21世纪 计算机基础教育系列教材

谭浩强 主编

计算机软件技术基础

■ 龚正良 等编著
■ 史济民 主审

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

浩强创作室
HAO QIANG STUDIO

21世纪计算机基础教育系列教材

谭浩强 主编

计算机软件技术基础

龚正良 许丽华 黄建华 史 令 编著

史济民 主审

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书为高校计算机基础教育第二层次的教材，是第一层次《计算机文化基础》的后续课程。本书共分7章，主要内容包括：软件工程、数据结构、操作系统、数据库技术、面向对象程序设计、计算机网络和网页设计。本书内容丰富实用，与1998年9月出版的本书第一版相比，本版新增了数据结构、网页设计两章，扩充了面向对象程序设计、软件工程两章，再加上操作系统、数据库技术、计算机网络等章，使全书内容更加丰富，并且继续保持了“强调环境与工具”，“重在应用，加强基础”等风格。

本书适用于大学非计算机专业学生作公共课教材，也可供具有高中以上文化程度、学过一种高级语言的读者自学使用。

图书在版编目(CIP)数据

计算机软件技术基础/龚正良等编著 .—2版 .—北京：电子工业出版社，2002.8
21世纪计算机基础教育系列教材
ISBN 7-5053-7694-2

Ⅰ.计… Ⅱ.龚… Ⅲ.软件—高等学校—教材 Ⅳ.TP31

中国版本图书馆CIP数据核字(2002)第041142号

责任编辑：应月燕
印　　刷：北京四季青印刷厂
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：20.25　字数：518千字
版　　次：2002年8月第2版　2002年8月第1次印刷
印　　数：8000册　定价：26.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。
联系电话：(010)68279077

《21 世纪计算机基础教育系列教材》序

21 世纪是信息时代，是科学技术高速发展的时代。计算机技术与网络技术的结合，使人类的生产方式、生活方式和思维方式发生了深刻的变化。在新世纪中，计算机知识已成为当代人类文化的一个重要组成部分。我们要将计算机知识和应用向一切有文化的人普及。

高等学校承担着为社会培养高层次人才的任务，大学生毕业后应当成为我国各个领域中的计算机应用人才，成为向全社会推广计算机应用的积极分子。在大学里应当把计算机教育放在十分重要的位置。

我国高校的计算机基础教育起步于 20 世纪 80 年代初。20 年来从无到有迅速地发展，从理工科专业发展到所有专业，从最初只开设一门语言课到按三个层次设置课程，学时也从三四十小时增加到一二百小时。计算机基础教育已经先后上了几个台阶，现在又需要上一个新的台阶。在新世纪初，我们要求进一步提高大学生应用计算机的能力，以适应科学技术和经济发展的需要。

我们在这里所说的计算机基础教育，是指面对全体大学生的计算机教育；而非计算机专业和计算机专业中的计算机教育的特点则有很大的区别。无论学生的基础、培养目标、教学要求、教学内容、教学方法和教材，都和计算机专业有很大的不同。绝不可简单地照搬计算机专业的模式，否则必事倍功半。计算机基础教育实际上是计算机应用的教育，应当以应用为目的，以应用为出发点。

计算机不仅是一种工具，也是一种文化，工具是可选的，文化却是必备的。对学生来说，它还是全面素质教育的一个重要部分，通过学习计算机知识能激发学生对先进科学技术的向往，启发学生对新知识的学习热情，培养学生的创新意识，提高学生的自学能力，锻炼学生动手实践的能力。多年来的实践证明，对计算机感兴趣的学生，绝大多数都是兴趣广泛、思想活跃、善于思考、自学能力较强、喜欢动手实践的。他们绝不是只会死背书本的书呆子。

