普通高等教育“十二五”规划教材
计算机类主干课程系列教材

软件工程（第二版）

钟 珞 袁景凌 主编

科学出版社

普通高等教育“十二五”规划教材
计算机类主干课程系列教材

软件工程

（第二版）

钟 珞 袁景凌 主编

科学出版社
北京

内 容 简 介

本书面向普通高校本科教学及软件工程技术发展的需要，主要围绕结构化软件工程和面向对象软件工程方法，以软件生命周期为线索，较为系统地介绍计算机科学技术和软件工程等相关专业必需的软件工程知识，并与时俱进地增加了面向服务软件工程的基础知识。主要内容包括：软件工程概述、可行性研究、需求分析、系统设计、详细设计、程序编码、软件测试、软件维护、软件开发实例、软件项目管理、面向服务的软件工程。本书概念清楚，内容丰富，并配有典型实例分析和开发文档模版，便于教学和学习。

本书可作为高等院校计算机科学技术和软件工程等相关专业的教材，也可供广大软件技术人员和计算机应用人员参考。

图书在版编目(CIP)数据

软件工程/钟珞，袁景凌主编. —2 版. —北京：科学出版社，2015.1

普通高等教育“十二五”规划教材

计算机类主干课程系列教材

ISBN 978-7-03-042593-5

Ⅰ.①软… Ⅱ.①钟… ②袁… Ⅲ.①软件工程－高等学校－教材 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字(2014)第 272425 号

责任编辑：张颖兵 杜 权/责任校对：肖 婷

责任印制：高 嵘/封面设计：苏 波

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

武汉市首壹印务有限公司印刷

科学出版社发行 各地新华书店经销

*

开本：787×1092 1/16

2014 年 12 月第 二 版 印张：15 3/4

2014 年 12 月第一次印刷 字数：370 000

定价：38.00 元

（如有印装质量问题，我社负责调换）

前　言

随着信息技术的高速发展，软件作为信息技术的核心起着至关重要的作用。软件工程是研究软件开发和管理的重要学科，是计算机科学技术及软件工程等相关专业的主干课程，也是程序员、系统设计师、系统分析员、软件测试员、软件管理者及项目决策者等必须具备的专门知识。面对信息化、网络化、智能化的需求，研究如何更快、更好、更经济、更智能地开发出相应的软件，并有效地实施软件项目管理，是软件工程领域所面临的难题。

计算机技术的飞速发展也促进了软件开发技术的深刻变化，使软件管理更加复杂。为摆脱软件危机，软件工程——从20世纪60年代末期开始迅速发展起来，现在已成为计算机科学技术的一个重要和独立的分支，以及各大高等院校的一级学科和重要专业。20世纪90年代以来，软件工程不仅从方法论的角度为开发人员和管理人员提供可见的结构和有序的思考，而且大量的成功软件总结出的设计经验，使软件开发人员可以充分利用设计模式、框架、部件等进行开发，并能将软件作为服务使用。软件工程的相关理论和技术，得到了不断的完善、广泛的应用。

本书特点在于理论、方法与应用相结合，不仅对软件的分析、设计、开发到维护过程，以及软件项目管理进行全面地讲述，而且配有丰富的实例和开发实践案例，并根据ANSI/IEEE的相关标准，提供了相应的开发文档模版。除了对传统的软件工程方法进行讲述外，还与时俱进地增添了如面向服务的软件工程方法等内容。软件工程在计算机专业及信息类专业的课程中占有非常重要的地位，对学生将来从事计算机研究、软件开发和软件项目管理等工作具有很好的指导作用。

本书的作者一直以来从事计算机科学的教学工作，积累了丰富的教学经验和教学心得，并有大量软件开发的实践经验，对软件工程技术及其发展有较全面的认识。

本书由钟珞、袁景凌任主编，钟饮任副主编，参加编写的有 邱奇志、袁胜琼、吕品。另外，刘天印、周盛、欧阳梅生提供了相关的实例，在此表示感谢。

目前，国内外有关软件工程技术与设计方面的资料很多，新理论、新技术层出不穷。因时间和水平有限，书中有不少不周到和不准确之处，恳请专家学者提出意见和建议，以便进一步完善。

