普通高等教育“十二五”规划教材

软件工程

理论及应用

RUANJIAN GONGCHENG LILUN JI YINGYONG

周屹 王丁◎主编

机械工业出版社
CHINA MACHINE PRESS

www.cmpedu.com

配电子课件

普通高等教育“十二五”规划教材

软件工程理论及应用

主　编　周　屹　王　丁
副主编　朱海龙　于雪梅
参　编　张　辉　徐小舟
主　审　贾宗福

机械工业出版社

软件工程是高等院校计算机相关学科各专业的专业基础课，其研究范围非常广泛。本书从实用的角度出发，在系统讲解软件工程理论和方法的同时，注重结合实例，分析软件工程技术与工具的综合应用；在强调传统的结构化方法的同时，着重介绍面向对象方法。

全书共分 10 章，包括软件产品、软件过程、项目管理和软件项目计划、项目进度安排及跟踪、软件工程的需求工程、软件设计、面向对象的分析方法、面向对象设计、面向对象测试和软件维护工程。

本书将理论知识与实践案例相结合，便于教学与应用，文字通俗易懂，概念清晰，实例丰富，实用性强，并配有习题。本书可作为高等院校计算机类专业软件工程相关课程的教材，也可作为软件开发人员的参考书。

为方便教学，本书配备电子课件等教学资源。凡选用本书作为教材的教师均可登录机械工业出版社教材服务网 www.cmpedu.com 免费下载。如有问题请致信 cmpgaozhi@sina.com，或致电 010-88379375 联系营销人员。

图书在版编目（CIP）数据

软件工程理论及应用/周峰，王[illegible]主编. —北京：机械工业出版社，2014.6
普通高等教育“十二五”规划教材
ISBN 978-7-111-46404-4

Ⅰ.①软… Ⅱ.①周… ②王… 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 069091 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：刘子峰 责任编辑：刘子峰 吴晋瑜
责任校对：申春香 封面设计：路恩中
责任印制：李 洋
中国农业出版社印刷厂印刷
2014 年 6 月第 1 版第 1 次印刷
184mm×260mm · 16.25 印张 · 425 千字
0001—2500 册
标准书号：ISBN 978-7-111-46404-4
定价：33.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

前　言

计算机技术的快速发展为人类社会带来了深刻的变革。如今，信息化的建设、发展和技术应用水平已成为衡量国家综合国力的重要指标，而软件工程又是信息化的核心与信息系统的关键所在。因此，随着市场需求与标准的不断提高，从事计算机软件开发、维护及管理的高层次专业人才的综合素质也越来越重要。

近年来，软件工程学科发生了巨大变化，从传统的结构化技术占主导地位，发展到面向对象技术占主导地位，继而发展到基于构件的技术成为开发技术的主流。随着 Internet 的普及与发展，软件工程出现了平台网络化、方法对象化、系统构件化、产品家族化、开发工程化、过程规范化、生产规模化、竞争国际化的态势，也使得软件在反映对象、提交形式、关注内容和运行方式等方面有了重大进展。

软件工程是高等院校计算机相关学科各专业的专业基础课程，它的研究范围非常广泛，包括的系列任务有立项、可行性分析、需求分析、概要设计、详细设计、编程、测试和修改维护。通过本课程的学习，学生需要掌握软件工程的基本概念、基本原理、实用的开发方法和技术；了解软件工程各领域的发展方向；掌握如何用工程化的思想和方法开发和管理软件项目，以及掌握开发过程中应遵循的流程、准则、标准和规范。

本书从实用的角度出发，注重结合实例，按软件生存周期的顺序介绍问题定义、可行性研究、需求分析、总体设计、详细设计、编码、测试与软件维护等各个阶段的任务、过程、方法和工具，目标是使学生能够针对具体软件工程项目，全面掌握软件工程管理、软件需求分析、软件初步设计、软件详细设计、软件测试等阶段的方法和技术。全书共分 10 章，内容包括软件产品、软件过程、项目管理和软件项目计划、项目进度安排及跟踪、软件工程的需求工程、软件设计、面向对象的分析方法、面向对象设计、面向对象测试和软件维护工程。

本书的编写队伍由具有丰富教学经验的高校软件专业一线教师和多年从事软件项目开发的工程师组成。周屹、王丁任主编，朱海龙、于雪梅任副主编，参加编写的有张辉、徐小舟。编写分工如下：朱海龙编写第 1、2 章，王丁编写第 3、4、6 章，于雪梅编写第 7、9 章，张辉编写第 5、10 章，周屹、徐小舟编写第 8 章，并参加了审校及修改等工作。全书由周屹最终统稿，贾宗福任主审。

本书在编写过程中，得到了各方面有关专家的大力支持和帮助，在此对所有人的支持表示衷心的感谢。

由于编者水平有限，书中难免存在不足之处，敬请广大读者批评指正。

编　者

目　录

第1章 软件产品

目前，软件担任着双重角色。它是一种产品，同时又是开发和运行产品的载体。作为一种产品，它表达了由计算机硬件体现的计算潜能。不管是应用在移动电话中，还是操作在个人计算机上，软件就是一个信息转换器——产生、管理、获取、修改、显示或转换信息，这些信息可以很简单，如一个单个的位（bit）；也可以很复杂，如多媒体仿真信息。作为开发运行产品的载体，软件是计算机控制（操作系统）的基础、信息通信（网络）的基础，也是创建和控制其他程序（软件工具和环境）的基础。

