21世纪高等学校应用型教材

RUANJIAN GONGCHENG JISHU

软件工程技术

■ 郑凤仁　任 波　主编

RUANJIAN GONGCHENG JISHU

中国计量出版社
CHINA METROLOGY PUBLISHING HOUSE

21世纪高等学校应用型教材

软件工程技术

郑凤仁　任　波　主编

中国计量出版社

图书在版编目（CIP）数据

软件工程技术/郑凤仁，任波主编．—北京：中国计量出版社，2008.2
21世纪高等学校应用型教材
ISBN 978-7-5026-2786-7

Ⅰ．软… Ⅱ．①郑… ②任… Ⅲ．软件工程—高等学校—教材 Ⅳ．TP311.5

中国版本图书馆CIP数据核字（2008）第001957号

内 容 提 要

本书全面系统地阐述了软件工程技术中所涉及的技术、工具和方法。主要包括软件生命周期与软件开发模型、结构化分析、系统设计、结构化实现、面向对象程序设计、UML建模基础、软件维护、配置管理与文档技术、软件质量管理、软件复用和CASE技术、软件并发计划、软件开发组织等。

本书从实践教学的角度出发，立足提高学生的实践动手能力，立意新颖、内容翔实。采用案例教学，结合实际，容易理解和掌握枯燥的理论和方法。

本书既可作为高等院校、高职高专机电类专业教材，又可作为非机电专业的选修教材，还可作为相关岗位培训教材，供有关技术人员阅读参考。

中国计量出版社 出版
地　址　北京和平里西街甲2号
邮　编　100013
电　话　（010）64275360
网　址　http://www.zgjl.com.cn
发　行　新华书店北京发行所
印　刷　北京市密东印刷有限公司
开　本　787mm×1092mm　1/16
印　张　20.75
字　数　495千字
版　次　2008年9月第1版　2008年9月第1次印刷
印　数　1—3 000
定　价　37.00元

如有印装质量问题，请与本社联系调换

本书编委会

前言

• FOREWORD •

计算机尽管功能强大，但计算机所做的一切都是由人的指令或人们编制的指令集合——程序控制其动作，人们把程序的集合称为软件。在20世纪60年代，只有少数计算机程序设计人员才能从事程序设计。当时人们根据自己的想法和理解随心所欲的编程，结果产生了一堆问题：程序质量低下，错误频出，进度延误，费用剧增……，这些问题导致了“软件危机”。

多年来，随着计算机硬件技术的迅猛发展，人们开发优质软件的能力远远落后于各个领域对计算机软件的需求。时至今日，仍然经受着“软件危机”的困扰。

为了克服“软件危机”，自20世纪60的代末期以来，人们在这一领域做了大量的研究与实践，积累了大量的软件开发技术和方法，提出借鉴传统工业的成功做法，主张通过工程化的方法开发软件来解决软件危机，这种做法称为“软件工程”，逐渐形成了系统的软件项目开发与管理理论。诞生了一门新兴的学科——软件工程。历经30多年的飞速发展，软件工程逐渐成熟，现已成为计算机科学与技术领域中的一门重要学科。

软件工程是指导计算机软件开发与维护的工程学科，基本上是软件实践者的成功经验和失败教训的总结。它采用工程的概念、原理、技术和方法来开发和维护软件，把正确的管理技术和当前最好的技术方法结合起来，以便经济地开发出高质量的软件并有效地维护它。软件工程研究的范围十分广泛，包括软件项目开发和软件维护的有关理论、技术、方法、标准、计算机辅助工具和环境以及软件项目管理等诸多方面。软件工程领域的研究成果为解决软件危机发挥了关键性作用。

随着计算机的日益普及，计算机软件已无处不在。以软件说明、开发、维护和管理为内容，作为信息产业的一个支柱，软件工程这一学科已逐渐为人们所熟悉和广泛应用。现在大家都认识到，如果有哪个项目不遵循软件工程原则，必定会受到实践的惩罚。因此，认真学习并在实际工作中正确地运用软件工程，是摆在我们面前的一项十分迫切的任务。