作　者

2014年10月

目　录

第 1 章　软件工程概述

软件工程是一种科学理论来指导软件开发、管理、标准化、自动化的过程。对培养学生的软件素质、提高学生的软件开发和软件项目管理能力具有重要的意义。目前，比较公认的软件工程(Software Engineering)的定义是美国电气与电子工程师协会(IEEE)给出的：将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件，并研究这个过程中的方法。

1.1　软件工程的产生

1.1.1　计算机软件及其特点

世界上第一个写软件的人是阿达(Augusta Ada Lovelace)，在 19 世纪 60 年代尝试为巴贝奇(Charles Babbage)的机械式计算机编写软件。尽管限于当时的制造条件，巴贝奇最终也没有造成理想中的计算机，但他们对后来计算机技术的诞生和发展同样产生了深远的影响，他们的名字永远载入了计算机发展的史册。

在 20 世纪中叶，软件伴随着第一台电子计算机的问世诞生了。以编写软件为职业的人也开始出现，他们多是经过训练的数学家和电子工程师。20 世纪 60 年代美国大学里开始出现授予计算机专业的学位，教学生如何编写软件。软件产业从零开始起步，在短短的 50 多年的时间里迅速发展成为推动人类社会发展的龙头产业，并造就了一批百万、亿万富翁。随着信息产业的发展，软件对人类社会的作用越来越重要。

软件对于人类而言是一个全新的东西，其发展历史不过五六十年。人们对软件的认识经历了一个由浅到深的过程。

随着计算机硬件性能的极大提高和计算机体系结构的不断变化，计算机软件系统更加成熟和更为复杂，从而促使计算机软件的角色发生了巨大的变化，其发展历史大致可以分为如图 1-1 所示的四个阶段。

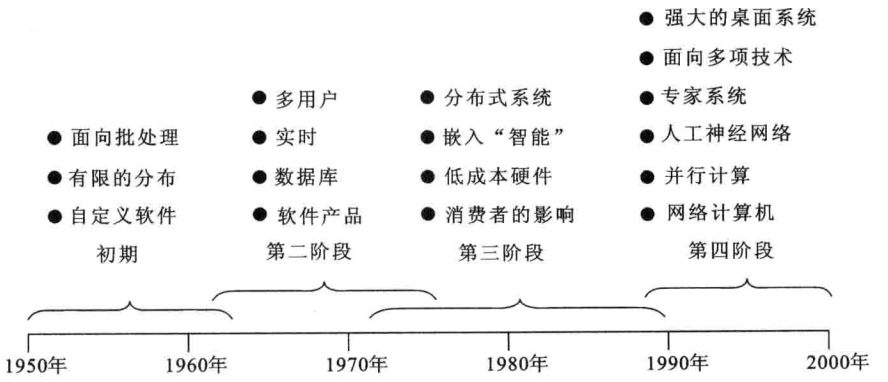

图 1-1　计算机软件发展的四个阶段

第一阶段是20世纪50年代初期至20世纪60年代初期的十余年，是计算机系统开发的初期阶段。当时的软件几乎都是为每个具体应用而专门编写的，编写者和使用者往往是同一个或同一组人。这些个体化的软件设计环境，使软件设计成为在人们头脑中进行的一个隐含过程，最后除了程序清单外，没有其他文档资料保存下来。

实际上，初期开发的计算机系统采用批处理技术，提高了计算机的使用效率，但不利于程序设计、调试和修改。在这个阶段，人们认为计算机的主要用途是快速计算，软件编程简单，不存在什么系统化的方法，开发没有任何管理，程序的质量完全依赖于程序员个人的技巧。

第二阶段跨越了从20世纪60年代中期到20世纪70年代末期的十余年，多用户系统引入了人机交互的新概念，实时系统能够从多个源收集、分析和转换数据，从而使得进程的控制和输出的产生以毫秒而不是分钟来进行，在线存储的发展产生了第一代数据库管理系统。

