21世纪大学计算机系列教材

软件工程

臧铁刚　主　编
冷晟　钱晓明　朱健江　副主编

科学出版社

21世纪大学计算机系列教材

软件工程

臧铁钢 主编

冷晟 钱晓明 朱健江 副主编

科学出版社

内 容 提 要

本书从实用的角度出发，全面介绍了软件工程的基础知识和软件工程技术方法。全书共分为 10 章，内容涵盖了软件工程概述，软件系统可行性研究与需求分析、软件设计技术、编码及程序设计语言、软件的技术量度及质量保障、软件测试技术、软件维护技术、软件项目管理以及新型的软件工程技术，最后还讲述了软件工程文件的相关内容。此外，为方便读者巩固所学知识，每章最后均配有适量的习题。

本书内容编排合理，在介绍传统理论体系的基础上，融入当前软件工程的最新发展和技术，并通过大量的练习和案例分析，帮助读者真正掌握书中内容。该书可作为高等院校计算机及相关专业的教材，也可作为软件项目管理者和软件开发人员的参考用书。

图书在版编目（CIP）数据

软件工程/臧铁钢主编.—北京：科学出版社，2009

21 世纪大学计算机系列教材

ISBN 978-7-03-024293-8

Ⅰ. 软… Ⅱ. 臧… 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（2009）第 041565 号

责任编辑：王少华 / 责任校对：科 海

责任印刷：科 海 / 封面设计：林 陶

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

北京市艺辉印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2009 年 5 月 第 一 版 开本：16 开

2009 年 5 月第一次印刷 印张：16.5

印数：0 001~3 000 字数：389 000

定价：28.00 元

（如有印装质量问题，我社负责调换）

前　言

随着计算机技术应用的不断深入，软件已不容置疑地成为信息化的核心。软件的复杂度和规模在近几十年与日俱增，使得软件开发成为了一项需要科学理论和相应技术支持的复杂的系统工程。软件工程就是这样一门能指导并管理软件开发过程的学问。它的产生和发展是人们对软件开发规律深入研究的结果。从20世纪60年代后期提出软件工程化的概念，到现在已发展成为一门综合性的学科，软件工程学科一直致力于研究软件开发的方法，配套发展相应的实现工具，构造良好的开发环境。

软件工程是计算机科学与技术专业教学计划中的核心课程，也是一门培养软件开发技术人员的必修课程。考虑到软件工程的实践性较强、发展迅速等特点，本书在编撰时突出了理论与实践相结合的思想，并引进了一些目前较新的开发技术，使读者不仅能系统地了解软件工程的理论与技术体系，还能把握其发展脉络，从而了解到新型的软件工程技术。本书在每一章最后都附有难度适宜的习题，有助于读者强化所学的知识。

本书共分为10章：第1章对软件技术的发展历程及软件工程学科产生的背景进行了全面的阐述；第2章对软件系统的可行性研究和需求分析进行了全面的阐述；第3章对软件设计中的结构设计和详细设计分别进行了深入的阐述，读者可以了解软件设计的概念、方法和支持工具；第4章全面介绍程序编码技术及程序设计语言；第5章阐述了软件质量保障技术方面的内容，以及为了提高软件的质量而建立的质量保证体系；第6章介绍了有关软件测试方面的内容，使读者能对软件测试的方法有所了解；第7章阐述了软件系统维护的概念、流程，以及维护方法等方面的内容；第8章从工程管理的角度对软件项目管理的方式和方法进行了全面阐述；第9章较全面地展示了新型的软件工程技术；第10章说明了与软件开发过程相关的一系列工程文件的一般性内容和格式，这对在实际开发中撰写规范化的技术文档非常有益。此外，为方便读者巩固所学知识，每章最后均配有适量的习题。

本书第1、2章由臧铁钢负责编撰并统编全书，第3、4、9章由冷晟负责编撰，第5、6、10章由钱晓明负责编撰，第7、8章由朱健江负责编撰。

由于时间仓促，加之编者水平有限，不足之处在所难免，恳请广大读者指正。

编者

2009年3月

目　录

第1章

软件工程概述

本章对软件技术的发展历程及软件工程学科产生的背景进行了全面的阐述，重点说明了软件工程的基本内容及其目标，以及在软件工程中应用到的基本原理、原则、软件生命周期模型及软件工程工具等内容。通过学习本章，读者能初步了解软件工程的学科框架。