“软件”这一名词在20世纪60年代初从国外引入，当时人们无法说清它的具体含义，也无法解释它的英文单词“software”，于是有人把它翻译成“软件”或“软制品”，现在统一称其为软件。早期，人们认为软件就是源程序。随着对软件及其特性的更深层的研究，人们认为软件不仅仅包括程序，还应包含其他相关内容。

目前，对软件通俗的解释为：软件=程序+数据+文档资料。其定义为计算机程序及其相关说明程序的各种文档。在该定义中，“程序”是计算任务的处理对象和处理规则的描述；数据是程序运行的基础和操作的对象；“文档”是有关计算机程序功能、设计、编制、使用的文字或图形资料。

软件与硬件一起构成完整的计算机系统，它们是相互依存、缺一不可的。软件是一种特殊的产品。

计算机软件的角色在20世纪后半叶发生了很大的变化。硬件性能的极大提高、计算机体系结构的不断变化、内存和硬盘容量的快速增加以及输入/输出设备的大量涌现，均促进了更为成熟和更为复杂的基于计算机的软件系统的出现。如果一个系统是成功的，那么这种成熟性和复杂性能够产生出奇迹般的结果，但它们同时也给建造这些复杂系统的人员带来了很多的问题。

计算机在使社会生产力得到迅速解放、社会高度自动化和信息化的同时，却没有使计算机本身的软件生产得到类似的巨大进步。软件开发面临着过分依赖人工、软件无法重用、开发大量重复和生产率低下等问题，特别是软件危机的出现，促使人们努力探索软件开发的新思想、新方法和新技术。

伴随计算机系统的发展，分布式系统（即多台计算机，每一台都在同时执行某些功能，并与其他计算机通信）极大地提高了计算机系统的复杂性。广域网和局域网、高带宽数字通信以及对“即时”数据访问需求的增加都向软件开发者提出了更高的要求。然而，软件仍然继续应用于工业界和学术界，个人应用很少。

微处理器孕育了一系列的智能产品，从汽车到微波炉，从工业机器人到血液检测设备，但哪一个也没有个人计算机那么重要。在不到10年的时间里，计算机真正成为大众化的产品。软件是信息化社会和知识经济的基础，它渗透到人们生活、工作的各个领域，并迅速地改变着人们的生活和工作方式，改变着社会的产业结构和面貌。人们对软件的依靠越来越多，社会需要大量功能各异的软件，并且应是随着社会的发展不断更新、升级的软件。

所有国家都在使用复杂的计算机系统。越来越多的产品把计算机和控制软件以一定的方式结合起来。软件工程是计算学科的9个领域之一。这9个领域包括算法和数据结构、计算机系统结构、人工智能和机器人学、数据库和信息检索、人机交互、操作系统、程序设计语言、软件方法学和软件工程以及数字和符号计算。

1.1 软件的发展

在计算机发展的早期阶段，大多数人把软件设计看成是不须预先计划的事情。当时，计算机编程很简单，没有什么系统化的方法。软件的开发没有任何管理，一旦计划提前了或成本提高了，程序员才开始手忙脚乱地弥补。

在通用的硬件已被普遍应用时，软件却相反，对每一类应用均须再自行设计，且应用范围很有限。软件产品还在“婴儿”阶段，大多数软件均是由使用它们的人员或组织自己开发的，比如编写软件，使其运行，如果有问题再负责改好。工作的可变性很低，管理者必须得到保证：一旦发生了错误就必须有人在那里处理。因为这种个人化的软件环境，设计往往仅是人们头脑中的一种模糊想法，而文档根本不存在。

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且更为复杂，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。

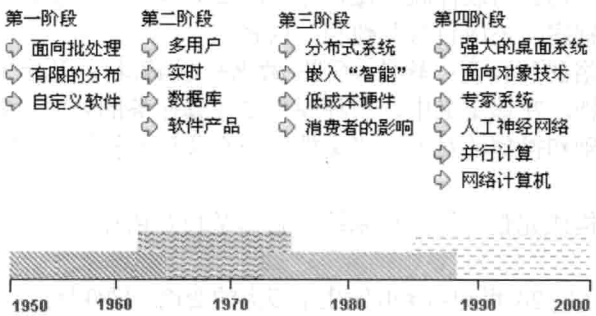

图 1-1 软件的发展阶段

第一阶段：在计算机发展的早期阶段，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，开发也没有任何管理，程序的质量完全依赖于程序员个人的技巧。

第一批软件公司是为客户开发定制解决方案的专业软件服务公司。在美国，这个发展过程是由几个巨型软件项目推进的，这些项目先是由美国政府出面筹划，后来是由几家美国大公司认购。这些巨型项目为第一批独立的美国软件公司提供了重要的学习机会，并使美国在软件业中成了早期的主角。例如开发于 1949 年～1962 年的 SAGE 系统，是第一个极大的计算机项目。在欧洲，几家软件承包商也在 20 世纪 50 年代和 60 年代开始发展起来，但总体上，比美国晚了几年。

第二阶段：计算机软件发展的第二阶段跨越了从 20 世纪 60 年代中期到 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数据源收集、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，在线存储的发展催生了第一代数据库管理系统。