“软件工程”课程是高等学校计算机学科教学计划中的一门主干课程。本书正是为普通高校计算机学科“软件工程”课程而编写的教材。本书内容新颖，实例丰富，语言文字通俗易懂；各章重点、难点突出，原理、技术和方法的阐述融于丰富的实例之中；便于教学和自学。本书可作为高等院校“软件工程”课程的教材或教学参考书，也可供从事软件开发与应用的工程技术人员和管理人员阅读参考。

本书具有以下几个特点：①结构合理，系统地介绍了软件工程的基本原理、概念、方法和工具。②在选材上注重了实用性，以期达到理论与实践相结合、学以致用的目的。③对计算机软件开发工具的介绍几乎贯穿全书。④概念清楚、通俗易懂、内容翔实、实例丰富，习题、思考题与内容配合紧密。由于时间仓促及编者水平有限，书中难免存在疏漏和不妥之处，恳请广大读者批评指正。

编　者

2008 年 7 月

目 录

• CONTENTS •

第一章 概 述 …… (1)

第一节 软 件 …… (2)
第二节 软件危机 …… (5)
第三节 软件工程 …… (8)
思考题与习题 …… (13)

第二章 软件生命周期与软件开发模型 …… (15)

第一节 软件生命周期的基本任务 …… (15)
第二节 瀑布模型 …… (18)
第三节 快速原型模型 …… (20)
第四节 增量模型 …… (22)
第五节 螺旋模型 …… (24)
第六节 V 型模型 …… (27)
第七节 渐进式阶段模型 …… (28)
第八节 喷泉模型 …… (30)
第九节 案例说明 …… (31)
思考题与习题 …… (34)

第三章 结构化分析 …… (36)

第一节 软件定义过程概述 …… (36)
第二节 结构化分析 …… (38)
第三节 需求规格说明与评审 …… (46)
第四节 需求规格说明书写作范例 …… (48)
思考题与习题 …… (50)

第四章 系统设计 …… (52)

第一节 概要设计的任务与步骤 …… (52)

第二节　软件设计的概要与原则 …… (53)
第三节　面向数据流的设计方法 …… (58)
第四节　概要设计文档评审 …… (64)
第五节　概要设计文档写作范例 …… (64)
第六节　详细设计的任务与原则 …… (66)
第七节　详细设计的方法 …… (66)
第八节　人机界面设计 …… (71)
第九节　详细设计规格说明书与评审 …… (75)
第十节　详细设计文档写作范例 …… (76)
思考题与习题 …… (78)

第五章　结构化实现 …… (79)

第一节　编码 …… (79)
第二节　软件测试基础 …… (83)
第三节　控制结构测试 …… (89)
第四节　黑盒测试技术 …… (96)
第五节　测试策略 …… (101)
第六节　调试 …… (108)
第七节　软件可靠性 …… (110)
思考题与习题 …… (114)

第六章　面向对象程序设计 …… (119)

第一节　面向对象方法学概述 …… (119)
第二节　面向对象方法学的主要优点 …… (121)
第三节　面向对象的概念 …… (124)
第四节　面向对象建模 …… (130)
第五节　对象模型 …… (131)
第六节　动态模型 …… (136)
第七节　功能模型 …… (137)
第八节　面向对象程序设计规则 …… (138)
第九节　设计问题域子系统 …… (141)
第十节　设计人—机交互子系统 …… (143)
第十一节　设计任务管理子系统 …… (145)
第十二节　设计数据管理子系统 …… (147)
第十三节　设计类中的服务 …… (150)
第十四节　面向对象实现技术 …… (151)
第十五节　面向对象测试 …… (159)

思考题与习题 …… (165)

第七章 UML 建模基础 …… (166)

