高等学校计算机基础教育系列教材

# 软件工程
## 原理、方法与应用

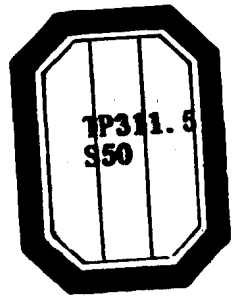
●史济民 主编

高等教育出版社

高等学校计算机基础教育系列教材

# 软件工程
# 原理、方法与应用

史济民　主编


高等教育出版社

(京)112号

## 内 容 简 介

软件工程是继程序设计课之后，对提高学生软件开发能力有重要作用的一门课程。本书以工程化的软件开发技术为主体，从应用出发，力求实用，同时介绍有关软件工程管理和软件工具与环境的基础知识。全书共十四章，其内容依次为：一、软件危机与软件工程；二、软件生存周期；三、软件需求分析；四、软件设计概述；五、结构化系统设计；六、详细设计；七、其它设计方法；八、编码和语言选择；九、软件测试；十、软件维护；十一、软件项目计划；十二、软件工程管理；十三、软件质量保证；十四、软件工程环境。

本书可作为高等学校软件工程课的教材，也可供在职的软件开发人员自学参考。全书由基本部份和扩充部份组成。前者适用于非计算机专业，凡学过一门程序设计语言的读者均可读懂，加上扩充部份后，可供计算机专业本科生用作教材。

责任编辑：程 胜

JS199/18

**图书在版编目(CIP)数据**

软件工程：原理、方法与应用/史济民主编. —北京：高等教育出版社，1999 重印
ISBN 7-04-002903-0

Ⅰ.软… Ⅱ.史… 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(95)第 20517 号

| | | | |
|---|---|---|---|
| 出版发行 | 高等教育出版社 | | |
| 社　　址 | 北京市东城区沙滩后街 55 号 | 邮政编码 | 100009 |
| 电　　话 | 010—64054588 | 传　　真 | 010—64014048 |
| 网　　址 | http://www.hep.edu.cn | | |
| 经　　销 | 新华书店北京发行所 | | |
| 排　　版 | 高等教育出版社照排中心 | | |
| 印　　刷 | 国防工业出版社印刷厂 | | |
| 开　　本 | 787×1092　1/16 | 版　　次 | 1990 年 5 月第 1 版 |
| 印　　张 | 14.75 | 印　　次 | 1999 年 4 月第 10 次印刷 |
| 字　　数 | 345 000 | 定　　价 | 12.20 元 |

凡购买高等教育出版社图书，如有缺页、倒页、脱页等质量问题，请在所购图书销售部门联系调换。

# 出 版 说 明

在计算机科学技术迅速推广应用的今天，在高等学校中向各个专业的学生普遍进行计算机的教育，使每个学生具有必要的计算机知识和应用计算机的能力，是高等教育的一项重要任务。

近年来，全国许多高等学校的理工、农医、经济管理等专业，相继设置了计算机的课程，非计算机专业中的计算机基础教育发展很快。非计算机专业的学生占全体大学生的95%以上，在这个领域中开展计算机教育的情况，在相当大的程度上决定了我国新一代知识分子应用计算机的水平。现在，人们已经认识到，计算机的知识和应用计算机的能力是当代知识分子知识结构中不可缺少的一个重要部分。

全国高等院校计算机基础教育研究会，在总结了许多学校进行计算机基础教育的经验的基础上，1985年提出了在非计算机专业中按四个层次开展计算机教育的方案设想（即：计算机应用入门和程序设计—微型机的原理与应用—计算机软件技术基础—结合各专业的计算机专门课程），得到各校的赞同，许多学校已按此规划设置计算机课程。

在非计算机专业中开展计算机教育，其要求内容、教材体系和教学方法等都和计算机专业有很大的不同，不能照搬计算机专业的做法。必须根据非计算机专业的特点和规律组织教学。课程设置应该以应用为出发点，以应用为目的，教材编写应该考虑到学生的基础，将科学性、实用性和通俗性结合起来，使之容易被接受。

为了向全国各类学校和专业提供一套适用的教材，我们决定组织编辑出版“高等学校计算机基础教育系列教材”，由高等教育出版社出版。系列教材包括四个层次中的课程所相应的教材。