在这个时期，软件产品和“软件作坊”的概念出现了，设计人员开发程序不再像早期阶段那样只为自己的研究工作需要，而是为了让用户更好地使用计算机。人们开始采用“软件工程”的方法来解决“软件危机”问题。

在第一批独立软件服务公司成立 10 年后，第一批软件产品出现了。它们被专门开发出来重复地销售给一个以上的客户。一种新型的软件公司诞生了，这是一种要求不同管理技术

的公司。第一个真正的软件产品诞生于 1964 年，是由 ADR 公司接受 RCA 委托开发的一个可以在一个程序里形象地代表设备的逻辑流程图的程序。

在这个时期，软件开发者设立了今天仍然存在的基础，包括一个软件产品的基本概念、它的定价、它的维护以及它的法律保护手段。此外，它们还证实了软件项目和软件产品企业是两个不同的行业。

第三阶段：计算机软件发展的第三阶段始于 20 世纪 70 年代中期，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。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算模型 COCOMO、软件能力成熟度模型 CMM 等。

在第二阶段的后期，众多独立软件公司一涌而现，为所有不同规模的企业提供新产品，由此可以看出他们超越了硬件厂商所提供的产品。最终，客户开始从硬件公司以外的卖主那儿寻找他们的软件来源并为其付钱。20 世纪 70 年代早期的数据库市场是最活跃的，原因之一是独立数据库公司的出现。数据库系统在技术上很复杂，而且几乎所有行业都需要它。但从由计算机生产商提供的系统被认为不够完善以来，独立的提供商“侵入”了这个市场，使其成为 70 年代最活跃的市场之一。

欧洲同样进入了这个市场。1969 年，在德国法兰克福南部的一个中等城市——达姆斯塔特的应用信息处理研究所的 6 位成员，创立了 Software AG，至 1972 年它进入了美国市场，而且此后不久，就在全世界销售它的主打产品。其他在这个市场扮演重要角色的公司有 CINCOM 系统公司（1968 年）、计算机联合（CA）公司（1976 年）和 Sybase（1984 年）。

在 20 世纪 80 年代和 90 年代，许多企业解决方案提供商从大型计算机专有的操作系统平台转向诸如 UNIX（1973 年）、IBM OS/2 和微软 NT 等新的平台。这个转变通常使这些公司从使用他们自己所开发的软件中获得了暴利。

第四阶段：计算机软件发展的第四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，从“软件=程序”发展为“软件=程序+数据+文档”。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需要的模式。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。

个人计算机的出现建立了一种全新的软件：基于个人计算机的大众市场套装软件。同样，这种市场的出现也影响了以前的营销方式。1975 年，第一批“个人”计算机诞生于美国 MITS 的 Altair 8800，同样还有苹果 II 型计算机于 1977 年上市，但是这两个平台都未能成为持久的个人计算机标准平台。直到 1981 年 IBM 公司推出了 IBM PC，才正式开启了一个新的软件时代。

这个时期的软件是真正独立的软件业诞生的标志，同样也是收缩-覆盖的套装软件引入的开端。微软是这个时代的最成功和最有影响力的软件公司代表。这个时期其他成功的代表公司有一些如 Adobe、Autodesk、Corel、Intuit 和 Novell。

总之，20 世纪 80 年代，软件业以每年 20%的增长率高速发展。美国公司的年收入在 1982 年增长到 100 亿美元，在 1985 年则为 250 亿美元，比 1979 年的数字高 10 倍。

由于 Internet 的介入，一个全新的时代到来了。尽管大部分软件公司还将进一步面临多

个不同标准和平台共存的挑战，软件业也许将会受到新的万维网商机和集中趋势的强烈影响。但还是要指出，不仅仅互联网是软件业的“奇迹”，通信、媒体和最终消费电子业将同样被深深“卷”入其中，这给软件业带来了一个新的方向，并可能由此导致软件业和其他行业的集中。

据数据显示，世界 IT 产业的年复合增长率 1984 年～2004 年为 12%，其中硬件 1984 年的占 IT 产业的 67%，1989 年占 IT 产业的 64%，1994 年占有比例为 55%，1999 年占有比例为 43%。逐年下降的趋势显示硬件的市场份额在减少中。1984 年，硬件的年复合增长率为 9%，IT 服务占 IT 产业的比例仅有 24%，而在 1994 年、1999 年却增长为 29%和 39%，年复合增长率达到 15%，呈上升趋势。软件业的市场份额从 1984 年的 9%发展到 1999 年的 22%，年复合增长率为 17%，增长速度最快。

计算机系统发展的第四个阶段已经不再是着重于单台计算机和计算机程序，而是面向计算机和软件的综合影响。由复杂的操作系统控制的强大的桌面机、广域或局域网络，配以先进的软件应用已成为标准。计算机体系结构迅速地从集中的主机环境转变为分布的客户机/服务器环境。事实上，Internet 本身就可以看做是能够被单个用户访问的“软件”。

一系列软件相关的问题在计算机系统的整个发展过程中一直存在，而且这些问题还会继续恶化：硬件的发展一直超过软件，使得软件难以发挥硬件的所有潜能；开发新程序的能力远远不能满足人们对新程序的需求，同时，开发新程序的速度也不能满足商业和市场的要求；计算机的普遍使用已使得社会越来越依赖于可靠的软件。如果软件“失败”，则会造成巨大的经济损失，甚至有可能给人类带来灾难；人们一直在不断努力建造具有高可靠性和高质量的计算机软件，而且随着硬件的发展这一需求还将不断增长；拙劣的设计和资源的缺乏使得难以支持和升级已有软件。