第一节 UML 概述 …… (166)
第二节 用例和用例图 …… (171)
第三节 类图和对象图 …… (181)
第四节 顺序图和协作图 …… (194)
第五节 状态图和活动图 …… (211)
第六节 构件图和部署图 …… (230)
第七节 包图 …… (240)
第八节 UML 的结构 …… (242)
第九节 扩展 UML …… (243)
思考题与习题 …… (244)

第八章 软件维护 …… (246)

第一节 软件维护的内容 …… (246)
第二节 软件维护的特点 …… (247)
第三节 软件维护的实施 …… (249)
第四节 维护的副作用 …… (252)
第五节 软件可维护性 …… (253)
第六节 软件再工程 …… (255)
思考题与习题 …… (257)

第九章 配置管理与文档技术 …… (258)

第一节 软件配置管理基本概念 …… (258)
第二节 软件配置管理的任务 …… (261)
第三节 软件文档技术 …… (267)
思考题与习题 …… (273)

第十章 软件质量管理 …… (275)

第一节 软件质量的定义 …… (275)
第二节 软件的质量保证 …… (276)
第三节 软件质量保证标准 …… (278)
第四节 软件过程成熟度模型 (CMM) …… (281)
思考题与习题 …… (287)

第十一章 软件复用和 CASE 技术 …… (288)

第一节 软件复用的概念 …… (288)

第二节　面向对象与软件复用 …… (290)
第三节　计算机辅助软件工程（CASE） …… (291)
思考题与习题 …… (294)

第十二章　软件开发计划 …… (295)

第一节　度量软件规模 …… (295)
第二节　工作量估算 …… (298)
第三节　进度计划 …… (302)
思考题与习题 …… (310)

第十三章　软件开发组织 …… (311)

第一节　民主制程序员组 …… (311)
第二节　主程序员组 …… (312)
第三节　现代程序员组 …… (314)
第四节　软件项目组 …… (315)
思考题与习题 …… (318)

参考文献 …… (319)

第一章 概 述

人类社会已经步入了21世纪，计算机系统的发展也经历了4个不同的阶段，但是，我们仍然没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的关键因素。

为了更有效地开发与维护软件，软件工作者在20世纪60年代后期开始认真研究消除软件危机的方法，从而逐渐形成了计算机科学技术领域中的一门新兴的工程学科——软件工程。

所谓计算机系统就是指适当地组织在一起的一系列系统元素的集合，这些系统元素相互配合、相互协作，通过对信息的处理而完成预先定义的目标。计算机系统中通常包含的系统元素有：计算机软件、计算机硬件、人员、数据库、文档和过程。其中，软件是程序、数据结构和相关文档的集合，用于实现所需要的逻辑方法、过程和控制；硬件是提供计算能力的电子设备和提供外部世界功能的电子机械设备（例如传感器、马达、水泵等）；人员是硬件和软件的用户和操作者；数据库是通过软件访问的大型的、有组织的信息集合；文档是描述系统使用方法的手册、表格、图形及其他描述性信息；过程是一系列步骤，它们定义了每个系统元素的特定使用方法或系统驻留的过程性语境。

迄今为止，计算机系统已经经历了4个不同的发展阶段。下面简要地介绍各个发展阶段的特点，以便读者了解产生软件危机的背景。

20世纪60年代初期以前，是计算机系统发展的早期时代。在这个时期硬件已经相当普遍，软件却是为每个具体应用而专门编写的，大多数人认为软件开发是无需预先计划的事情。这时的软件实际上就是规模较小的程序，程序的编写者和使用者往往是同一个（或同一组）人。由于规模小，程序编写起来相当容易，也没什么系统化的方法，对软件开发工作更没有进行任何管理。这种个体化的软件环境，使得软件设计往往只是在人们头脑中隐含进行的一个模糊过程，除了程序清单之外，根本没有其他文档资料保存下来。

从60年代中期到70年代中期，是计算机系统发展的第二代。在这10年中计算机技术有了很大进步。多道程序、多用户系统引入了人机交互的新概念，开创了计算机应用的新境界，使硬件和软件的配合上了一个新的层次。实时系统能够从多个信息源收集、分析和转换数据，从而使得进程控制能以毫秒而不是分钟来进行。在线存储技术的进步导致了第一代数据库管理系统的出现。