本系列教材具有以下特色：内容新颖、实用性强、概念清晰、通俗易懂、层次配套。本系列教材的适用对象是：高等学校中非计算机专业、计算机应用专业、中专计算机专业、计算机培训班，以及计算机的自学者。也可供计算机其它有关专业选用。多数教材的写法便于自学。计算机的初学者可以根据需要按照四个层次循序渐进地学习各门课程，以获得所需的知识。

由于计算机科学技术的迅速发展，本系列教材的书目和各书的内容也会不断更新。期望全国各校专家和广大读者对本系列教材的内容和编写方法提出意见，以便不断改进，以满足全国广大读者的要求。

为了编辑好这套系列教材，特成立了由全国高等学校中具有丰富教学经验的专家组成的编辑委员会，负责制订规划、组织稿件、择优出版。

全国高等院校计算机基础教育研究会

1988.11

# 高等学校计算机基础教育系列教材
# 编辑委员会

# 前　言

本书系根据编者在大学讲授软件工程课的讲义修订而成，由成都科技大学和华东化工学院两校的有关教师合作编写。

对于软件工程，不少读者久闻其名、但又不知其详。有人认为，它是用于开发大型软件的技术，对开发小规模软件没有多少用处；不少人相信学习软件工程有用，但担心自己的时间少、基础差、迟迟不敢问津；更多的读者顾虑找不到实用的、适合自己程度的教材，生怕花了功夫学不懂，白耽误宝贵的时间。

软件工程的范围很宽，可以覆盖软件开发技术、软件工程环境、软件经济学、软件心理学，以及软件工程管理等多方面的内容。就软件开发技术而言，又可区分为形式化方法与非形式化方法两大分支。前者以形式化的程序变换和验证技术为主要内容，多流行于学术界；后者旨在用工程方法生产出高质量、可维护的软件产品，多流行于工业界。从我们见到的国内外部分教材来看，选材和重点多有不同，相互间差异颇大。我们以为，除了计算机专业的研究生以外，所有大学本科学生、非计算机专业的研究生和各类继续教育学员的软件工程课，都应该立足于应用，把重点放在非形式化的软件开发技术上，同时讲一点有关管理的知识，讲一点软件工具与环境。要使学生（员）在原来掌握某种语言程序设计技术的基础上，比较系统地了解软件工程的原理、方法与技术，并能直接运用这些知识来指导软件的开发工作。为此，我们确定了编写本书的指导思想：

一、目标：从应用出发，力求实用，并便于自学；

二、范围：以工程化的软件开发技术为主，兼顾环境与管理；

三、起点：只要学过一门程序设计语言的人都能学懂。

编者在大学及各类训练班讲授软件工程课多年。在教学实践中深深感到，无论大学生或专业人员，有没有学习过软件工程，学习的内容是否适当，对他们的软件开发能力与开发效果有明显的影响。为使初学者和从事软件工作的专业人员均能从本书获益，我们把内容分为基本部分和扩充部分，并在扩充部分的章节前冠以“*”号。初学者和非计算机专业的学生可只学习基本部分，扩充部分则可供计算机专业本科生和其它需要深入了解的读者学习。

本书由史济民主编，共含十四章。其中第六、八两章由李昌武执笔，第九、十两章由乔沛荣执笔，其余各章均由史济民执笔。宋国新参加了编写本书大纲的讨论，李建平阅读校对了全部书稿，他们都对初稿提出了有益的意见。北京航空航天大学计算机系麦中凡副教授审阅了全部书稿，并对全书的写作提出了指导性的意见。借此机会。编者谨向他们表示诚挚的感谢！

把《软件工程》教材从计算机专业推广到非计算机专业，是一种新尝试。编者衷心感谢全国高等学校计算机基础教育研究会对这一工作的支持和鼓励。限于我们的水平与经验，书中阙错难免，诚恳希望读者不吝指正。

编者

1989年1月

# 目　录

# 第一章　软件危机与软件工程

## 1.1　软 件 危 机

随着计算机应用的逐步扩大，软件需求量迅速增加，规模也日益增长。长达数万行、数十万行乃至百万行以上的软件，已不鲜见。美国阿波罗登月计划的软件长 1,000 万代码行，航天飞机软件长达 4,000 万行，就是两个突出的例子。