在这个时期，出现了软件产品和“软件作坊”的概念，设计人员开发软件不再像早期阶段那样只为自己的研究工作需要，而是为了用户更好地使用计算机，但“软件作坊”基本上沿用或仍然沿用早期形成的个体式的软件开发方法。随着计算机应用的日益普及，软件需求量急剧膨胀。在程序运行时发现的错误必须设法更正；用户有了新需求时，必须相应地修改或添加程序；硬件或操作系统更新时，又可能要修改程序以适应新的环境。这样，软件的维护工作以惊人的比例耗费资源，更严重的是，程序设计的个体化和作坊化特性使软件最终成为不可维护的，从而出现了早期的软件危机。人们随之也就开始寻求采用软件工程的方法来解决软件危机问题。

第三阶段是20世纪70年代中期至20世纪80年代末期，分布式系统极大地提高了计算机系统的复杂性，网络的发展对软件开发提出了更高的要求，特别是微处理器的出现和广泛应用，孕育了一系列的智能产品。硬件的发展速度已经超过了人们对软件的需求速度，硬件价格下降、软件的价格急剧上升，导致软件危机的加剧，致使更多的科学家着手研究软件工程学的科学理论、方法和时限等一系列问题。软件开发技术的度量问题受到重视，最著名的有软件工作量估计COCOMO模型、软件过程改进模型CMM等。

第四阶段是从20世纪80年代末期开始的。这个阶段是强大的桌面系统和计算机网络迅速发展的时期，计算机体系结构由中央主机控制方式变为客户机/服务器方式，专家系统和人工智能软件终于走出实验室进入了实际应用，虚拟现实和多媒体系统改变了与最终用户的通信方式，出现了并行计算和网络计算的研究，面向对象技术在许多领域迅速取代了传统软件开发方法。

在软件的发展过程中，软件的需求成为软件发展的动力，软件的开发从自给自足模式发展为在市场中流通以满足广大用户的需要。软件工作的考虑范围也发生了很大变化，人们不再只顾及程序的编写，而是涉及软件的整个生命周期。

软件从个性化的程序变为工程化的产品，人们对软件的看法发生了根本性的变化。现在，软件的正确的定义应该是：软件(software)是计算机系统中与硬件(hardware)相互依存的另一部分，它包括程序(program)、相关数据(data)及其说明文档(document)。

其中程序是按照事先设计的功能和性能要求执行的指令序列；数据是程序能正常操

纵信息的数据结构；文档是与程序开发维护和使用有关的各种图文资料。

软件同传统的工业产品相比，有其独特的特性。

(1) 软件是一种逻辑产品，它与物质产品有很大的区别。软件产品是看不见摸不着的，因而具有无形性，它是脑力劳动的结晶，它以程序和文档的形式出现，保存在计算机存储器和光盘介质上，通过计算机的执行才能体现它的功能和作用。

(2) 软件产品的生产主要是研制，软件产品的成本主要体现在软件的开发和研制上。软件一旦研制开发成功后，通过复制就产生了大量软件产品。

(3) 软件在使用过程中，没有磨损、老化的问题。软件在使用过程中不会因为磨损而老化，但会为了适应硬件、环境以及需求的变化而进行修改，而这些修改又不可避免地引入错误，导致软件失效率升高，从而使得软件退化。当修改的成本变得难以接受时，软件就被抛弃。

(4) 软件对硬件和环境有着不同程度的依赖性，这导致了软件移植的问题。

(5) 软件的开发主要是进行脑力劳动，至今尚未完全摆脱手工作坊式的开发方式，生产效率低，且大部分产品是“定做”的。

(6) 软件是复杂的，而且以后会更加复杂。软件是人类有史以来生产的复杂度最高的工业产品。软件涉及人类社会的各行各业、方方面面，软件开发常常涉及其他领域的专业知识，这对软件工程师提出了很高的要求。

(7) 软件的成本相当昂贵。软件开发需要投入大量的、高强度的脑力劳动，成本非常高，风险也大。现在软件的开销已大大超过了硬件的开销。

(8) 软件工作牵涉到很多社会因素。许多软件的开发和运行涉及机构、体制和管理方式等问题，还会设计到人们的观念和心理。这些人的因素，常常成为软件开发的困难所在，直接影响到项目的成败。