我们必须认真分析非计算机专业的特点，根据教学上的需要与可能，制定出恰当的教学要求，使学生在有限的时间内能学到最多的有用的知识。全国高等院校计算机基础教育研究会曾提出了在计算机基础教育中应当正确处理的 10 个关系，即：(1) 理论与应用的关系，(2) 深度与广度的关系，(3) 当前与发展的关系，(4) 硬件与软件的关系，(5) 追踪先进水平与教学相对稳定的关系，(6) 课内与课外的关系，(7) 课程设置与统一考试的关系，(8) 计算机课程与其他课程的关系，(9) 要求学生动手能力强与当前设备不足的矛盾，(10) 计算机技术发展迅速与师资现状的矛盾。在教学实践中，许多学校都创造了丰富的经验。

在非计算机专业的教学中，首先需要解决的问题是：准确定位，合理取舍教学内容。我们必须分清楚：哪些内容是需要的，哪些内容是不需要的；哪些内容是目前暂时可以不学而留待以后学的，哪些内容是目前不必学而以后也不必学的；哪些内容是主要的，哪些是次要的。绝不可眉毛胡子一把抓，不加分析、不问主次，使学生感到难以入门。

在教学方法和教材的编写上，要善于用通俗易懂的方法和语言说明复杂难懂的概念。传统的教学三部曲是：提出概念—解释概念—举例说明。我在多年教学实践中对于计算机应用课程总结了新的三部曲：提出问题—介绍解决问题的方法—归纳出必要的概念和结论。从具

体到抽象，从实际到理论，从个别到一般。这是符合人们的认识规律的。实践证明，这样做已取得了很好的效果。

为了推动高校的计算机基础教育，我在1996年主编了《计算机教育丛书》，由电子工业出版社出版。编写这套丛书的指导思想是20个字："内容新颖、实用性强、概念清晰、通俗易懂、层次配套。"（也可简单地概括为："新颖、实用、清晰、通俗、配套"。）先后出版的近20种供大学非计算机专业使用的教材，受到高校广大师生的欢迎，几年内发行达75万册，大家认为它定位准确、程度适当、内容丰富、通俗易懂、便于自学。

在进入21世纪之际，我们根据新时期的要求，按照上面所述的指导思想，重新进行规划，对原有的教材进行了筛选，淘汰了部分内容已过时的教材，同时根据计算机技术和高校计算机基础教育的发展，组织编写了一些新教材，并对原有教材进行了修订和补充，以实现推陈出新、不断提高。

我们遴选了具有丰富教学经验的高校老师编写这套教材。在这套系列教材中，我们提供了多种课程的教材供各校选用，其中包括必修课和选修课。不同专业、不同层次的学校都可以从中选到合用的教材，我们还将根据计算机基础教育的需要不断推出新的教材。

本系列教材是由浩强创作室策划、组织和编写的。参加工作的有：谭浩强、薛淑斌、史济民、吴功宜、边奠英、徐士良、赵鸿德、李盘林、孟宪福、张基温、宋国新、龚正良、徐安东、毛汉书、李凤霞、许向荣、周晓玉、张玲、刘星、秦建中、王兴玲、蔡翠平、訾秀玲等。电子工业出版社对本丛书的出版给予了大力的支持，使得本丛书得以顺利出版。

由于我们的水平和经验有限，加以计算机科学技术发展很快，本丛书肯定会有不少缺点和不足，诚恳地希望专家和读者不吝指正，我们将继续努力工作，使本丛书能尽量满足广大读者的要求。

全国高等院校计算机基础教育研究会会长
《21世纪计算机基础教育系列教材》主编
谭浩强
2001年7月1日

前　言

“计算机软件技术基础”适用于“计算机文化基础”和“高级语言程序设计”的后续课程。本书作为计算机基础教学第二层次的一门主课，由软件工程、数据结构、操作系统、数据库技术、面向对象程序设计、计算机网络、网页设计等7章组成。

自本书第一版《微机软件技术基础：环境与工具》1998年问世以来，计算机基础教学有了很大的变化。为了适应教学的需要，本书除新增了“数据结构”与“网页设计”两章外，并为全书确定了以下的编写宗旨。