计算机发展的第二代的一个重要特征是出现了“软件作坊”，广泛使用产品软件。但是，“软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。随着计算机应用的日益普及，软件数量急剧膨胀。在程序运行时发现的错误必须设法改正；用户有了新的需求时必须相应地修改程序；硬件或操作系统更新时，通常需要修改程序以适应新的环境。上述种种软件维护工作，以令人吃惊的比例耗费资源。更严重的是，许多程序的个体化特性使得它们最终成为不可维护的。“软件危机”就这样开始出现了。1968年北大西洋公约组织的计算机科学家在联邦德国召开国际会议，讨论软件危机问题，在这次会议上正式提出并使用了“软件工

程”这个名词，一门新兴的工程学科就此诞生了。

计算机系统发展的第三代从20世纪70年代中期开始，并且跨越了整整10年。在这10年中计算机技术又有了很大进步。分布式系统极大地增加了计算机系统的复杂性，局域网、广域网、宽带数字通信以及对“即时”数据访问需求的增加，都对软件开发者提出了更高的要求。但是，在这个时期软件仍然主要在工业界和学术界应用，个人应用还很少。

这个时期主要的特点是出现了微处理器，而且微处理器获得了广泛的应用。以微处理器为核心的“智能”产品随处可见，当然，最重要的智能产品是个人计算机。在不到10年的时间里，个人计算机已经成为大众化的商品。

在计算机系统发展的第四代已经不再看重单台计算机和程序，人们感受到的是硬件和软件的综合效果。由复杂操作系统控制的强大的桌面机及局域网和广域网，与先进的应用软件相配合，已经成为当前主流。计算机体系结构已迅速地从集中的主机环境转变成分布的客户机/服务器（或浏览器/服务器）环境。世界范围的信息网为人们进行广泛交流和资源的充分共享提供了条件。

软件产业在世界经济中占有举足轻重的地位。随着时代的前进，新的技术也不断地涌现出来。面向对象技术（本书第三章将系统地讲述）已经在许多领域迅速地取代了传统的软件开发方法。

软件开发的“第四代技术”改变了软件界开发计算机程序的方式。专家系统和人工智能软件终于从实验室走出来进入了实际应用，解决了大量实际问题。应用模糊逻辑的人工神经网络软件，展现了模式识别与拟人信息处理的美好前景。虚拟现实技术与多媒体系统，使得与用户的通信可以采用和以前完全不同的方法。遗传算法使我们有可能开发出驻留在大型并行生物计算机上的软件。

第一节　软　件

一、软件的定义

软件是计算机系统中与硬件相互依存的部分，是包括程序、数据及相关文档的完整集合。其中，程序是按实现设计的功能和性能要求执行的指令序列；数据是程序所处理信息的数据结构；文档是与程序开发、维护和使用相关的各种图文资料。

二、软件的特点及最新发展

为全面、正确地理解计算机系统及其软件，我们必须了解软件的以下特点。

1. 抽象性

软件是一种逻辑实体，而不是具体的物理实体。这种抽象性是软件与硬件的根本区别。软件一般寄存在诸如纸、内存储器、磁带、磁盘或光盘等载体上，我们无法观察到它的具体形态，而必须通过对它的运行来分析了解它的功能和特征。

2. 无明显的制造过程

软件的生产与其他硬件的生产不同，它没有明显的制造过程。在硬件的制造过程中必须对每一个制造环节都进行质量控制，以保证整个硬件产品的质量，并且每一个硬件都几乎付

出与样品一样的生产资料成本。而软件是将人类的知识和技术转化成产品，软件产品的开发成本几乎全部用在样品的开发设计上，其制造过程则非常简单，人们可以用很低的成本进行软件产品的复制，因此也产生了软件产品的保护问题。软件产品保护这个问题已引起国际上的普遍重视，为了保护软件开发者的根本利益，除国家在法律上采取有力的措施之外，开发者在技术上也采取了各种措施，防止对软件产品的随意复制。