1.1.2 软件危机

在19世纪60年代，很多的软件最后都得到了一个悲惨的结局。很多的软件项目开发时间大大超出了规划的时间。一些项目导致了财产的流失，甚至某些软件导致了人员伤亡。同时软件开发人员也发现软件开发的难度越来越大。

IBM/360被认为是一个典型的案例。IBM/360的开发总投资5亿美元，达到美国研究原子弹的曼哈顿计划投资20亿美元的1/4。在研制期间，布鲁克斯(Frederick Phillips Brooks,Jr.)主持了这个项目，率领着2000名程序员夜以继日地工作，仅OS/360操作系统每年的开发就用了5000人。在当时，IBM/360以其通用化、系列化和标准化的特点，对全世界计算机产业的发展产生了深远的影响，以致被认为是划时代的杰作。迄今为止，它仍然被使用在IBM/360系列主机中。但据统计，这个操作系统每次发行的新版本都是从前一版本中找出上千个程序错误而修正的结果。如今经历了数十年，这个极度复杂的软件项目甚至产生了一套不包括在原始设计方案中的工作系统。后来，布鲁克斯在他的作品《人月神话》(The Mythical Man-Month)中对这个项目的开发研制过程进行分析及总结，承认由于管理方面的原因，项目在某些方面是失败的，甚至犯了一个价值数百万美元的错误。

软件的错误可能导致巨大的财产损失，2002年12月欧洲阿里亚娜火箭的爆炸就是

一个最为惨痛的教训。而且由于计算机软件被广泛应用于包括医院等与生命息息相关的行业，这也使由于软件错误而导致人员伤亡成为了可能。在工业上，某些嵌入式系统导致机器的不正常运转，从而使工作人员陷入险境。

20世纪60年代末至20世纪70年代初，软件危机一词在计算机界广为流传。事实上，软件危机几乎从计算机诞生开始起就出现了，只不过到了1968年，北大西洋公约组织的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“软件危机”(software crisis)这个名词。

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。这类问题绝不仅仅是“不能正常运行的软件”才具有的，实际上几乎所有软件都有不同程度地存在这类问题。概括来说，软件危机包含两方面问题：其一是如何开发软件，以满足不断增长、日趋复杂的需求；其二是如何维护数量不断膨胀的软件产品。

具体地说，软件危机主要有以下表现。

(1) 对软件开发成本和进度的估计常常不准确。开发成本超出预算，实际进度比预定计划一再拖延的现象并不罕见。

(2) 用户对“已完成”系统不满意的现象经常发生。

(3) 软件产品的质量往往靠不住。“缺陷”一大堆，“补丁”一个接一个。

(4) 软件的可维护程度非常之低。

(5) 软件通常没有适当的文档资料。

(6) 软件的成本不断提高。

(7) 软件开发生产率的提高赶不上硬件的发展和人们需求的增长。

之所以出现软件危机，其主要原因一方面是与软件本身的特点有关，另一方面是与软件开发和维护的方法不正确有关。

软件的特点前面已经有一个简单介绍。软件开发和维护的不正确方法主要表现为忽视软件开发前期的需求分析；开发过程没有统一的、规范的方法论的指导，文档资料不齐全，忽视人与人的交流；忽视测试阶段的工作，提交给用户的软件质量差；轻视软件的维护。这些大多数都是软件开发过程中管理上的原因。

1.1.3 软件工程的定义

1968年秋季，NATO(North Atlantic Treaty Organization，北大西洋公约组织)的科技委员会召集了近50名一流的编程人员、计算机科学家和工业界巨头，讨论和制定摆脱“软件危机”的对策。在那次会议上第一次提出了“软件工程”(software engineering)这个概念。当时提出这个概念的Fritz Bauer的主要思路是想将系统工程的原理应用到软件的开发和维护中。

软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法去进行软件的开发和维护的学科。人们曾经给过许多定义，下面给出两个比较典型的定义。

1968年NATO会议上首次提出：“软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件，而建立和使用完善的工程原理。”这个定义不仅指出了软件工程的目标是经济地开发出高质量的软件，而且强调了软件工程是一门工程学科，它应该建立并

使用完善的工程原理。

1993年IEEE进一步给出了一个更全面更具体的定义："软件工程是：①将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中；②对①中所述方法的研究。"