软件规模的增长，带来了它的复杂度的增加。如果说编写一个数十到数百行的程序连初学者也不难完成，则开发一个数万以至数百万行的软件，其复杂度将大大上升，即使是富有经验的程序员，也难免顾此失彼。其结果是，大型软件的开发费用经常超出预算，完成时间也常常脱期。尤其糟糕的是，软件可靠性往往随规模的增长而下降，质量保证也越来越困难。

众所周知，任何计算机系统均由硬件、软件两部分组成。在计算机应用早期，软件仅包含少量规模不大的程序，应用部门花费在软件上的投资（成本)仅占很小的份额。随着应用面的不断扩大，软件的花费越来越大，所占的百分比也越来越高。B.Boehm 在 1973 年发表的一篇文章中曾预计，到 1985 年，美国空军的软件费用将上升到计算机总费用的 90%（参阅图 1.1)。即在每 100 元用于计算机的投资总额中，软件将花费 90 元。

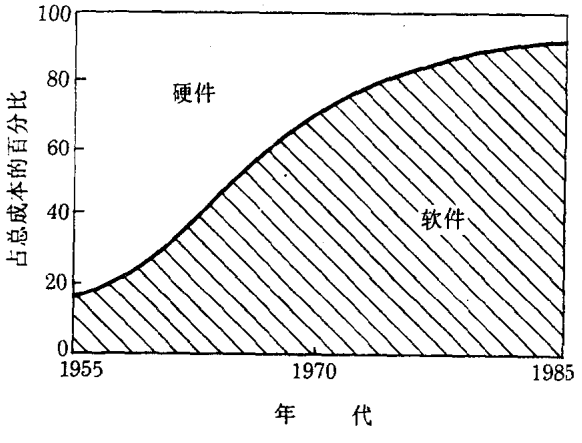



图 1.1　硬件/软件成本变化趋势

庞大的软件费用，加上软件质量的下降，对计算机应用的继续扩大构成巨大的威胁。面对这种严峻的形势，软件界的有识之士发出了软件危机的警告。

从以下两个方面，可以解释软件费用的急剧上升原因所在：

1. 与硬件需要维护一样，软件也需要维护。实践表明，软件费用不仅花费在开发上，尤其要花费在维护上。

一个大型软件，即使在开发时经过严格的测试与纠错，也不能保证运行中不再出现错误。维护的第一件事，就是纠正软件中遗留的错误（纠错性维护)。不仅如此，在软件的生存周期中，还常常要为完善功能、适应环境变更等原因对它修改，即进行所谓“完善性维护”与“适应性维护”（详见第 10.1 节)。不言而喻，软件的规模愈大，以上各种维护的成本必然愈高。

维护既耗费财力，也耗费人力。为了维护，要占用计算机厂家或软件公司大批软件人员，使他们不能参加新软件的开发。难怪有些文献把维护比作冰海中横在前进航道上的冰山，或直称之为维护墙（maintenance wall），视之为软件生产中难以逾越的障碍。

2. 生产技术的落后，是软件成本持续上升的又一个重要原因。

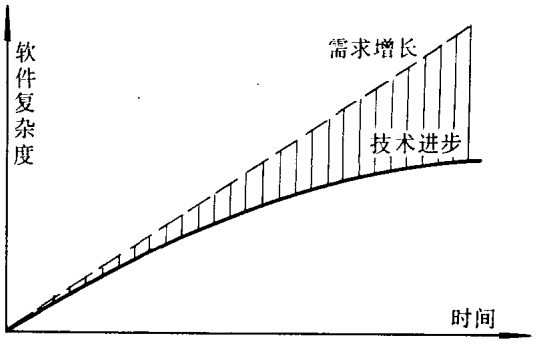


图 1.2　软件产业危机的增长

有人统计，硬件的性能价格比在近 30 年中增长了 $10^6$。一种新器件的出现，其性能较旧器件提高；价格反有所下降，这就是微电子技术创造的奇迹。软件则相形见绌，从几千行到几千万行，虽然规模与复杂度增长了几个数量级，但生产方式长期未突破手工业的方式，生产力提高得十分缓慢。一个在突飞猛进，另一个改善不多，更突出了软件生产技术跟不上需求的矛盾，如图 1.2 所示。

## 1.2　软件工程学的范畴

“软件工程”一词，是 1968 年北大西洋公约组织（NATO）在联邦德国召开的一次会议上首次提出的。它反映了软件人员认识到软件危机的出现以及为谋求解决这一危机的一种努力。