在计算机发展的早期，计算机系统是采用面向硬件的管理方法来开发的。项目管理者着重于硬件，因为它是系统开发中最大的预算项。为了控制硬件成本，管理者建立了规范的控制和技术的标准，要求在真正开始开发系统之前进行详尽的分析和设计。管理者度量过程以发现哪里还可以进一步改进，他们坚持质量控制和质量保证，并设立规程以管理变化。简言之，他们应用了控制、方法和工具，可以称之为“硬件工程”。

在早期，程序设计被看做是一门“艺术”。几乎没有规范化的方法，也没有人使用它们。程序员往往从试验和错误中积累经验。开发计算机软件的专业性和挑战性，使程序披上了一种神秘的面纱，管理者们很难了解它。软件世界真是完全无序，这是一个开发者为所欲为的时代。

今天，计算机系统开发成本的分配发生了戏剧性的变化。最大的成本项是软件而不再是硬件。在近 20 年里，管理者和很多开发人员在不断地探讨以下问题：

为什么需要那么长时间才能结束开发？

为什么成本如此之高？

为什么不能在把软件交给客户之前就发现所有错误？

为什么在软件开发过程中难以度量其进展？

这些问题以及其他许多问题都表明对软件及其开发方式的关注是必须的，而正是这种关注最终导致了软件工程实践的出现。

许多人相信 21 世纪最重要的产品是信息，软件充分印证了这一观点：它处理个人数据（如个人的金融事务），使得这些数据在局部范围中更为有用；它管理商业信息增强了商业竞争力；它还提供了通往全球信息网络（如 Internet）的途径；它也提供了以各种形式获取信息的手段。

软件产业在世界经济中不再无足轻重。由产业巨子（如微软）做的一个决定可能会带来成百上千亿美元的风险。随着第四阶段的进展，一些新技术开始涌现。面向对象技术在许多领域中迅速取代了传统软件开发方法。虽然关于“第五代计算机”的预言仍是一个未知数，但是软件开发的“第四代技术”确实改变了软件行业开发计算机程序的方式。结合模糊逻辑应用的人工神经网络软件揭示了模式识别和类似人的信息处理能力的可能性。虚拟现实和多媒体系统使得与最终用户的通信可以采用完全不同的方法。

1.1.1 软件产业

软件产业不是一个新概念，也不是一个新兴的产业，而是一个相对成熟的产业。在20世纪90年代，软件始终是风险投资的第一“大户”，直到互联网崛起，才打破了这一态势。软件产业早已造就出一大批地位稳固甚至垄断性的巨头，比如SAP、Oracle、CA等，这些公司的规模已经超过诸多实体公司，成为信息产业的代表，IBM的软件业务更是庞大。但是，为什么谈到软件，总是感觉那样奇妙而神秘，总是那么新鲜而神奇。因为，现在国内人士大凡谈及软件时，总是将其描述成一个笼统的概念，根本没有体现软件产业本身丰富而复杂的内涵。

在企业发展战略层面，软件也只是一个概念性的名称，很少有企业真正把自己的软件战略放置在几十年发展历史和内在规律之下。甚至在信息产业发展战略方面，软件发展也根本没有放眼当今全球软件业最新格局和未来趋势之下，成为有选择、有重点的引导方向。

互联网实验室为了使人们深刻、清晰地认清软件业的真实现状，对整个产业的发展历史、当今格局以及变化规律进行了长期而深入的研究和探讨，总结出一些非常有价值的结论。对软件产业的新格局进行了全面解剖，使人们对软件产业一目了然，可以在纵览全局之后，从容把握。当今世界正处在由工业化向信息化过渡的重要历史时期，信息技术渗透到了国民经济的各个领域，加快了信息化的进程，信息产业悄然成了各国的经济增长点，软件则是信息产业的核心和关键。

对于中国这样一个大国来说，软件不仅是一个具有广阔发展前景的“朝阳”产业，也是一个至关重要的战略性产业。在我国国产硬件制造业取得了瞩目的业绩，国内市场占有率日益提高的情况下，软件特别是应用软件的整体水平已经越来越成为我国民族信息产业发展的关键。因此，必须抓住软件这个关键，突破这个瓶颈，否则将不仅影响软件产业本身，也必然影响硬件制造业的发展，最终影响整体信息产业的健康发展。

软件产业的特点：软件市场容量巨大、软件企业成长迅猛、软件产品种类繁多、软件行业竞争激烈、行业发展日新月异。

软件的工业化生产过程应具备的特点：明确的工作步骤；详细具体的规范化文档；明确规范的质量评价标准。

从工业和信息化部运行监测协调局了解到，近年来我国软件产业的发展稳中有升，数据处理和运营服务持续高速增长，集成电路设计和嵌入式系统软件发展加速，但软件出口仍呈波动和低迷态势，区域发展仍存在较大差异。软件收入增长总体平稳，2012年11月有所加快。2012年1月～11月，我国软件产业总体保持平稳增长，实现软件业务收入2.19万亿元，同比增长27.3%，增速比2012年1月～10月提高1.5个百分点，比电子信息制造业高15个百分点，但低于上年同期5.5个百分点。其中11月完成收入2377亿元，同比增长41.4%，达前11个月最高水平。