3. 无磨损、老化的问题

在软件的运行和使用期间，没有像硬件那样的磨损、老化问题。任何机械、电子设备在运行和使用的过程中，其失效率大致遵循如图 1—1（硬件失效率曲线）中所示的 U 型曲线（即浴盆曲线）。软件的情况则与此不同，它不存在磨损和老化的问题，然而它却存在退化的问题，设计人员必须不断地修改软件，如图 1—2（软件失效率曲线）所示。

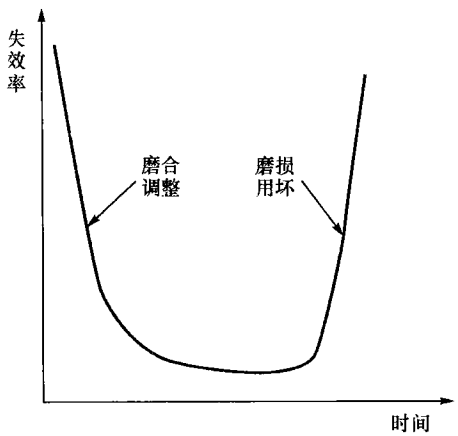

图 1—1　硬件失效率曲线

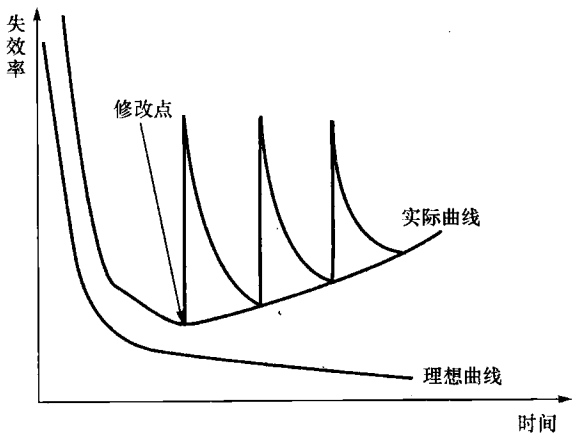

图 1—2　软件失效率曲线

4. 对硬件系统的依赖性

软件的开发和运行往往受到计算机系统的限制，对计算机系统有着不同程度的依赖性。为了减少这种依赖性，在软件开发中提出了软件的可移植问题。

5. 复杂性

软件本身是复杂的。软件的复杂性可能来自它所反映的实际问题的复杂性，也可能来自程序逻辑结构的复杂性。

6. 成本昂贵

软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它投入的成本是较高的。

7. 社会性

相当多的软件工作涉及各种社会因素，许多软件的开发和运行涉及机构设置、体制运作及管理方式等问题，甚至涉及人们的观念和心理，这些因素直接影响软件项目的成败。

自 20 世纪 40 年代世界上第一台计算机问世以来，软件经历了程序设计、程序系统及软

件工程3个阶段的发展。①程序设计阶段（20世纪50年代至60年代）的软件指的是程序，程序的开发采用个体工作的方式，开发工作主要依赖于开发人员的个人技能和程序设计技巧，软件的质量得不到保证，缺乏与程序有关的文档。②程序系统阶段（20世纪60年代至70年代）的软件是指程序和说明书，软件的开发采用开发小组工作的方式，开发工作主要依赖于开发小组的水平，文档资料的不齐全给软件维护带来了很大的难度，软件技术的发展不能满足需要。③软件工程阶段（20世纪70年代以后）的软件则是指程序、文档、数据，由开发小组及大中型软件开发机构承担开发软件的任务，开发工作主要依赖于整个机构的管理水平，采用多种开发技术。