人们曾从不同的角度，给软件工程下过各种定义。但是不论有多少种说法，它的中心思想，是把软件当作一种工业产品，要求“采用工程化的原理与方法对软件进行计划、开发和维护”。这样做的目的，不仅是为了实现按预期的进度和经费完成软件生产计划，也是为了提高软件的生产率与可靠性。

近 20 年来，人们围绕着实现软件优质高产这个目标，从技术到管理做了大量的努力，逐渐形成了“软件工程学”这一新学科。图 1.3 列举了它所包含的主要内容。

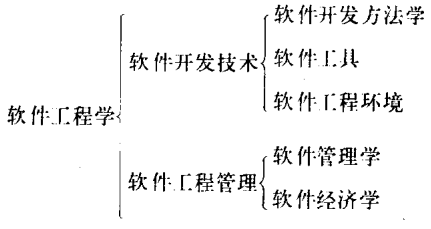


图 1.3　软件工程学的范畴

**1. 软件的演变**

软件的发展，大体上经历了程序、软件和软件产品等 3 个阶段。

早期的程序规模较小。随着系统程序的增加，人们把程序区分为系统程序和应用程序，并将前者称为软件。但是，无论是软件或程序，在开发过程中都很少考虑到它们的维护。只是当软件工程兴起后，人们才把软件视为产品，强调软件的“可维护性”，同时确定了各个开发阶段

必需完成的“文档”（documents）。“软件（产品）＝程序+文档”，已成为软件专业人员熟知的公式。缺乏文档仅由程序组成的清单，不能够称为软件产品。

**2. 软件开发方法学的形成**

与上述三个发展阶段相对应，软件开发方法也发生了巨大变化。早期的程序设计基本上属于个人活动性质，程序员各行其是，并无统一的方法可循。60年代后期兴起的结构程序设计，使人们认识到采用结构化的方法来编写程序，不仅可以改善程序的清晰度，而且也能提高软件的可靠性与生产率。随后，人们又认识到编写程序仅是软件开发过程中的一个环节，有效的开发应该包括“需求分析”—“软件设计”—“编码”等数个阶段。把结构化的思想扩展到分析阶段和设计阶段，于是又形成了著名的“结构分析”与“结构化设计”等软件开发技术。与此同时，也出现了一些从不同基点出发的软件开发方法，如Jackson方法、LCP方法等。尽管这些方法的具体内容各有不同，但它们都遵循某些共同或类似的原则，都对软件开发步骤和文档格式提出了规范化的要求。软件生产已经摆脱了过去随心所欲的“个人技艺”的状态，进入了有章可循的、向结构化和标准化迈进的“工程化”阶段。

**3. 软件工具**

“工欲善其事，必先利其器”，人类早就认识到工具在生产过程中的重要作用。伴随着软件开发方法的发展，也研制出了众多“帮助开发软件的软件”，人们称之为软件工具（software tools）。它们对提高软件生产率，促进软件生产的自动化都有重要的作用。

试设想我们要在微机上用PASCAL语言开发一个应用软件。首先要在编辑程序支持下把源程序输入计算机。然后，调用PASCAL的编译程序，把源程序翻译成目标程序。如果发现错误，就重新调入编辑程序对源程序修改。编译通过后，应调用连接程序把所有通过了编译的目标程序连同与之有关的库程序连接起来，构成一个能在计算机上运行的可执行软件。在这里，编译程序、编辑程序、连接程序以及支持它们的计算机操作系统，都属于软件工具。离开了这些工具，软件开发就失去了支持，会变得十分困难和低效，甚至不能进行。

以上提到的，仅是在编码阶段常用的一些软件工具。在其余的开发阶段，例如分析阶段、设计阶段和测试阶段，也研制了许多有效的工具，以后各章将陆续介绍。众多的工具组合起来，配套成龙，可以组成“工具箱（tool box）”或“集成工具（integrated tool）”，供程序员在不同的阶段按需选用。

**4. 软件工程环境**

工具和方法，是软件开发技术的两大支柱，它们密切相关。当一种方法提出并证明有效后，往往随之研制出相应的工具，来帮助实现和推行这种方法。新方法在推行初期，总有人不愿接受和采用。若将新方法融合于工具之中，使人们通过使用工具来了解新方法，就能更快促进新方法的推广。