从2012年以来，软件行业结构调整步伐继续加快，信息技术服务业比重不断加大，前

11 个月信息技术服务收入比重达到 51%，增速达 27.2%。其中，集成电路设计增长步伐加快，2012 年 1 月～11 月实现收入 678 亿元，同比增长 34%，高出上年同期 10.4 个百分点。

1.1.2 软件的竞争

许多年来，大、小公司雇用的软件开发人员仅仅在公司内部服务，因为每一个计算机程序都是自行开发的，这些“自家”的软件人员控制着成本、进度和质量。如今，软件是一个竞争很强的行业。曾经需要自行开发的软件现在可以在货架上买到，许多公司过去雇用了大量的程序员开发特定的软件，现在大部分软件工作已交给第三方厂商去完成。

成本、进度和质量将是未来若干年中导致软件激烈竞争的主要因素。美国和西欧有很成熟的软件产业，而亚洲（如印度、中国、新加坡、韩国）和东欧的一些国家拥有大量的有天分、受过良好教育的专业人才。这种压力导致了必须迅速采用现代化的软件工程实践的需要，同时软件开发成本也是一个必须认真考虑的因素，因为世界范围的软件从业人员都在减少软件的开发费用。中国软件产业的特点见表 1-1 和表 1-2。

表 1-1 中国软件产业的地理分布

地 区	代表地域	产业优势	地区优势
北京	北方	软件开发、销售	技术、核心市场
广州、深圳	南方	软件制造、开发、引进	制造、口岸市场
武汉	华中	软件销售、发行	技术、发行中心
成都	西南	软件销售、发行	营销、发行
上海	华东	软件开发、引进	技术、口岸城市

表 1-2 中国软件企业优势产品的领域分布

产品领域	软件名称（开发类）
财务及商用管理	用友、金蝶、安易、乔克、佳运、科情、王特、博科、万能、金蜘蛛、远方、金算盘、德克赛诺、东大阿尔派
教育及知识普及	电脑报、金洪恩、清华光盘中心、科利华、武大华软、翰林汇、雅奇、树人、吴思通、深圳多媒体开发公司
游戏及娱乐	金山、目标、智冠（中国台湾）、光荣（中国台湾）奥美
开发工具	雅奇-奔腾、王特、金国科、北大方正、华正 CAD
操作系统	金山
专业软件	华工 CAD、中国地大、武测科大、安易、北大方正、博科

1.2 软件危机与软件工程

随着计算机系统的增多，计算机软件库开始扩展。内部开发的项目产生了上万行的源程序，从外面购买的软件产品加上几千行新代码就可以使用了。这时，用户才意识到，当发现错误时需要纠正所有这些程序（即所有这些源代码）；当用户需求发生变化时需要修改这些程序；当硬件环境更新时需要修改这些程序以适应硬件变化。这些活动统称为软件维护。在软件维护上所花费的精力开始以惊人的速度消耗资源。

从第一台计算机诞生以来，软件的生产就开始了，到目前为止，已经过了程序设计、程序系统和软件工程 3 个时代，其特点见表 1-3。

表 1-3　软件开发的特点

	程序设计	程序系统	软件工程
特点	硬件通用，软件专用 程序规模小，编写者和使用者为同组人	出现“软件作坊”、出现产品软件 “个体化”开发方法	软件开发成为一门新兴的工程学科——软件工程
软件的范畴	程序	程序及说明书	产品软件（项目软件）
主要语言	汇编	高级语言	高级语言系统
软件工作范围	程序编写	程序编写 软件设计和测试	软件生存期
硬件特征	价高、存储量小、可靠性差	降价；速度、容量、可靠性明显提高	向超高速、大容量、微型化发展
软件特征	完全不受重视	软件技术的发展不满足需要，出现了软件危机	开发技术有进步，但未获得突破性进展，软件危机未完全摆脱

更糟糕的是，许多程序的个人化特性使得它们根本不能维护。软件危机出现了。软件危机是指计算机软件开发和维护过程中所遇到的一系列严重的问题。

软件危机爆发于 20 世纪 60 年代末期，虽然人们一直致力于发现解决危机的方法，但是软件危机至今依然困扰着，并没有一种“灵丹妙药”可以完全“治愈”它。

不管称之为“软件危机”还是“软件苦恼”，该术语都是指在计算机软件开发中所遇到的一系列问题。这些问题不仅局限于那些“不能正确完成功能的”软件，还包含那些与“如何开发软件”、“如何维护大量已有软件”以及“开发速度如何跟上目前对软件越来越大的需求”等相关的问题。

引起软件危机的诸多原因可以追溯到软件开发的早期阶段产生的“神话”。它不像古代的神话那样可以给人以经验和教训，而是使人产生了误解和混乱。软件“神话”具有的一些特征使得它们很有欺骗性。例如，它们表面上看很有道理，有时含有一定真实的成分；它们符合人的直觉；它们常常是有经验的实践者发布出来的。