1.1 软件工程概况

1.1.1 计算机软件简介

20世纪40年代出现的计算机是科学技术进步的结晶，在人类文明史上具有历史性的意义。计算机系统的应用使人类的脑力得到了解放，思维能力获得了极大的延伸。如今，社会生活的各个领域都离不开计算机，计算机已成为信息化社会的基础。

计算机系统由硬件系统和软件系统组成，这两部分互为依赖，缺一不可。由此可见计算机软件的重要地位。计算机的硬件和软件相互促进，共同发展。硬件系统的每次技术突破，都为软件技术的发展提供了更为广阔的发展空间；软件应用的需求也为硬件系统的发展指明了方向。

计算机硬件是计算机系统中各种设备的总称。硬件系统直观可触，是计算机工作的物理平台。自计算机出现以来，作为计算机硬件核心的处理器芯片已经历了电子管、晶体管、集成电路和大规模集成电路四个时代，现正在向集成度更高和运行速度更快的方向发展，计算机的功能也越来越强。1965年，戈登·摩尔根据统计材料总结出了所谓的摩尔定律。该定律表明：芯片上可容纳的晶体管数目，约每隔18个月就会增加一倍，性能也将提升一倍。由于人类科学技术的累积加速效应，近40年来，硬件的发展速度极其迅猛，摩尔定律周期正在呈逐渐递减的趋势。

在计算机投入应用后不久，人们意识到了程序的独立性，并逐渐形成了软件的概念。随着计算机应用的深入，软件的概念也不断地变化和扩展。

人们对软件定义的认识经历了一个较长的过程。20世纪50年代，程序设计还处于个体手工生产阶段，程序是由设计者本人开发和维护的结构简单、功能单一、可靠性差的小型源程序。所以，人们认为软件就是程序，软件系统就是程序系统。60年代，软件的功能、规模日益增大，软件的开发只能由群体来承担，软件设计进入到作坊式生产阶段。为了提高程序的可维护性，人们认为软件等于程序加文档。所谓文档就是指软件开发过程中与各环节相关的资料，但不包括与软件开发过程相关的管理文档。对管理文档的全面认识是从70年代开始的。在这一时期，随着软件需求规模、数量的剧增和交付的紧迫，要求大型程序的开发活动按照工程项目的模式运作。此时，软件工程方法被引入到软件开发过程中，对软件定义的认识也得到了提高。目前，对于计算机软件概念的一种公认的解释是：软件是计算机系统中与硬件相互对应的另一部分，是程序、数据及相关文档的完整集合。即：

软件=程序+数据+文档

由上式可知，程序和软件有着不同的概念，软件的范畴要比程序大。程序是按事先设计的功能和性能要求执行的指令序列；数据是使程序正常操作的信息；文档是与程序开发、维护和使用有关的图文材料，文档包括了管理文档。数据则不仅包括初始化数据、测试数据、研发数据、运行数据、维护数据，还包括软件企业积累的项目工程数据和项目管理数据中的大量决策原始记录数据。80年代，人们开始认识到软件管理是一个过程管理。因此，到90年代，出现了软件过程能力成熟度模型，人们开始研究软件过程管理的具体内容与方法。1997年出现的统一建模语言（Unified Modeling Language，UML）就是突出的研究成果之一。UML的目标之一就是为开发团队提供标准通用的设计语言来开发和构建计算机应用程序。UML的出现标志着软件开发进入到了一个以规范化、过程化、工厂化、大型化、自动化为特征的较成熟的阶段。

一方面，作为人造产品，计算机软件与一般的产品具有共同点，另一方面，计算机软件也具有与一般产品不同的特点。

（1）软件体现的是达到某种计算目标的一系列逻辑流程，这种逻辑流程在静态情况下是不可见的，只有处于动态运行时才能体现出其效果。软件体现的逻辑流程可以存在于存储介质上。

（2）复制就是软件的生产，就是把逻辑流程转存于存储介质上。这一过程花费极小，主要生产成本就是存储介质的成本。大批量发行的存储介质成本很低，故软件的生产成本相当低廉。

（3）软件只存在退化的现象，而不存在磨损和老化的问题。即因为软件会为了适应新的运行环境和需求而进行修改，当修改成本最终变得不可接受时，软件就会出现退化，从而被抛弃。