目前在很多的应用领域，人们开始采用面向对象的软件开发技术。专家系统、人工智能软件开始走向实际应用。软件技术呈现国际化、网络化、服务化等多种发展趋势。互联网作为21世纪最重要的科技成果之一，给人类生活和经济发展带来了深远的影响，它所展现出的勃勃商机，吸引了众多厂商围绕互联网开发软件，与分布计算、网络和互联网相关的软件技术成为软件领域的主要技术热点。此外，自由软件潮流、智能化、简易化、多样化等趋势正极大地拓展软件产业的发展空间，派生出许多具有成长潜力的新兴领域。

三、软件的分类

20世纪40年代以来，尽管人们开发了大量的软件，积累了丰富的软件资源并使之广泛应用于各个领域，但软件的品种、质量和价格方面仍然满足不了人们日益增长的需要。随着软件复杂性和交互性的增加，我们难以对目前应用着的软件进行一个标准化的分类，这里只是简单地介绍计算机软件在计算机系统、实时系统、商业管理、科学和工程计算、嵌入式系统、人工智能等方面的应用。

1. 系统软件

系统软件是指能与计算机硬件系统紧密配合，使计算机系统的各个部件、相关软件和数据协调高效地工作的软件。系统软件是计算机系统的重要组成部分，它支持应用软件的开发和运行。系统软件包括：操作系统、网络软件、编译程序、数据库管理程序、文件编辑系统、系统检查与诊断软件等。

2. 实时软件

监视、分析和控制现实世界中发生的事件，能以足够快的速度对输入信息进行处理并在规定的时间内作出反应的软件，称之为实时软件。实时软件包括4个组成部分：数据采集器（负责从外部环境中获取和格式化信息）、分析器（负责将信息转换成应用所需要的形式）、输出/控件器（负责响应外部环境）、管理器（负责协调系统各个部件工作，使系统能保持在一个可接受的响应时间内给予实时响应）。实时系统必须在严格的时间范围内响应，因此实时软件和计算机系统必须有很高的可靠性和安全性。

3. 商业软件

商业软件可以访问一个或多个商业信息的大型数据库，将已有的数据重新构造，变换成一种能够辅助商业操作和管理决策的形式。商业信息处理是当今最大的软件应用领域，典型的商业软件有银行储蓄软件、电子商务软件、仓库管理软件、ERP软件等。

4. 科学和工程计算软件

科学和工程计算软件的特征是利用数值分析算法，主要用于科学和工程的计算。如天气

预报、弹道计算、石油勘探、地震数据处理、计算机系统仿真、计算机辅助设计等。

5. 嵌入式软件

嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分，控制该系统的运行，进而实现一个特定的物理过程。用于嵌入式计算机系统的软件称为嵌入式软件。大型的嵌入式计算机系统软件可用于航空航天系统、指挥控制系统、武器系统等；小型的嵌入式计算机系统软件可用于工业的智能化产品之中，这时嵌入式软件驻留在只读存储器内，为该产品提供各种控制功能和仪表的数字或图形显示功能等。如汽车的刹车控制，空调、洗衣机的自动控制等。

6. 人工智能软件

人工智能软件利用非数值算法去解决复杂的问题，一般说来，这些问题都不能通过计算或直接分析而得到答案。迄今为止，在专家系统、模式识别、自然语言理解、人工神经网络、程序验证、自动程序设计、机器人学等领域开发了许多人工智能应用软件，用于诊断疾病、产品检测、自动定理证明、图像和语音的自动识别、语言翻译等。

第二节　软件危机

一、软件危机的含义

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。这些问题绝不仅仅是不能正常运行的软件才具有的，实际上，几乎所有软件都不同程度地存在这些问题。概括地说，软件危机包含下述两个问题：如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件。鉴于软件危机的长期性和症状不明显的特征，近年来有人建议把软件危机更名为“软件萧条（depression）”或“软件困扰（affliction）”。不过“软件危机”这个词强调了问题的严重性，而且也已为绝大多数软件工作者所熟悉，所以本书仍将沿用它。