方法与工具相结合，再加上配套的软、硬件支持，就形成环境。创造适用的软件工程环境（software engineering environment，简称$SE^2$），一直是软件工程研究中的热门课题。

为了说明软件开发对环境的依赖，不妨回顾一下分时系统所产生的影响。在批处理时代，用户开发的程序是分批送入计算中心的计算机的，有了错误，就得下机修改。程序员对自己写的程序只能断续地跟踪，思路经常被迫中断，效率难于提高。分时系统的使用，使开发人员从此能在自己的终端上跟踪程序的开发，仅此一点，就明显提高了开发的效率。近十几年来出现

的 UNIX 环境，Ada 环境，以及形式繁多的工作站，不仅反映了人们创造良好软件环境的努力，也把对软件工程环境的研究提到一个新的高度。

**5. 软件工程管理**

在工业生产中，即使有先进的设备与技术，管理不善的企业也不能获得良好的经济效益。对于软件生产，不能按质按时完成计划，其中管理混乱往往是一条重要原因。可惜的是，至今软件管理尚未获得普遍的重视。

软件工程管理的目的，是为了按照进度及预算完成软件计划，实现预期的经济和社会效益。它包括成本估算、进度安排、人员组织、质量保证等多方面的内容，涉及管理学、经济学等多项学科。

显然，软件管理也可以借助计算机来实现。现在帮助管理人员估算成本、制订进度、生成报告等工具已研制出来。一个理想的软件工程环境，应该同时具备支持开发和支持管理两个方面的工具。

## 1.3 软件工程的应用

以上简单介绍了组成软件工程学的主要成份——软件开发方法学、软件工具、软件工程环境、以及软件工程管理的内容和作用。它们包含了广大软件人员行之有效的好方法与好经验。这是他们长期实践活动的科学归纳与总结，也包含为了实现更高的软件可靠性、更高的软件生产自动化程度所进行的一系列理论课题的探索与研究。

软件开发在技术和管理两个方面的复杂程度，均与软件的规模密切相关。越是规模大的软件，越要在开发中严格遵守软件工程的原则和方法。表 1.1 列出了软件规模的分类。以下将按照这一分类，分别说明软件工程方法在各类软件开发中的应用。

**表 1.1 软件规模分类表**

| 分类 | 程序规模 | 子程序数 | 开发时间 | 开发人数 |
|---|---|---|---|---|
| 极小 | 500 行以下 | 10～20 | 1～4 周 | 1 人 |
| 小 | 1K～2K 行 | 25～50 | 1～6 月 | 1 人 |
| 中 | 5K～50K 行 | 250～1,000 | 1～2 年 | 2～5 人 |
| 大 | 50K～100K 行 | | 2～3 年 | 5～20 人 |
| 甚大 | 1M 行 | | 4～5 年 | 100～1,000 人 |
| 极大 | 1M～10M 行 | | 5～10 年 | 2,000～5,000 人 |

**1. 中小型程序**

极小程序大都为个人软件，由个人开发和使用，且常常只用几个月就废弃了。这类程序一般不需要正式的分析和详细的设计文档，也用不着范围广泛的测试计划。但即使是这类极小的程序，如能在开发中作一点分析和系统设计，遵守结构化编码和合乎规范的测试方法，对提高程序质量和减少返工，仍有不小的帮助。

小程序包括工程师们用于解数值问题的科学计算程序，数据处理人员生成报表或完成数据操作所用的小型商业应用程序，以及大学生们在编译原理或操作系统等课程设计中编写的程序。这类程序的长度一般不超过 2K 行，与其它程序也没有什么联系。开发者通常仅有一人，无需或很少需要和用户或其它程序员打交道。在开发这类程序时，应贯彻软件工程中的技术标准和表示法（notations），按标准编写文档，并系统地进行项目复审。当然，上述工作的正规程度不必象开发大程序时那样严格。

中规模程序包括汇编程序、编译程序、小型管理信息系统、仓库系统以及用于过程控制的一些应用程序。这类程序可能与其它程序有少量联系，也可能没有。但是在开发过程中，程序员与用户间或程序员之间均存在一定的联系。所以在制订软件计划、编制文档、进行阶段复审等方面，正规化的要求都比较高。在开发中如能系统地应用软件工程的原理，对改进软件质量，提高程序员生产率和满足用户需求，都将有很大好处。