（4）确定的软件需要在特定的硬件平台上运行，软件对硬件运行环境有着不同程度的依赖性。因此，软件存在可移植性问题。

（5）软件的开发涉及的影响因素众多，如管理方法、开发模型、开发人员的观念和理念等，都直接影响到软件的开发过程。

（6）软件的开发及维护成本相当昂贵，需要投入大量的人力，是高风险的生产活动。很多软件的开销大大超过硬件的开销。

（7）很多软件涉及诸多社会因素，如体制运作及管理、人的观念和心理等，这些因素都会直接影响到软件的开发和应用。

计算机系统中，硬件平台根据所应用的领域会有所不同，但其基本的结构却大同小异。软件则更能体现出应用领域的特点，因而差异很大。从计算机开始应用时的科学计算软件到现今大量出现的非数值计算软件，软件已从最初作为硬件的附属，发展成为计算机系统不可缺少的组成部分。随着计算机技术应用的深入，计算机软件形成了内容丰富的体系结构。从软件使用和开发的角度来看，给出软件体系的描述是必要的。可以从多种角度对软件进行分类，按功能来分可分为系统软件、支撑软件和应用软件，按工作方式来分可分为实时软件、分时软件、交互式软件、批处理软件，按规模来分可分为微型软件、小型软件、中型软件、大型软件、超大型软件、极大型软件，按服务对象来分可分为定制软件、产品软件，按应用领域来分可分为操作系统、数据库管理系统、软件开发系统、工程软件、办公软件、财会软件、网络工具软件、图形图像处理软件、多媒体软件、游戏软件、教育软件等，按收费方式来分可分为商品软件、共享软件、自由软件。

软件是动态发展着的，软件的概念还会因其自身的发展而不断变化，软件的体系结构也会不断地丰富，各类软件的功能和复杂程度也将不断提升。可以说，软件的价值就体现在其不断适应需求而进行的改变上。

1.1.2 软件的发展历程

“二战”期间，出于军事需要，研制出了世界上第一台计算机。计算机投入使用后，人们就开始意识到软件的重要性。但是，当时人们更注重昂贵的硬件，对软件的开发相对不是很重视。指导软件开发的理论尚未成形，软件开发工具也很原始。软件的开发只能以小作坊的形式进行，软件的性能和可靠性往往取决于开发者的技术素质，开发的随意性很大。随着计算机的性能和应用水平的提高，人们深刻地理解了软件开发是一门学问，需要构建相应的理论和技术体系。回顾软件技术的发展史，可知其经历了以下三个阶段。

1. 程序设计阶段（1946年－1956年）

在这一阶段，关于软件的相关概念还没有正式出现。人们认为软件就是程序，开发软件就是编写程序指令。当时的软件结构功能简单，规模小。软件生产方式主要是个体手工劳动，使用的工具大多是机器语言、汇编语言；软件开发者片面追求编程技巧和程序运行效率，代码缺乏规范制约，使得程序难以维护，程序可靠性很差。这一阶段的软件开发只重视编程，不重视程序设计方法，没有系统化的软件开发理论支持，几乎没有开发文档资料保存下来。到20世纪60年代末期，由于软件开发方法的落后与不断提升的软件规模和复杂度间的矛盾，爆发了影响软件技术发展进程的重要事件——软件危机。

2. 程序系统阶段（1956年－1968年）

这一时期，集成电路被广泛应用，计算机硬件水平得以大大提高，软件技术有了技术

突破的条件和动力。由此，程序开始了系统化和商品化，软件规模和数量也因此急剧增加。生产方式由个体劳作过渡到作坊式的小集团合作化生产。软件生产工具采用了高级语言，生产率极大提高。在软件生产方法上，提出了模块化、结构化、由顶向下逐步求精、程序变换、程序的推理与综合、数据类型抽象、程序正确性证明、符号测试等各种程序设计和验证的方法。提出了以正确性、结构清晰、便于阅读、便于测试和维护为软件质量的标准，使软件的概念得以扩展，人们普遍接受软件就是程序及其规格说明书的综合的观念。这一时期，滞后的开发技术已不适应规模大、功能结构复杂的软件的开发，软件危机还在延续，积累的大量经验形成了软件技术进一步发展的基础。