美国南加州大学的巴里·贝姆(Barry Boehm)教授总结了国际上软件工程的发展历程：20世纪50年代的类似硬件工程(Hardware Engineering)、60年代的软件手工生产(Software Crafting)、70年代的形式化方法和瀑布模型(Formality and Waterfall Processes)、80年代的软件生产率和可扩展性(Productivity and Scalability)、90年代的软件并发和顺序进程(Concurrent vs. Sequential Processes)和21世纪初的软件敏捷性和价值(Agility and Value)。我国北京大学杨芙清院士也系统地回顾了起步于1980年的中国软件工程的研究与实践。代表性工作包括软件自动化系统、XYZ系统4、MLIRF系统和青鸟工程等，都在国内外有广泛的影响。

总的来说，软件工程包括两方面内容：软件开发技术和软件项目管理。其中，软件开发技术包括软件开发方法学、软件工具和软件工程环境。软件项目管理包括软件度量、项目估算、进度控制、人员组织、配置管理、项目计划等。

统计数据表明，大多数软件开发项目的失败，并不是由于软件开发技术方面的原因。它们的失败是由于不适当的管理造成的。遗憾的是，尽管人们对软件项目管理重要性的认识有所提高，但在软件管理方面的进步远比在设计方法学和实现方法学上的进步小，至今还提不出管理软件开发的通用指导原则。

在软件的长期发展中，人们针对软件危机的表现和原因，经过不断的实践和总结，越来越认识到：按照工程化的原则和方法组织软件开发工作，是摆脱软件危机的一个主要出路。今天，尽管"软件危机"并未被彻底解决，但软件工程三十多年的发展仍可以说是硕果累累。

1.2　软件工程的基本原理

1.2.1　软件工程的研究对象

软件工程是相当复杂的。涉及的因素很多，不同软件项目使用的开发方法和技术也是不同的，而且有些项目的开发无现成的技术，带有不同程度的试探性。一般说来，软件工程包含四个关键元素：方法(methodologies)、语言(languages)、工具(tools)和过程(procedures)。

软件方法提供如何构造软件的技术。它包括以下内容：与项目有关的计算和各种估算，系统和软件需求分析，数据结构设计，程序体系结构，算法过程，编码，测试和维护等。软件工程的方法通常引入各种专用的图形符号以及一套软件质量的准则。概括地说，软件工程方法规定了以下内容：明确的工作步骤与技术；具体的文档格式；明确的评价标准。

软件语言用于支持软件的分析、设计和实现。随着编译程序和软件技术的完善，传统的编程语言表述能力更强、更加灵活，而且支持过程实现更加抽象的描述。与此同时，规格说明语言和设计语言也开始有更大的可执行子集。而且现在还发展了原型开发语言，

所谓原型开发语言就是除必须具有可执行的能力外，还必须具有规格说明和设计这两种语言的能力。

软件工具是人类在开发软件的活动中智力和体力的扩展和延伸，为方法和语言提供自动或半自动化的支持。软件工具最初是零散的，后来根据不同类型软件项目的要求建立了各种软件工具箱，支持软件开发的全过程。更进一步，人们将用于开发软件的软、硬件工具和软件工程数据库（包括分析、设计、编码和测试等重要信息的数据结构）集成在一起，建立集成化的计算机辅助软件工程（Computer-aided software engineering，CASE）系统。

软件过程贯穿于软件开发的各个环节。它定义了方法使用的顺序、可交付产品（文档、报告以及格式）的要求，为保证质量和协调变化所需要的管理，以及软件开发过程各个阶段完成的标志。

从内容上说，软件工程包括：软件开发理论和结构、软件开发技术，以及软件工程管理和规范。其中，软件开发理论和结构包括程序正确性证明理论、软件可靠性理论、软件成本估算模型、软件开发模型以及模块划分原理；软件开发技术包括软件开发方法学、软件工具以及软件环境；软件工程管理和规范包括软件管理（人员、计划、标准、配置）以及软件经济（成本估算、质量评价）。即软件工程可分为理论、结构、方法、工具、环境、管理、及规范等。理论和结构是软件开发的基础；方法、工具、环境构成软件开发技术，好的工具促进方法的研制，好的方法能改进工具；工具的集合构成软件开发环境；管理是技术实现与开发质量的保证；规范是开发遵循的技术标准。