有人以为，软件工程适用于开发大型软件，对开发规模小的软件没有多少用处。这其实是误解。许多系统软件和大多数应用软件都属于中、小型软件。如上所说，它们的开发都需要软件工程作指导，更不用说开发大型以上的程序了。

**2. 大型程序**

大型编译程序、小型分时系统、数据库软件包、以及某些图形软件和实时控制系统，都是大型软件的实例。它们的长度可达 5—10 万行，且常与别的程序或软件系统有种种联系。开发人员通常由几个程序员小组（例如 3 个小组、每组 5 人）组成，在组与组间、组中不同成员间、程序员同管理人员及用户之间，都存在着大量的通信。在长达 2 至 3 年的开发过程中，中途调走或增加部分开发人员，也是难以避免的。

长达百万行的软件称为甚大型软件，常见于实时处理、远程通信和多任务处理等应用领域。例如大型的操作系统和数据库系统，军事部门的指挥与控制系统，等等。IBM / 360 系列的操作系统，就是拥有 100 万行源程序的甚大型软件，开发历时 5 年，参加开发的程序员先后达 5 千人之多。有人预测，到 1990 年，数据处理领域典型操作系统的规模可能增长到 400 万行，通常这类软件都包含若干个重要子系统，每一子系统均构成一个大型软件。

极大型软件一般由数个甚大型的子系统构成，常含有实时处理、远程通信、多任务处理、以及分布处理等软件，实例有空中交通管制系统，洲际导弹防御系统，军事指挥和控制系统等。它们的源代码可长达数百万以至数千万行，开发周期长达 10 年，并要求有极高的软件可靠性。直至今日，真正开发成功的极大型软件在世界上屈指可数。就在 80 年代初，美国国防部还放弃了一项与反洲际导弹系统有关的极大型软件，原因是它超越了现代技术的能力、难以实现。

毫无疑问，所有大型以上软件的开发必须从头至尾坚持软件工程的方法，严格遵守标准文档格式和正规的复审制度。而且，由于大型软件本身的复杂性，对它们的计划和分析都很难一劳永逸。因此，管理工作十分重要，必须坚决贯彻软件工程关于管理工作的要求，才能避免或减少混乱。

由上可见，从极小程序到极大程序，软件工程都有它的用武之地，其作用不可低估。当然，软件工程也不是解决软件危机的灵丹妙药。正如美国著名软件科学家 F.Brooks 在 1987 年题为“没有银弹：论软件工程中的本质和偶然性问题”一文中所指出，人们至今尚未找到象神

话中所说的那种能够制服“狼人”（暗喻导致软件危机的本质性问题）的“银弹”。但在同一篇文章中，Brooks 也指出了提出软件工程以来的 20 年中，在软件开发方法、软件工具和环境方面所取得的令人兴奋的成就。国内外的实践一致说明，软件开发中是否采用软件工程的方法，其效果将明显不同。

本书的宗旨，就是向读者介绍软件工程中有代表性的成就，引导读者用它们的基本原理与方法来指导软件开发，并进一步激起读者研究软件生产工程化的兴趣。全书分基本部分和扩充部分两部分，强调学以致用，学用结合。这既是学习软件工程的目的，也是学好软件工程的主要方法。

## 习　题

1. 什么是软件危机？为什么会产生软件危机？
2. 什么是软件生产工程化？工程化生产方法与早期的程序设计方法主要差别在哪里？
3. 说明（1）软件方法与软件工具；（2）软件技术与软件管理的相互关系。
4. 由你的亲身实践，谈谈软件工具在软件开发过程中的作用。
5. 软件工程环境应包含哪些内容？谈谈你对环境重要性的认识。
6. 软件按规模大小可分成哪些类？怎样在各类软件的开发中应用软件工程的原理和方法？

# 第二章　软件生存周期

一切工业产品都有自己的生存周期，软件（产品）也不例外。一个软件从开始计划起，到废弃不用止，称为软件的生存周期。一般来说，软件生存周期包括计划、开发与运行三个时期，每一时期又可细分为若干更小的阶段。

生存周期是软件工程的一个重要概念。把整个生存期划分为较小的阶段，是实现软件生产工程化的重要步骤。给每个阶段赋予确定然而有限的任务，就能够简化每一步的工作内容，使因为软件规模增长而大大增加了的软件复杂性变得较易控制和管理。