3. 软件工程阶段（1968 年以后）

该时期的计算机硬件正向超高速、大容量、微型化以及网络化的方向发展，计算机的应用已经深入到社会生活中的各个领域。“软件作坊”式的生产方式受到了批判，软件业开始打破软件开发的个体化特征，软件开发有了软件工程化的设计原则、方法和标准可以遵循，以软件产品化、系列化、工程化、标准化为标志的软件产业逐渐兴起。这一时期的生产方式是工程化的生产，强调软件的生命周期，网络、分布式技术、面向对象技术成为了软件开发的基础。目前，计算机应用正在逐渐向企业计算机化、社会信息化的计算机系统工程阶段过渡，以满足现代社会各种计算机应用对计算机软件的巨大需求。

以上三个阶段描述了软件技术发展的过程，即从无序到有序，从自由发挥到规范管理，从简单到复杂，从不可靠到更可靠。随着计算机系统的发展，新的软件技术还将不断出现，有关软件的理论和方法还将进一步丰富。

1.1.3 软件工程的产生和发展

软件工程的产生是计算机发展到一定阶段的必然结果，其产生的诱因是软件危机。人们在反思软件危机产生的原因，并提出摆脱软件危机的系列方法的过程中，形成了软件工程的理论构架。值得注意的是软件工程学科的产生一开始就是和应用紧密相联的。因此，软件工程的迅猛发展得益于实践的丰富营养。

在软件技术发展的程序系统阶段，计算机硬件发展较为迅速，社会认识到了计算机强大的能力，计算机得以在各个领域普及。计算机被应用于军事、商业、工业等领域，应用领域的增加使得对软件的需求急剧增加，软件的种类增多，规模膨胀，并产生了对于软件维护的强烈需求。然而，这一时期仍然沿用的是个体化的软件开发方法。落后于硬件水平的软件开发和维护技术使得软件的开发和维护质量令人失望。在用户的需求和软件开发支持技术之间就形成了尖锐的矛盾，导致所谓的软件危机。此时，软件业面临的问题可概括为：以何种方式开发软件以满足用户日益增长的需求？如何维护已有的数量庞大的软件？

软件危机的具体表现之一就是软件可靠性非常差，导致投入的开发成本与开发成果严重不匹配，很多软件项目的成本远远高于预算。从客观上讲，这种情况严重打击了人们开发软件的热情，对计算机技术的发展起到了消极作用。但从另一方面来看，软件危机暴露了当时在软件开发方法上的缺陷，激励人们寻找更好的解决方案。一个典型的软件危机的例子就是IBM公司开发的OS/360系统。当时，IBM公司投资了几千万美元，花费了五千多

人年，历时数年才完成开发。投入使用后问题不断，错误重重，软件退化严重。昂贵的维护费用让用户和开发者都不能承受。其他大型软件也有着OS/360同样的结局。1979年，美国财政部对政府投资开发的软件项目进行了调查，结果令人触目惊心：47%的资金花费在从未使用过的软件上，另外29%的资金花费在那些半途而废和交付后还需进一步完善的软件上，可以称之为成功的项目仅占3%～4%。当时人们认为软件投资技术风险很大，这种观念扼制了软件的发展。

一般而言，软件危机的表现和原因如下：

（1）由于缺乏软件开发的科学指导，开发者不能准确地估计软件开发的成本和进度，在有限的时间内交付使用可能导致软件质量的下降。成本规划往往大大低于实际的成本，可能导致软件开发进度的停滞。

（2）由于缺乏使开发人员与用户进行交流的有效机制，开发人员常常不能很好地理会用户的本意，用户也不能很好地理解开发者的意图。从而会造成用户对已完成的软件系统不满意的结果。

（3）由于软件测试技术的落后，测试过程的工作自然不够充分，加之没有好的软件质量保证技术，导致成品软件的质量较差。

（4）由于软件开发方法的缺陷，软件不易于二次修改，故软件的可维护性差。这使得程序中的错误难以改正，改正错误后的程序又可能引入新的错误，使人束手无策。由于程序对于硬件平台的依赖使软件的可移植性很差，软件很难适应不同的运行环境。软件的可复用性差，导致大量的软件人员要重复开发。