在软件工程中，软件的可靠性是软件在所给条件下和规定的时间内，能完成所要求功能的性质。软件工程的软件可靠性理论及其评价方法，是贯穿整个软件工程各个阶段所必须考虑的问题。

软件工程的目标在于研究一套科学的工程化方法，并与之相适应，发展一套方便的工具与环境，供软件开发者使用。

1.2.2 软件工程的基本原理

自从 1968 年提出“软件工程”这一术语以来，研究软件工程的专家学者们陆续提出了许多关于软件工程的准则或信条。美国著名的软件工程专家 Boehm 综合这些专家的意见，并总结了 TRW 公司多年的开发软件的经验，于 1983 年提出了软件工程的 7 条基本原理。

Boehm 认为，这 7 条原理是确保软件产品质量和开发效率的原理的最小集合。它们是相互独立的，是缺一不可的最小集合；同时，它们又是相当完备的。下面简要介绍软件工程的 7 条原理。

1. 用分阶段的生命周期计划严格管理

这一条是吸取前人的教训而提出来的。统计表明，50％以上的失败项目是由于计划不周而造成的。在软件开发与维护的漫长生命周期中，需要完成许多性质各异的工作。这条原理意味着，应该把软件生命周期分成若干阶段，并相应制定出切实可行的计划，然后严格按照计划对软件的开发和维护进行管理。Boehm 认为，在整个软件生命周期中应

指定并严格执行6类计划：项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划、运行维护计划。

2. 坚持进行阶段评审

统计结果显示大部分错误是设计错误，大约占63%；错误发现得越晚，改正它要付出的代价就越大，相差大约2到3个数量级。因此，软件的质量保证工作不能等到编码结束之后再进行，应坚持进行严格的阶段评审，以便尽早发现错误。

3. 实行严格的产品控制

开发人员最痛恨的事情之一就是需求改动。但是实践告诉我们，需求的改动往往是不可避免的。这就要求我们要采用科学的产品控制技术来顺应这种要求，也就是要采用变动控制，又叫基准配置管理。当需求变动时，其他各个阶段的文档或代码随之相应变动，以保证软件的一致性。

4. 采纳现代程序设计技术

从20世纪六七十年代的结构化软件开发技术，到最近的面向对象技术，从第一、第二代语言到第四代语言，人们已经充分认识到：方法比气力更有效。采用先进的技术既可以提高软件开发的效率，又可以减少软件维护的成本。

5. 结果应能清楚地审查

软件是一种看不见、摸不着的逻辑产品。软件开发小组的工作进展情况可见性差，难以评价和管理。为更好地进行管理，应根据软件开发的总目标及完成期限，尽量明确地规定开发小组的责任和产品标准，从而使所得到的标准能清楚地审查。

6. 开发小组的人员应少而精

开发人员的素质和数量是影响软件质量和开发效率的重要因素，应该少而精。这一条基于两点原因：高素质开发人员的效率比低素质开发人员的效率要高几倍到几十倍，开发工作中犯的错误也要少得多；当开发小组为 N 人时，可能的通信信道为 $N(N-1)/2$，可见随着人数 N 的增大，通信开销将急剧增大。

7. 承认不断改进软件工程实践的必要性

遵从上述6条基本原理，就能够较好地实现软件的工程化生产。但是，它们只是对现有的经验的总结和归纳，并不能保证赶上技术不断前进发展的步伐。因此，Boehm提出应把承认不断改进软件工程实践的必要性作为软件工程的第7条原理。根据这条原理，不仅要积极采纳新的软件开发技术，还要注意不断总结经验，收集进度和消耗等数据，进行出错类型和问题报告统计。这些数据既可以用来评估新的软件技术的效果，也可以用来指明必须着重注意的问题和应该优先进行研究的工具和技术。

1.3　软件的生存期及开发模型

1.3.1　软件的生存期

从某个待开发软件的目的被提出并着手实现，直到最后停止使用的这个过程，一般称之为软件生存期。软件工程采用的生命周期方法学就是从时间角度对软件开发和维护的