今天，大多数专业人员已经认识到这些“神话”误导了人们，给管理者和技术人员都带来了严重的问题。但是，旧的观念和习惯难以改变，软件“神话”仍被不少人相信着。

管理者的“神话”：负责软件的管理者像大多数其他行业的管理者一样，都有着巨大的压力，例如要维持预算、保持进度及提高质量。就像溺水者抓住一根救命稻草一样，软件管理者常常抓住软件“神话”不放，以期这些“神话”能够缓解其压力。

“神话”1：已经有了关于建造软件的标准和规程的书籍，难道它们不能给人们提供所有其需要知道的信息吗？

事实：不错，关于建造软件的标准的书籍已经存在，但真正用到了它们吗？软件实践者知道它们的存在吗？它们是否反映了现代软件开发的过程？它们完整吗？很多情况下，对于这些问题的答案均是“不”。

“神话”2：已经有了很多很好的软件开发工具，而且，为它们配备了最新的计算机。

事实：为了使用最新型号的主机、工作站和 PC 去开发高质量的软件，已经投入太多的费用。实际上，计算机辅助软件工程（Computer Aided Software Engineering，CASE）工具相比起硬件而言对于获得高质量和高生产率更为重要，但大多数软件开发者并未使用它们。

“神话”3：如果已经落后于计划，则可以增加更多的程序员来赶上进度。

事实：软件开发并非像制造一样是一个机械过程。给一个已经延迟的软件项目增加人手只会使其更加延迟。看起来，这句话与人的直觉认识正好相反。但实际上，增加新人使原来正在工作的开发者必须花时间来培训新人，这样就减少了他们花在项目开发上的时间。人手可以增加，但只能是在计划周密、协调良好的情况下。

用户的“神话”：需要计算机软件的用户可能就是邻桌的人，或是另一个技术组，也可能是市场/销售部门，或另外一个公司。在许多情况下，用户相信关于软件的“神话”，因为负责软件的管理者和开发者很少去纠正用户的错误理解。“神话”导致了用户对软件抱有过高的期望值，并引起对开发者的极度不满。

“神话”4：有了对目标的一般描述就足以开始写程序了——细节可以以后再补充。

事实：不完善的系统定义是软件项目失败的主要原因。关于待开发项目的应用领域、功能、性能、接口、设计约束及确认标准的形式化的、详细的描述是必需的。这些内容只有通过用户和开发者之间的通信交流才能确定。

“神话”5：项目需求总是在不断变化，但这些变化能够很容易地满足，因为软件是灵活的。

事实：软件需求确实是经常变化的，但这些变化产生的影响会随着其引入的时间不同而不同。如果很注重早期的系统定义，这时的需求变化就可被很容易地满足。用户能够复审需求，并提出修改的建议，这时对成本的影响会相对较小。如果在软件设计过程中才要求修改，那么对成本的影响就会提高得很快。资源已经消耗了，设计框架已经建立了，这时的变化可能会引起大的改动，需要额外的资源和大量的设计修改，例如额外的花费。实现阶段（编码和测试阶段）功能、性能、接口及其他方面的改变对成本会产生更大的影响。当软件已经投入使用后再要求修改，这时所花的代价比起较早阶段做同样修改所花的代价可能是呈几何级数级的增长。

开发者的“神话”：那些至今仍被软件开发者相信的“神话”是由几十年的程序设计文化“培植”起来的。正如在一开始就提到的，在软件的早期阶段，程序设计被看做是一门艺术。这种旧的观念和方式是很难改变的。

“神话”6：一旦写出了程序并使其正常运行，设计的工作就结束了。

事实：有人说过，“越早开始写程序，就要花越长时间才能完成它”，产业界的数据表明，在一个程序上所投入的50%～70%的努力是花费在第一次将程序交给用户之后。

“神话”7：在程序真正运行之前，没有办法评估其质量。

事实：从项目一开始就可以应用的最有效的软件质量保证机制之一是正式的技术复审。技术复审是“质量的过滤器”，比通过测试找到某类软件错误要有效得多。

“神话”8：一个成功项目唯一应该提交的就是运行程序。

事实：运行程序仅是软件配置的一部分，软件配置包括：程序、文档和数据。文档是成功开发的基础，更重要的是，文档为软件维护提供了指导。

许多软件专业人士认识到上述这些“神话”是错误的。但令人遗憾的是，旧的观念和方法培植了拙劣的管理和技术习惯，虽然现实情况已经有所改观，但仍旧需要更好的方法。对软件现实的认识是形成软件开发的实际解决方案的第一步。

软件危机的具体表现如下：

1）软件开发的进度难以控制，经常出现经费超预算、完成期限一再拖延的现象。一个复杂的软件系统需要建立庞大的逻辑体系，而这些往往只存在于人们的大脑中，正如一个大项目负责人所说：“软件人员太像皇帝新衣故事中的裁缝，当检查软件开发工作时，所得到的回答总是‘正忙于编织这件带有魔法的织物，只要一会儿，就会看到这件织物是极其美丽的’。但是什么也看不到，什么也摸不到，也说不出任何一个有关的数字，没有任何办法得到一些信息说明事情确实进行得非常顺利，而且已经知道许多人最终编织了一大堆昂贵的‘废物’，还有不少人最终什么也没有做出来。”