（5）开发过程缺乏标准和规范的指导，各个开发组织都有自己的开发方法，开发组织之间进行工作交接的流程不规范。这是导致开发时间延长的重要原因。

（6）软件开发生产率的速度难以满足对软件需求的增长速度，软件产品供不应求。软件短缺出现完全是技术瓶颈造成的。

（7）由于软件开发和维护方法不当，使得软件的成本居高不下，已严重制约了软件的开发。开发项目的萎缩进一步加剧了开发资金与开发项目间的矛盾。

软件工程正是在解决软件危机问题的过程中形成的一门综合技术与管理两个方面的新兴学科，并逐渐成为指导计算机软件开发、维护、管理的理论依据。造成软件危机的原因很多，归结起来主要有两个方面：一是与软件高投入的特点有关，二是与软件开发与维护方法不当有关。因此，人们采取了以下三种解决软件危机的措施：

（1）作为一种人造产品，软件的开发过程可以借鉴一般工业品的生产模式，即工程化的生产模式。软件开发不应只是个体化的劳动，而更应该是由组织良好、管理严密的生产集体完成的工程项目。软件的开发应该吸收和借鉴一般工程项目行之有效的管理方法，以提高软件的生产率和质量。软件开发工程化的核心是标准化和规范化。软件工程标准主要有5个层次：国际标准、国家标准、行业规范、企业标准和项目规范。

（2）软件危机的重要原因之一就是缺乏相应理论的指导，落实到具体的层面上就是缺乏体现理论效果的工具。采用先进的技术、方法、工具开发和设计软件可以解决软件开发过程中的技术支持的问题，即采用先进的管理技术、规范的开发方法和模型，以及各种提高开发效率的软件工具等。

（3）不同的资源组织方式会有完全不同的运作结果。高效的组织管理措施是可以优化

开发资源，提高运行的效率。从经验来看，采用高效的组织管理措施是提高软件生产率的有效方法。

在寻求以上三种措施的具体实现方法的过程中，逐渐形成了软件工程的概念和方法体系。软件危机使研究者和软件开发人员意识到：作为一种特殊的产品，软件开发工作也应跟机械产品、电子产品、化工产品一样，遵循系统工程的思想和方法，才能得到好的开发效果。北大西洋公约组织于1968年在德国举行的一次学术会议中首先提出了软件工程的概念，引起了人们的重视。经过不懈的努力，20世纪70年代末至80年代初软件工程学科开始形成，并被广泛应用于实践。

对软件工程的定义体现了人们对软件开发过程规律认识的不断深入。1983年IEEE（Institute of Electrical and Electronics Engineers）对软件工程下的定义是：软件工程是开发、运行、维护和修复软件的系统方法。1993年IEEE进一步给出了一个更全面的定义：

（1）把系统化的、规范的、可量度的信息途径应用于软件开发、运行和维护的过程，也就是把工程化应用于软件中。

（2）研究（1）中提到的途径。概括起来，所谓软件工程就是采用工程化的概念、原理、技术和方法来开发和维护软件，把经过时间考验证明是正确的管理技术和当前好的技术方法结合起来指导计算机软件开发和维护的工程学科。

软件工程已经经历了传统软件工程时代、对象软件工程时代、过程软件工程时代、构件软件工程时代。目前的软件工程发展趋势正在这四个时代的基础上朝着流水线装配软件工程的方向发展。

1.2 软件工程的基本概念

1.2.1 软件工程的基本内容

软件工程是一门系统工程学科，目的是成功地开发软件系统。软件工程追求的总体目标可概括为：选择适当的方法论做指导，使用相应的工具做手段，运用成熟的技术从事软件开发活动，最终实现提高软件产品质量和开发效率，得到可靠性高的、经济适用的、易于维护的软件产品。

软件工程的基本内容主要包括软件开发模型、软件开发方法、软件工程工具、软件过程管理四个方面。这四个方面映射了软件开发的各个阶段，各有各的功用，组成了一个完整的方法体系。现简单介绍如下：

（1）软件开发模型就是软件开发流程的模型，它确定了一个软件开发过程的基本模式。采用不同的开发模型对开发维护成本、开发质量等都有重要的影响。因此，软件开发项目组在一开始就应该选定合适的软件开发模型。常见的软件开发模型有瀑布模型、增量模型、原型模型及迭代模型等。