具体来说，软件危机主要有以下一些典型表现。

①对软件开发成本和进度的估计常常很不准确。实际成本比估计成本有可能高出一个数量级，实际进度比预期进度拖延几个月甚至几年的现象并不罕见。这种现象降低了软件开发组织的信誉。而为了赶进度和节约成本所采取的一些权宜之计又往往损害了软件产品的质量，从而不可避免地会引起用户的不满。

②用户对“已完成的”软件系统不满意的现象经常发生。软件开发人员常常在对用户要求只有模糊的了解，甚至对所要解决的问题还没有确切认识的情况下，就仓促上阵匆忙着手编写程序。软件开发人员和用户之间的信息交流往往很不充分，“闭门造车”必然导致最终的产品不符合用户的实际需要。

③软件的产品质量往往靠不住。软件可靠性和质量保证的确切定量概念刚刚出现不久，质量保证技术还没有坚持不懈地应用到软件开发的全过程中，这些都导致软件产品发生质量问题。

④软件常常是不可维护的。很多程序中的错误是非常难改正的，实际上不可能使这些程序适应新的环境，也不能根据用户的需要在原有程序中增加一些新的功能。“可重用的软件”

还是一个没有完全做到的、正在努力追求的目标，人们仍然在重复开发类似的或基本类似的软件。

⑤软件通常没有适当的文档资料。计算机软件不仅仅是程序，还应该有一整套文档资料。这些文档资料应该是在软件开发过程中产生出来的，而且应该是“最新式的”（即和程序代码完全一致的）。软件开发组织的管理人员可以使用这些文档资料作为“里程碑”，来管理和评价软件开发工程的进展状况；软件开发人员可以利用它们作为通信工具，在软件开发过程中准确地交流信息；对于软件维护人员而言，这些文档资料更是至关重要、必不可少的。缺乏必要的文档资料或者文档资料不合格，必然给软件开发和维护带来许多严重的困难和问题。

⑥软件成本在计算机系统总成本中所占的比例逐年上升。由于微电子学技术的进步和生产自动化程度不断提高，硬件成本逐年下降。然而软件开发需要大量人力，软件成本随着通货膨胀以及软件规模和数量的不断扩大而持续上升。美国在1985年软件成本大约已占计算机系统总成本的90%。

⑦软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。软件产品“供不应求”的现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些明显的表现，与软件开发和维护有关的问题远远不止这些。

二、产生软件危机的原因

在软件开发和维护的过程中存在这么多的严重问题，一方面与软件本身的特点有关，另一方面也和软件开发与维护的方法不正确有关。

软件不同于硬件，它是计算机系统中的逻辑部件而不是物理部件。由于软件缺乏“可见性”，在写出程序代码并在计算机上试运行之前，软件开发过程的进展情况较难衡量，软件的质量也较难评价，因此，管理和控制软件开发过程相当困难。此外，软件在运行过程中不会因为使用时间过长而被“用坏”，如果运行中发现错误，很可能是遇到了一个在开发时期引入的在测试阶段没能检测出来的错误。因此，软件维护通常意味着改正或修改原来的设计，这就在客观上使得软件较难维护。

软件不同于一般程序，它的一个显著特点是规模庞大，而且程序复杂性将随着程序规模的增加而呈指数上升。为了在预定时间内开发出规模庞大的软件，必须由许多人分工合作，然而，如何保证每个人完成的工作合在一起确实能构成一个高质量的大型软件系统，更是一个极端复杂困难的问题，不仅涉及许多技术问题，诸如分析方法、设计方法、形式说明方法、版本控制等，更重要的是必须有严格而科学的管理。

软件本身独有的特点确实给开发和维护带来一些客观困难，但是人们在开发和使用计算机系统的长期实践中，也确实积累和总结出了许多成功的经验。如果坚持不懈地使用经过实践考验证明是正确的方法，许多困难是完全可以克服的，过去也确实有一些成功的范例。但是，目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念，在实践过程中或多或少地采用了错误的方法和技术，这可能是使软件问题发展成软件危机的主要原因。