2）软件需求在开发初期不明确，导致矛盾在后期集中暴露，从而给整个开发过程带来

灾难性的后果。软件需求的缺陷将给项目成功带来极大风险，如产品的成本过高、产品的功能和质量无法完全满足用户的期望等。即使一个项目团队的人员和配备都很不错，但不重视需求缺陷也会付出惨痛的代价。导致需求缺陷的主要原因包括需求的沟通与理解、需求的变化与控制、需求说明的明确与完整。模棱两可的需求所带来的后果便是返工，一些认为已做好的事情，其返工会耗费开发总费用的 40%，而 70%～85%的重做是由于需求方面的缺陷引起的。

3）由于缺乏完整规范的资料，加上软件测试不充分，从而造成软件质量低下，在运行中出现大量问题。

例如：1985 年～1987 年，至少有两个病人是死于 Therac-25 医疗线性加速器的过量辐射，事故起因是控制软件中的一个故障。1966 年，IBM 360 的操作系统花费了 5000 多人一年的工作量，写出的近 1 万行代码错误百出。每次的新版本就是从前一版本中找 1000 个程序错误而修正的结果。1963 年，美国用于控制火星探测器的计算机软件中的一个“,”被误写为“.”，而致使飞往火星的探测器发生爆炸，造成高达数亿美元的损失。美国丹佛的新国际机场自动化行李系统软件投资 1.93 亿美元，计划 1993 年万圣节启用。但开发人员一直为系统错误困扰，屡次推后软件启用时间，直到 1994 年 6 月，设计者承认仍无法预测何时能启用。1996 年，欧洲阿里亚纳 5 型运载火箭坠毁，造成 5 亿美元损失，其原因是控制软件中的一个错误。

由此可见，软件错误的后果是十分严重的，医疗软件的错误可能给病人的生命造成危险，银行软件系统的错误会使金融混乱，航管软件系统的错误会造成飞机失事等。

由于认识到软件的设计、实现、维护和传统的工程规则有相同的基础，于是北大西洋公约组织（NATO）于 1967 年首次提出了“软件工程（Software Engineering）”的概念。关于编制软件与其他工程任务类似的提法，得到了 1968 年在德国召开的 NATO 软件工程会议的认可。会议委员会的结论是，软件工程应使用已有的工程规则的理论和模式，来解决所谓的“软件危机”。

软件危机至今仍然困扰着开发设计者和用户，这表明软件生产过程在许多方面和传统的工程相似，但却具有其独特的属性，存在着特殊的问题。

这些问题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。

概括来讲，软件危机包括下述两方面的问题：如何开发软件，以满足用户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。

软件生产不能满足日益增长的客观需要，可谓“供不应求”。软件开发成本和进度估计不准确。节约成本所采取的“权宜之计”损害了软件的质量，引起用户的不满。软件开发人员对用户的需求缺乏了解。“闭门造车”导致软件产品不符合实际需要，软件产品质量差。软件质量保证技术（审查、复查、测试）没有贯穿于开发的全过程，软件可维护性差。错误难以改正，新功能难以增加，“再用性”的软件未能实现，重复开发类似的软件。没有文档资料、资料不完整，这些都给软件交流、管理、维护造成了困难。软件成本逐年上升，软件的价格昂贵。

软件工程是指导计算机软件开发和维护的一门工程学科。它采用工程的概念、原理、技术和方法开发维护软件，把经过时间考验、被证明是正确的管理技术和当前能够得到最好的技术方法结合起来，以经济地开发出高质量的软件，并有效地维护它。

软件危机产生的原因有：软件本身的特点、对软件开发与维护存在许多错误认识和做法、软件开发与维护的方法不正确、开发人员素质低下等。

总之，为了解决软件危机，既要有技术措施，又要有必要的组织管理措施。软件工程正

是从管理和技术两方面研究如何更好地开发维护计算机软件的一门新兴学科。

软件工程准则可以概括为6条基本原理：用分阶段的生存周期计划严格管理；坚持进行阶段评审；实行严格的产品控制；采用现代程序设计技术；应能清楚地审查结果；合理安排软件开发小组的人员。

软件工程学的内容可包括理论、结构、方法、工具、环境、管理、规范等。软件工程的目标是在给定成本、进度的前提下，开发出具有可修改性、有效性、可靠性、可理解性、可维护性、可重用性、可适应性、可移植性、可追踪性和可互操作性，并满足用户需求的软件产品。在具体项目的实际开发中，让以上几个目标都达到理想的程度往往是非常困难的。而且上述目标很可能是互相冲突的。例如若降低开发成本，很可能同时也降低了软件的可靠性。另一方面，如果过于追求提高软件的性能，则可能造成开发出的软件对硬件有较大的依赖，从而直接影响到软件的可移植性。

可修改性（Modifiability）：对软件系统进行修改而不增加其复杂性。它支持软件调试与维护，是一个难以度量和难以达到的目标。

有效性（Effectiveness）：能有效地利用计算机的时间资源和空间资源。

可靠性（Reliability）：具有能够防止因概念、设计和结构等方面的不完善而造成的系统失效，具有可挽回因操作不当造成软件系统失效的能力。

可理解性（Understandability）：系统具有清晰的结构，能直接反映问题的需求。可理解性有助于控制软件系统的复杂性，并支持软件的维护、移植和重用。

可重用性（Reusability）：指软件可以在多种场合使用的程度。

可适应性（Adaptability）：采用流行的程序设计语言、运行环境、标准的术语和格式。

可维护性（Maintainability）：指软件产品交付使用后，在实现改正潜伏的错误、改进性能等属性、适应环境变化等方面工作的难易程度。由于软件的维护费用在整个软件生存周期中所占比重较大，因此，可维护性是软件工程中的一个十分重要的目标。软件的可理解性和可修改性支持软件的可维护性。