1. 继续以软件“环境与工具”作为主题，把全书重点放在软件操作环境与应用软件的使用上。但鉴于数据结构是计算机程序设计的重要基础，至今仍有不少学校将其列为第三层次的选修课。本版在第2章介绍了数据结构的实用知识，以兼顾等级考试（软件类）关于数据结构基本知识的需要。

2. 介绍新知识、新工具与加强基础理论相结合。例如，第1章（软件工程）增加了软件测试技术与软件重用；第3章（操作系统）既介绍资源管理，也介绍Windows操作系统的典型功能；第5章（面向对象程序设计）一方面结合VB程序设计介绍OOP的概念，一方面简要介绍了VB对数据库应用和多媒体应用的支持；第6章（计算机网络）不但概述了因特网及其应用的一般原理，也讲解了网页设计的常用语言与工具等。

本书各部分内容相对独立，自成体系，讲授时可根据教学需要酌情取舍。各章后面附有小结和习题。本书除可供高校本、专科学生作为公共课教材外，也可供需要学习软件基础知识的读者自学使用。

本书由龚正良主编，史济民主审，参加编写的还有许丽华、黄建华和史令。其中第1章、第4章和第3章的第7节由龚正良编写，第2章和第3章的其余各节由许丽华编写，第6章和第7章由黄建华编写，第5章由史令编写。全书由龚正良统稿。在本书编写过程中得到了刘江老师和李静同志的帮助，在此表示诚挚的感谢。

由于开设本课程的高校目前尚不普遍，课程内容也不尽一致，诚恳欢迎读者对本教材的内容与不足提出意见，以利于以后改进。

编　者

2002年5月于上海

目　录

第1章 软件工程

计算机硬件的飞速发展和计算机应用领域的急剧扩大，对计算机软件的需求也随之猛增，同时对软件的规模和可靠性提出了越来越高的要求。现代软件应该是一个可维护的产品，不仅要写出高质量的程序，还需要编制一整套文档(Documents)，供开发、应用和维护程序者使用。因此，软件也应当像其他产业的产品一样，以系统的、工程的方法开发制作，并提供售后服务。本章在简单介绍了软件工程的形成和发展以后，重点介绍软件开发的不同方法和软件测试策略与方法，最后就软件开发环境和软件重用技术作一简要介绍。读者在阅读本章时将遇到一些新技术和新概念，建议读者不必深究。在学习以后各章后读者将会加深对本章所述概念的理解。

1.1 概述

软件工程的提出源于20世纪[①]60年代末期出现的“软件危机”，并在较短的时间内发展成一个完整的学科方向，30多年来，在理论研究和工程实践两个方面作了大量的工作。本节将通过对软件工程的形成与发展，软件工程范型的介绍，使读者对软件工程学的研究范畴有一个粗浅的了解。

1.1.1 软件工程的形成与发展

1. 软件发展的三个阶段

自从1946年第一台电子计算机诞生以来，软件开发方法从机器语言编程到软件工程方法，经历了三个阶段。

(1) 程序设计时期(1946年至60年代中期)

在这个时期将软件看做是按具体问题编制的专用程序。程序设计凭个人的经验和编程技术单独进行，使用的工具是机器语言、汇编语言、服务性程序；程序的运行和维护等工作也全部由程序设计者承担。生产方式完全是手工生产、个体劳动。这个时期只有程序的概念，没有软件的概念。

(2) 软件时期(60年代中期至70年代中期)

在这一时期，随着计算机技术的发展，计算机硬件已进入晶体管、小规模集成电路时代，计算机内存容量增大、运行速度加快、外部设备配置增加，因而急需大量的编程人员。由于程序规模增大，个体生产方式已经不能适应生产要求，需要多人分工，共同协作来编制一个程序，即采用了所谓的“作坊式”生产方式。同时，人们已开始认识到程序可以使计算机发挥巨大的灵活性，程序不再是硬件的附属。这时，已提出“软件”的概念，并且在软件开发中，开始强调软件工程师的作用，强调开发者的相互通信和协作，软件技术取得一定的进展。随着计算机应用领域