显然，生存周期的划分应该适应软件生产工程化的需要，而不应该是千篇一律的。事实上，学者们就曾提出过各种不同的划分方法，从而形成了不同的软件生存周期模型（SW life cycle model）。大体上说，这些模型可以归结为两大类：即传统的**瀑布模型**（waterfall model）和后来兴起的**原型模型**（prototype model）。本章将简要地介绍这两类不同的生存周期模型，说明它们各自的特点，使读者在深入了解各阶段的详细工作以前，对软件生存周期的全过程先有一个总体的概念。

## 2.1　生存周期的瀑布模型

瀑布模型把软件生存期划分为计划、开发、运行三个时期，每一时期又区分为若干阶段。图 2.1 是根据 B.W.Boehm 划分的七个阶段画出的瀑布模型。稍加修改，就可以得到适用于结构化开发技术的典型的瀑布模型，如图 2.2 所示。在这类瀑布模型中，各个阶段的工作顺序展开，恰如奔流不息拾级而下的瀑布，总是从上面的台阶依次流向下面的台阶。

在这里，我们把软件工程方法与程序设计方法作一简单比较。程序设计教科书中通常把程序设计归结为"建立数学模型—算法设计—编写程序"三个步骤，它们的功能虽然与上述生存周期中的"分析—设计—编码"三个阶段相似，但却反映了计算机的早期应用偏重于科学计算，程序规模又比较小的情况。随着软件规模的增长，阶段的划分越来越细。可见怎样划分生存周期的阶段，要受到软件规模、软件种类、开发方法和开发环境等多种因素的影响。在不同的课本上，对生存周期的划分也不尽一致。对初学者来说，重要的不是去研究那些细节上的差别，而是要着重理解把生存周期划分为阶段的目的与实质，即①便于控制开发工作的复杂性；和②通过一定的有限的步骤，把用户需要解决的问题从抽象的逻辑概念逐步转化为具体的物理实现。