与软件开发和维护有关的许多错误认识和做法的形成，可以归因于在计算机系统发展的

早期阶段软件开发的个体化特点。错误的认识和作法主要表现为忽视软件需求分析的重要性，认为软件开发就是写程序并设法使之运行，轻视软件维护等。

事实上，对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。只有用户才真正了解他们自己的需要，但是许多用户在开始时并不能准确具体地叙述他们的需要，软件开发人员需要做大量深入细致的调查研究工作，反复多次地和用户交流信息，才能真正全面、准确、具体地了解用户的要求。对问题和目标的正确认识是解决任何问题的前提和出发点，软件开发同样也不例外。急于求成，仓促上阵，对用户要求没有正确认识就匆忙着手编写程序，这就如同不打好地基就盖高楼一样，最终必然垮台。事实上，越早开始写程序，完成它所需要用的时间往往越长。

一个软件从定义、开发、使用和维护，直到最终被废弃，要经历一个漫长的时期，这就如同一个人要经过胎儿、儿童、青年、中年和老年，直到最终死亡的漫长时期。通常把软件经历的这个漫长的时期称为生命周期。软件开发最终的工作应是问题定义，也就是确定要求解决的问题是什么；然后要进行可行性研究，决定该问题是否存在一个可行的解决办法；接下来应该进行需求分析，也就是深入具体地了解用户的要求，在所要开发的系统（不妨称之为目标系统）必须做什么这个问题上和用户取得完全一致的看法。经过上述软件定义时期的准备工作才能进入个人开发时期，而在开发时期首先需要对软件进行设计（通常又分为概要设计和详细设计两个阶段），然后才能进入编写程序的阶段，程序编写完之后还必须经过大量的测试工作（需要的工作量通常占软件开发全部工作量的 40%～50%）才能最终交付使用。所以，编写程序只是软件开发过程中的一个阶段，而且在典型的软件开发工程中，编写程序所需的工作量只占软件开发全部工作量的 10%～20%。

另一方面还必须认识到程序只是完整的软件产品的一个组成部分，在上述软件生命周期的每个阶段都要得出最终产品的一个或几个组成部分（这些组成部分通常以文档资料的形式存在）。也就是说，一个软件产品必须由一个完整的配置组成，软件配置主要包括程序、文档和数据等成分。必须清除只重视程序而忽略软件配置其余成分的糊涂观念。

做好软件定义时期的工作，是降低软件成本提高软件质量的关键。如果软件开发人员在定义时期没有正确全面地理解用户需求，直到测试阶段或软件交付使用后才发现“已完成的”软件不完全符合用户的需要，这时再修改就为时已晚了。

严重的问题是，在软件开发的不同阶段进行修改需要付出的代价是很不相同的，在早期引入变动，涉及的面较少，因而代价也比较低；而在开发的中期软件配置的许多成分已经完成，引入一个变动要对所有已完成的配置成分都做相应的修改，不仅工作量大，而且逻辑上也更复杂，因此付出的代价剧增；在软件“已经完成”时再引入变动，当然需要付出更高的代价。根据美国一些软件公司的统计资料，在后期引入一个变动比在早期引入相同变动所需付出的代价高 2～3 个数量级。图 1—3 定性地描绘了在不同时期引入一个变动需要付出

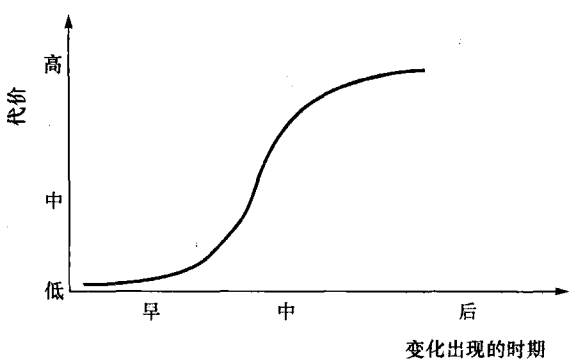

图 1—3　引入同一变动付出的代价随时间变化的趋势