（2）软件开发方法为软件开发提供了具体的指导，明确了软件开发的核心对象。软件开发方法对软件系统的开发效率和可靠性至关重要。目前常用的软件开发方法有面向过程的方法、面向数据的方法和面向对象的方法。

（3）软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境。能有效提高软件开发的效率，并减少人为原因造成的错误。智能化是软件工程工具未来发展的方向。

（4）软件过程管理是对软件开发的过程进行全程监控，以保证开发和维护任务的顺利进行。软件过程管理包含了多方面的任务，如项目的计划与成本估算、软件系统的需求分析、数据结构和系统总体结构的设计、算法过程的设计、编码、测试以及维护等。软件工程方法常采用相应的标准（如ISO9000）来保证软件的质量。

对一个具体的软件工程项目而言，采用软件工程的技术和管理方法组织软件的开发，最终目标是希望获得软件项目的成功，使软件产品能正确、可靠和高效率地运行。软件工程的目标主要包括可靠性、可适应性、有效性、可理解性、可互操作性、可复用性、可追踪性、可维护性和可移植性等。在实际的开发过程中，要想使以上的几个目标都能达到理想程度往往是很困难的，有些目标之间可能存在着冲突。因此，软件开发项目的管理要努力协调以上目标，突出重点，取得平衡。

1.2.2 软件工程的基本原理和原则

自软件工程学科成型以来，专家、学者们陆续提出了很多关于软件工程的原理或准则。由软件工程专家B. W. Boehm提出的软件工程的7条基本原理被普遍采用。这7条原理是确保软件产品质量和开发效率原理的最小集合。这7条原理是互相独立而完备的，其中任意6条原理的组合都不能代替另一条原理。可以证明其他软件工程原理都可以由这7条基本原理的任意组合蕴涵或派生。这7条原理是：

（1）应用生命周期理论，分阶段地计划、管理和控制软件开发过程

该原理是在吸取前人的教训的基础上而提出来的。经统计发现，由于计划不周而失败的软件项目占全部失败软件项目的一半左右。该原理把软件生命周期划分成若干个阶段，并相应地制定出切实可行的计划，然后严格按照计划对软件的开发与维护工作进行管理。软件的整个生命周期中应该制定6类计划，即项目概要计划、阶段计划、项目控制计划、产品控制计划、测试计划和维护计划。当然，该原理在具体执行过程中也有其他的版本，应根据项目的特点灵活变化。

（2）应坚持进行阶段评审

在软件开发过程中，错误的发现和改正越迟，所付出的代价就越高。因此，在每个开发阶段都应进行严格的评审，尽早发现在软件开发过程中所犯的错误是一条必须遵循的重要原则。书面文档是阶段性活动的直观成果，因此，强调文档管理是必要的。

（3）对开发过程中的需求变动实行严格的控制

一方面在软件开发过程中不应随意改变需求，因为改变需求意味着计划的修正，这往往需要付出较大的代价；另一方面客户提出改变需求的要求也应是允许的，但这就需要进行严格的约束。也就是说，当改变需求时，为了保持软件各个配置成分的一致性，必须实行严格的产品控制，其中主要是实行基准配置管理。基准配置是经过阶段评审后的软件配置成分。一切有关修改软件的建议，特别是涉及对基准配置的修改建议，都必须按照严格的规程进行评审，获得批准以后才能实施。

（4）采用现代程序设计技术

采用先进的程序设计技术不仅可以提高软件开发和维护的效率，而且可以提高软件产品的质量。自从提出软件工程的概念以来，人们一直致力于研究各种新的程序设计技术，已被广泛采用的程序设计技术有结构设计技术、面向对象技术等。

（5）结果应能清楚地审查

由于软件本身的特点，使得软件产品的开发过程比一般产品的开发过程更难于评价和管理。为了提高软件开发过程的精细管理水平，应该根据软件开发项目的总目标及完成期限，在给定成本和目标进度的前提下，规定开发组织的责任和产品标准，从而使得所得到的结果能够清楚地审查。

（6）开发小组的成员应该少而精

首先，根据所谓的“二八定律”所述：20%的人，解决了软件中80%的问题。其次，随着开发小组人数的增加，因交流讨论而造成的通信开销也急剧增加，同时，通信路径交叉过多，更易产生错误。也就是说软件开发小组的组成人员应具备相应的素质，而人数不宜多。

