方法解决更改实现中的问题。

约束与经济因素

研究理论是很有用的，没有理论知识就不能成为有效的软件维护人员。但是不能把软件维护看做是在真空中实施，或永远都可以毫无限制地得到能够使用的资源。要认识到经济条件和其他约束条件在做出恰当决策的过程中是至关重要的。

维护过程模型

实施软件维护的过程已经经过多年的改进，并可以对基层过程建模。理解实施软件更改所使用的模型，可以更好地掌握整体过程，因此有助于做出好的决策。

讨论要点

提出这些讨论要点是为了使读者能够思考和讨论软件维护的基本问题。提出的问题都要在本部分中讨论，但是如果读者能够在查阅文中的解答之前，围绕这些问题思考并尝试得出自己的结论，一定会有很大收益。

什么时候实现更改

请读者设想自己正负责一个大型软件项目，而且正要实施一项重要升级，并要支持在不同Windows平台上运行的三个不同的版本，以及少量用户仍然还在使用的一个DOS版本。有一些客户在一周的时间内找到你要求做以下更改：

- 一位DOS用户已经见到同事的Windows版本。他并不想升级自己的整个系统，但是喜欢一项具体功能。你能提供吗？
- 一位更热心的用户试用过最新Windows版本中的最新功能。出现了一个问题是数据库有可能被损坏。你能解决这个问题吗？
- 另一位Windows客户听说马上就要推出非常好的升级版本，但是看起来还要等很长时间。你能在当前的版本中提供少量额外功能吗？

请讨论以上提出的问题。在研究这些请求时，需要考虑哪些因素？哪些因素会使你同意请求，哪些因素会使你拒绝请求？对于每个请求，请设想一种自己能够同意请求的场景和一种自己必须拒绝请求的场景。对于每种情况，更有可能同意还是更有可能拒绝更改？

第1章　基本概念介绍

“从来就没有什么‘已完成’的计算机程序”

Lehman[169 第2章]

本章主题

1. 定义并介绍软件维护和软件改进。
2. 讨论软件维护和进化在更广泛的背景范围内的地位。
3. 讨论软件维护与软件开发的差别。
4. 概括为什么需要维护。
5. 给出实施有效软件更改所需的有关理论背景和关键技能。
6. 介绍构成软件维护的具体活动。

1.1 引言

传统上，软件系统交付之后对其实施更改的学科叫做软件维护。这里要从定义的角度研究软件维护，并讨论为什么需要软件维护。由于成本非常高，因此重视软件维护是很重要的。有很多问题，包括安全性和成本，都说明寻找减少或消除软件维护问题的方法是很紧迫的需要[211]。

在过去的几十年里，软件系统一直在广泛的工作环境中发挥作用。在日常运营中使用软件系统的各行各业数不胜数，包括制造业、财务公司、信息服务公司、医疗服务机构和建筑业[176、263]。人们越来越严重地依赖软件系统[69]，软件系统越来越重要，这些系统不仅要完成期望完成的工作，而且还要完成得很好。换句话说，系统的有用性是至关重要的。如果系统失去了有用性，就不能被用户接受，也就是不能被使用。

在今天的世界中，正确地使用软件系统，发挥系统的作用，可能是生死攸关的大问题。软件系统有用性中包含的一些要素是功能、灵活性、程序可用性和正确操作[170]。为了在系统生存周期内支持这些要素通常需要进行一些更改[256]。例如，为了满足性能改进、功能增强或处理系统中发现的错误的要求，可能需要进行一些更改[176]。

软件工程师所面临的最大挑战是，软件系统更改的管理和控制[131]。交付软件系统之后使系统能够正常运行所花费的时间和工作，可以清晰地说明这一点。有关软件系统交付之后对系统实施更改的特点和成本调查结果说明，软件系统整个生存周期总成本的大约40%到70%要用于软件维护[4、33、35、176]。

1.2 定义

改进：从较低级、较简单或较差的状态持续修改到更高、更复杂或更好的状态的过程。

可维护性：能够实施维护的容易程度。

维护：使某个实体处于维修、效率或有效性状态，防止失效或退化。

软件：将计算机变得对人们有用的程序、文档和操作过程[192]。

软件维护：软件产品交付之后的修改目的是，修改缺陷、提高性能或其他属性，或使该软件产品适应经过修改后的环境[272]。

1.3 基础

为了理解软件维护，需要清楚什么是“软件”这个词的真正含义。人们常常有错误的概念，认为软件就是程序[184]。这种错误概念会导致对包括“软件”在内的一些术语的错误理解。例如，在考虑软件维护活动时，往往只考虑对程序所实施的活动。这是因为很多软件维护人员更熟悉，或更确切地说，更多地看到的是程序，而不是软件系统的其他成分。对软件更全面的理解可参阅定义部分。

McDermid的定义[192]明确说明，软件不仅仅包括程序，即源代码和目标代码，还包括程序所涉及的各种文档，例如需求分析、规格说明、设计、系统和用户手册、设置并操作软件系统所使用的规程。表1.1给出了软件系统的组成部分，每个部分都给出了一些例子。

表1.1 软件系统的组成部分

软件组成部分	举例
程序	1 源代码 2 目标代码
文档	1 分析/规格说明：　(a) 正式规格说明

（续表）

软件组成部分			举例	
			(b)	背景图
			(c)	数据流图
	2	设计：	(a)	流程图
			(b)	实体-关系图
	3	实现：	(a)	源代码清单
			(b)	交叉引用表
	4	测试：	(a)	测试数据
			(b)	测试结果
操作规程	1	设置和使用软件系统所需的指令		
	2	应对系统失效所需的指令		

McDermid的定义不仅是软件系统的定义[264]，而且是一种被广泛接受的全面定义，也是本书要使用的定义。

人们公认软件系统的可维护性很难定义。可以测量系统的特定方面。例如，有多种不同的方法测量复杂性。具体的特性，例如可互操作性或是否遵守标准规定也很重要，但是并没有简单的可以计算的整体“可维护性因子”。不过了解使系统易于维护的特性和特征，则是优秀软件维护工程师的主要能力，也是使个人在商业企业能够具有更高评价的一个因素。人们越来越多地认识到软件维护技能的真正价值，软件维护人员做为软件工程团队的“精英”，地位接近软件开发人员。本书稍后还要深入讨论构成可维护性的不同因素，以及软件维护的“公共形象”问题。

读者在附录[209、70、6、176]中会找到软件维护的很多不同定义。有些定义采用狭义具体的观点，有些定义采用更一般的观点。采用更一般观点的定义，例如把维护定义为“软件系统交付之后所实施的所有工作”[70]，包含了所有内容，但是却没有说明维护的要求。而采用具体观点的定义虽然说明了维护的活动，但是面太窄。这类定义最典型的是：

- 修改程序缺陷观点——维护是检测并修改错误。
- 满足需要观点——维护是当运行环境或原始需求发生变化时对软件的修改。
- 支持用户观点——维护是对用户提供支持。

本书采用的定义来自IEEE软件维护标准“IEEE STD 1219-1993”。上一节已经给出的这个定义对维护做了分类，提供了全面的定义。