下面以图 2.2 的生存期模型为例，简要说明各阶段的主要任务。说明中要提到一些文档的名称，其内容将在后续各章介绍，读者暂时不必深究。

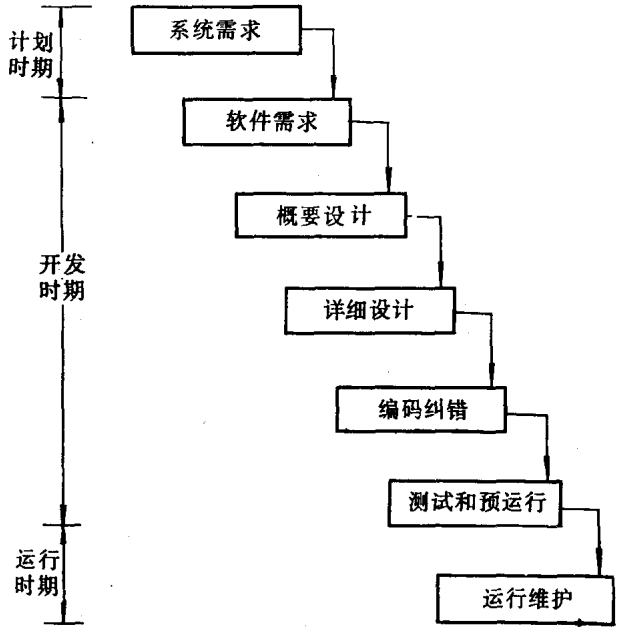



图 2.1　Boehm 建议的软件生存周期

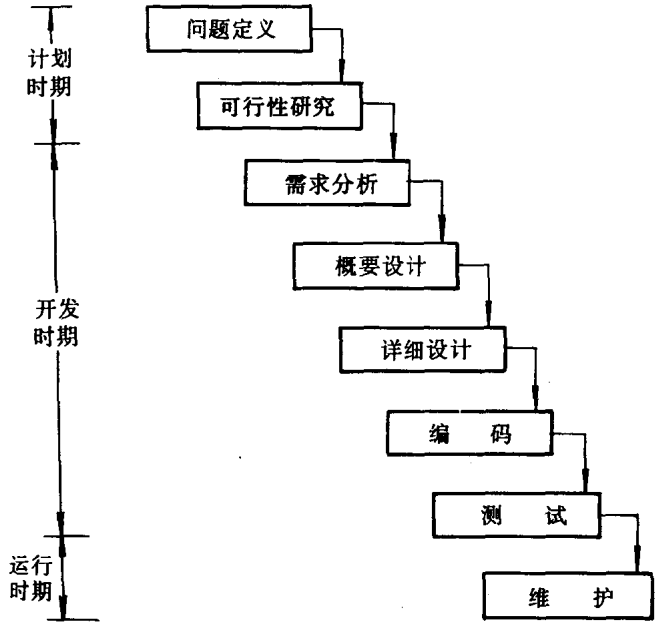



图 2.2　典型的瀑布模型

**1. 计划时期**

计划时期的主要任务是分析：分析用户要求，分析新系统的主要目标，分析开发该系统的可行性。用户和系统分析员的相互理解与配合，是搞好这一时期工作的关键。

(1) 问题定义　这是计划时期的第一步，主要弄清“用户要计算机解决什么问题”。由系统

分析员根据对问题的理解，提出关于〈系统目标与范围的说明〉(详见第 11.1 节)，请用户审查和认可。

(2) 可行性研究　目的是为前一步提出的问题寻求一种至数种在技术上可行、且在经济上有较高效益的解决方案。为此，系统分析员应在高层次上作一次大大简化了的需求分析与概要设计，并写出〈可行性论证报告〉(详见第 11.2 节)。如结论认为该项目值得进行，应接着订出〈项目实施计划〉(详见第 11.3 节)，否则便应提出终止该项目的建议。

可行性论证报告应包含关于新系统组成 (硬件与软件) 的描述。这种描述通常用〈系统流程图〉(详见第 11.2 节) 来表示。

同时，计划时期还应制订出人力、资源及进度计划。

**2. 开发时期**

开发时期要完成设计和实现两大任务。其中设计任务用需求分析、概要设计和详细设计三个阶段完成，实现任务用编码和测试两个阶段完成。把设计和实现分成两步走，目的是在开发初期让程序人员集中全力搞好软件的逻辑结构，避免过早地为实现的细节分散精力。

(1) 需求分析　其任务在于弄清用户对软件系统的全部需求，并用〈需求规格说明书〉(详见第 3.3 节) 的形式准确地表达出来。当采用结构化分析方法 (详见第 3.4 节) 时，需求规格说明书通常由数据流图、数据字典和加工说明等一整套文档组成。这些文档既是软件系统逻辑模型的描述，也是下一步进行设计的依据。

(2) 概要设计　主要任务是建立软件的总体结构，画出由模块组成的〈软件结构图〉(详见第 5.3 节) 或〈层次图〉(详见第 4.4 节)，所以又称为结构设计。

从需求规格说明书导出软件结构图，在软件开发中起着承上启下的作用，占有重要的地位。所以系统设计员应选择有经验的高级程序员担任，或直接由系统分析员兼任。

(3) 详细设计　是针对单个模块的设计。目的是确定模块内部的过程结构。这一步要求设计师为每一模块提供一个〈模块过程性描述〉，详细说明实现该模块功能的算法和数据结构，所以有时也称为算法设计 (见第 6.2 节)。

模块过程性描述通常用伪代码或 IPO 图等图形表达工具表示，由程序员承担编写。

(4) 编码　即按照选定的语言，把模块的过程性描述翻译为源程序。与"需求分析"或"设计"相比，"编码"要简单得多，所以通常由编码员 (coder) 或初级程序员担任。

请读者注意，直到这一阶段，才产生能在计算机上执行的源程序。前面各个阶段产生的，都属于软件的文档。

(5) 测试　是开发时期最后一个阶段。按照不同的层次，又可细分为单元测试、综合测试、确认测试和系统测试等步骤。测试是保证软件质量的重要手段。为确保这一工作不受干扰，大型软件的测试通常由独立的部门和人员进行。测试阶段的文档称为〈测试报告〉，包括测试计划、测试用例与测试结果等内容。

**3. 运行时期**

是软件生存周期的最后一个时期。软件人员在这一时期的工作，主要是做好软件维护。

维护的目的，是使软件在整个生存周期内保证满足用户的需求和延长软件的使用寿命。对于大型的软件，维护是不可避免的。每一次进行维护，都应该恪守规定的程序，并填写和更改好有关的文档 (详见第十章)。