（7）承认不断改进软件工程实践的必要性

为了保证软件开发与维护的过程能跟上时代前进的步伐，不仅要积极主动地采纳新的软件技术，而且要注意不断总结经验，以便今后的软件开发者借鉴。该原理保证了软件工程技术的先进性。

在软件系统的具体开发过程，还应遵循以下四条指导性原则：

（1）项目负责人应对软件开发的可行性负责，要明智地对项目进行取舍。不能完成的项目或没有把握的项目往往会给项目组带来极大的麻烦。

（2）应以用户的需求作为开发基础，不能将开发者的意志强加给用户。否则，后期会产生不必要的矛盾。

（3）软件系统的开发应得到用户高层管理者的授权，并以双方认可的形式明确授权。不然软件开发的成果可能会被否定。

（4）开发过程是一个不断完善，往复进行的非线性过程，应有相应的工作机制适应这一规律。

为了开发出低成本高质量的软件产品，软件开发组织还应遵循以下软件开发方法学原则：

（1）抽象原则

抽取事物最基本的特性和行为，忽略非基本细节。采用层次抽象，自顶向下、逐步细化的方法控制软件开发过程的复杂性。

（2）信息隐藏原则

将模块中对外的部分设计成清晰的接口，而实现细节隐藏在模块内部，不让模块的使用者直接访问，也就是所谓的信息封装。使用者只能通过模块接口访问模块中封装的数据。

（3）模块化原则

模块是程序中在逻辑上相对独立的功能集成体，它应有良好的接口定义，模块之间或模块与操作者间通过设定的接口进行联系。模块化有助于信息隐藏和抽象，有助于表示功能复杂的系统。

（4）局部化原则

在一个模块内集中逻辑上相互关联的计算机资源，并保证模块之间松散的耦合，而模块内部有较强的内聚，这就是计算机资源的局部化，局部化有助于控制软件结构的复杂化趋势。

（5）确定性原则

软件开发过程中所有概念的表达都应是确定的，无歧义的，规范的。这样有助于人们在交流时不会产生误解、遗漏，保证维护开发工作的协调一致。

（6）一致性原则

整个软件系统的各个模块应使用一致的概念、符号和术语，程序内、外部接口应保持一致，软件同硬件、操作系统的接口应保持一致，系统规格说明与系统行为应保持一致，用于形式化规格说明的公理系统应保持一致。

（7）完备性原则

软件系统不丢失任何重要成分，可以完全实现系统所要求的功能。为了保证系统的完备性，在软件开发和运行过程中需要严格的技术评审。

（8）可验证性原则

开发大型的软件系统时需要对系统自顶向下、逐层分解。系统分解应遵循使系统易于检查、测试、评审的原则，以确保系统的正确性。

实际的软件开发过程所面临的问题是多样的、复杂的。只有在以上各原理、指导性原则和方法学原则的指导下，软件工程的目标才能得以实现。

1.2.3　软件生命周期模型简介

引入软件生命周期的概念对软件开发过程的管理意义重大，可使软件生产有相应的模式、流程、工序和步骤可遵循。软件生命周期是指一个软件系统从目标提出到最后淘汰的整个过程。软件工程基本原理强调软件生命周期的阶段性，其基本思想是各阶段任务相对独立，具有明确的完成标志。同一阶段各项任务的性质要尽可能相同。阶段的划分使每个阶段任务的复杂度降低，简化了不同阶段之间的联系，以利于软件开发工程的组织管理。在实际的软件开发过程中，多数场合不能一次就全部精确地生成需求规格说明，故软件生命周期的各个阶段也不一定是严格按顺序进行的，根据需求的精化会出现回溯或超越的情况。原则上，前一阶段任务的完成是后一阶段工作的前提和基础，而后一阶段的任务则是对于前一阶段问题求解方法的具体化。目前划分软件生命周期阶段的方法有多种，软件的规模、种类、用途、开发方式、开发环境以及使用方法都会影响到软件生命周期阶段的划分。一般来说，整个软件生命周期可以划分为三个阶段，即软件定义、软件实现和运行维护。各阶段的划分如图1-1所示。