可移植性（Portability）：把程序从一种计算环境（硬件配置和操作系统）转移到另一种计算环境，并使之正常运行的难易程度。

可追踪性（Traceability）：对软件进行正向和反向追踪的能力。

可互操作性（Interoperability）：多个软件要素相互通信协同完成任务的能力。

在软件开发过程中，为了达到软件开发目标，必须遵循抽象、信息隐藏、模块化、局部化、一致性、完整性和可验证性等原则。

抽象（Abstraction）：抽取各个事物中共同的最基本的特征和行为，暂时忽略它们之间的差异。一般采用分层次抽象的方法来控制软件开发过程的复杂性。抽象可增强软件的可理解性并有利于开发过程的管理。

信息隐藏（Information hiding）：将模块内部的信息（数据和过程）封装起来。其他模块只能通过简单的模块接口来调用该模块，而不能直接访问该模块内部的数据或过程，即将模块设计成“黑箱”。信息隐藏可使开发人员把注意力集中于更高层次的抽象上。

模块化（Modularity）：模块是程序中一个逻辑上相对独立、具有良好的接口定义的编程单位，包括过程、函数、类、程序包等。模块化是将复杂的系统分解为一个个相对独立的模块来加以实现，有助于抽象和信息隐藏以及表示复杂的系统。

局部化（Localization）：即在一个物理模块内集中逻辑上相互关联的计算资源。局部化支持信息隐藏，从而保证模块之间具有松散的耦合、模块内部有较强的内聚。这有助于控制每一个解的复杂性。

一致性（Consistency）：指整个软件系统（包括程序、数据和文档）的各个模块应使用一致的概念、符号和术语；程序内部接口应保持一致；软件与环境的接口应保持一致；系统规格说明应与系统行为保持一致等。

完整（备）性（Completeness）：指软件系统不丢失任何重要成分，完全实现所需的系统功能的程度。为了保证系统的完整性，在软件的开发和维护过程中需要严格的技术评审。

可验证性（Verifiability）：开发大型软件系统需要对系统逐层分解。系统分解应遵循易于检查、测试、评审的原则，以使系统可验证。

确保正确性的方法通常是有条件的。一个完整的软件系统，甚至是以今天的标准来看的一个小型软件系统，涉及如此众多的区域，以致它不可能在一个单一层次处理所有的组件和特性来保证它的正确性。换而言之，分层的方式是必需的。每个层次依赖于较低的一层：在达到正确性的条件下，我们只关心每个层次的正确性保证（这是假设在较低的层次、正确的基础上的）。这只是现实的技术，它完成关系分离，并且把有限的问题集中在一个层次上，但不能有效地检查在高级语言中的一个程序是否正确，除非能够假定手上的编译器能正确地实现。这并不意味着必须盲目地信赖编译器，只不过把问题分成两个方面，编译器的正确性和程序所用语言的语义正确性。

1.2.1　软件特征

软件是逻辑的而不是物理的产品。实际上，逻辑往往只存在于人的大脑当中，软件的开发过程极难控制；因此，软件具有如下与硬件完全不同的特征：

1）软件是由开发或工程化形成的，而不是传统意义上的制造产生的。软件是通过人们的智力活动，把知识与技术转化成信息的一种产品，是在研制、开发中被创造出来的，没有明显的制造过程。这意味着软件项目不能像硬件制造项目那样来管理。

虽然在软件开发和硬件制造之间有一些相似之处，但两者本质上是不同的。两者都可以通过良好的设计获得高质量，但硬件在制造过程中可能会引入质量问题，这种情况对于软件而言几乎不存在（或是很容易改正）。软件成为产品之后，其制造只是简单的复制而已；两者都依赖于人，但参与的人和完成的工作之间的关系不同；两者都是制造一个产品，但方法不同。

在 20 世纪 80 年代中期，“软件工厂”的概念被正式引入，这个术语并没有把硬件制造和软件开发认为是等价的，而是借此促进软件开发中模块化设计、组件复用等意识的全面提升。

2）软件不会“磨损”。图 1-2 为硬件和软件的故障变化曲线。

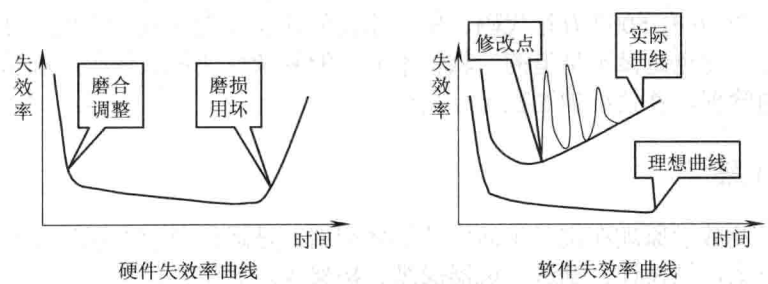

图 1-2　硬件和软件的故障曲线

图 1-2 描绘的是随着时间的改变，硬件故障率的变化曲线，又常被称作“浴缸曲线”，