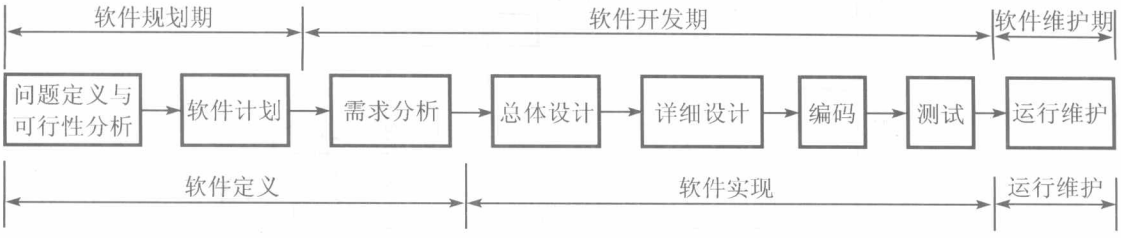

图 1-1　软件生命周期

软件定义期的任务是确立软件开发项目的总目标，分析工程的可行性，确定为达到项目目标所应该采取的策略及系统必须完成的功能，估计完成该工程需要的资源和成本，并制定工程进度。软件实现期具体设计和实现在前一时期所定义的软件，并进行必要的测试，以确保所开发软件的质量。运行维护期的主要任务是使软件持久地满足用户的需要，延缓软件的衰退。

软件生命周期模型是描述软件开发过程中各种活动如何安排的模型。它为软件开发提供了有力的支持，为软件开发过程中所有活动提供了统一的纲领，为参与软件开发的所有成员提供了帮助与指导。它展示了软件的演绎过程，是软件生命周期模型化技术的基础，也是建立软件开发环境的核心。

目前有许多种软件生命周期模型，常见的如瀑布模型、原型模型、螺旋模型、增量模型、喷泉模型、智能模型等。这些模型各有各的特点，软件开发人员要根据需要进行选择。现分别介绍各主要软件生命周期模型。

1. 瀑布模型

瀑布模型（waterfall model）也称为软件生命周期模型，它是由W. Rogce于1970年提出来的。它将软件生命周期的各项活动规定为依照固定顺序连接的若干阶段工作，并规定每个阶段应完成的任务。把生命周期划分为小的阶段是为了控制软件开发工作的复杂性，其目的是通过确定的步骤将用户需求从抽象的概念渐变为具体的软件系统。各阶段包括问题定义与可行性分析、软件计划、需求分析、总体设计、详细设计、编码、测试和运行维护。在瀑布模型中，各相应阶段被规定为由前至后、相互衔接的固定次序，故上一阶段对其以下阶段有影响。为了及时发现开发中的错误，避免影响扩散，瀑布模型要求每一阶段工作完成后必须对该阶段的结果进行评审，通过后才能进入下一阶段。就如同瀑布流水，逐级下落，该模型之名由此而来。20世纪80年代以前，瀑布模型一直是唯一广泛采用的生命周期模型。软件项目需要具备一定的条件才能较好地使用瀑布模型，如需求具有一定的稳定性、分析人员对应用领域较熟悉、项目风险较低、用户很少干涉开发过程等，而在实际的开发中，同时满足这些条件的情况往往并不普遍。所以，开发人员往往要对瀑布模型进行一定的改良。瀑布模型的表示如图1-2所示。

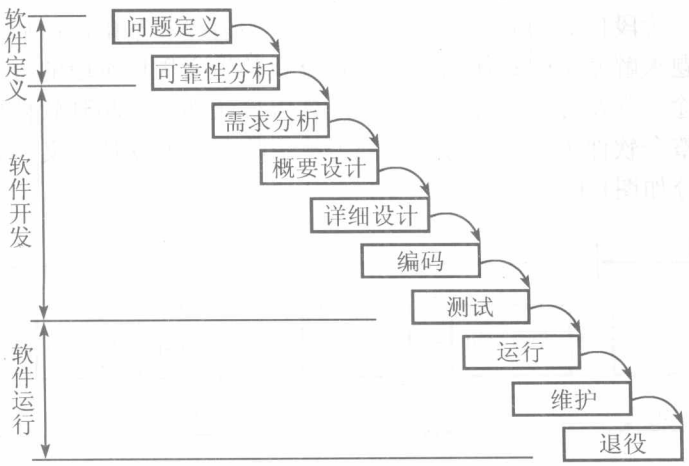

图 1-2　瀑布模型