① 注：本书所述年代均为20世纪。

的不断扩大，软件的规模及结构的复杂程度也不断增加，作坊式的生产方式已难以满足软件生产的质量和数量上的要求，从而出现了所谓的“软件危机”。

(3) 软件工程时期(70 年代至今)

针对 60 年代后期使用传统的软件开发方法已不能适应大型软件的生产，难以满足软件生产的质量和数量上的要求而出现的“软件危机”，1968 年、1969 年北大西洋公约组织成员国的软件工作者召开了两个研讨会，讨论如何摆脱这一困境，同时，提出了“软件工程”这一术语，其根本目的在于克服“软件危机”中所遇到的困难问题，从此软件生产进入软件工程时代。

2. 软件危机

(1) 软件危机的主要表现

① 对软件开发成本和进度的估计常常很不准确，有时竟然出现实际费用超过预算一个数量级以上、进度推迟几个月甚至几年的情况。

② 用户往往对已完成的软件不满意。

③ 软件的质量常被怀疑。

④ 现有的软件极难维护。要在许多程序中纠正潜在的错误是很困难的，要使已有的软件适应于新的计算机或者增加用户新的需求，实际上也难以做到。

⑤ 缺乏良好的软件文档。软件文档往往与当前的软件产品不符，不能保证在软件开发过程中有合格的文档作为软件开发者相互通信的工具，不完整的文档给软件维护带来许多困难。

⑥ 软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的趋势。软件生产的低效率使人类不能充分利用现代计算机硬件提供的巨大潜力。

例如，IBM 公司 60 年代对 OS/360 操作系统的开发过程中遭受到的挫折是一个典型的例子。它由 4000 个模块组成，共约有 100 万条指令，工作量是 5000 人年(1 人年表示 1 个人工作 1 年的工作量)，开发费用达数亿美元，但人们在使用时从操作系统的程序中发现了 2000 个以上的错误，使该系统的开发未达到预期的效果，在经济上蒙受重大损失。

(2) 软件危机的产生原因

一般以为，软件危机的发生与软件产品的特征和软件产品开发与维护的方法不当有关。

其一，软件是逻辑的系统部件而不是物理的系统部件，它以程序和文档形式存在，具有无形性。因此不应根据多个实体部件的质量来衡量软件的成败，而应将软件作为一个完整的产品来衡量。软件在运行过程中不会因为使用时间长而被“用坏”，运行中如果发现错误往往是开发时期引入而在测试阶段未能发现的故障，在软件维护中要换掉有毛病的部件，不能采用简单的更换备份的方法，而只能修改原来的设计。同时，软件开发过程是一种智力活动，其开发过程的进展情况难以衡量，软件开发的质量也较难评价，因此，管理和控制软件开发过程相当困难。

其二，软件规模越来越大，功能越来越强，导致软件结构非常复杂。例如，以美国宇航局的软件系统为例，1963 年的水星计划有 200 多万条指令，1967 年的双子星座计划达到 400 多万条指令，1973 年阿波罗计划有 1000 多万条指令，1979 年的哥伦比亚航天飞机计划达到 4000 多万条指令。在大型软件开发中，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实际上是一个极端复杂的问题，它既涉及到技术和方法问题，也涉及到管理问题。早期软件开发的个体化特点导致了对软件开发和维护没有统一的标准可以遵循，由于没有注意软件开发方法和技术的研究，使软件技术的发展跟不上解决软件复杂性的要求。

(3) 解决软件危机的途径

认识软件开发中的问题和造成这些问题的原因只是解决软件危机的开始，更重要的是要充分吸取和借鉴人类长期以来从事各种工程项目所积累的行之有效的原理、概念、技术和方法，并应用于软件开发的实践中，将软件开发变成一种组织良好、管理严密、各类人员协同配合，共同完成的工程项目。

为了实现上述目标，不仅要有协调、控制、管理软件开发的理论，还应该有支持软件开发各个阶段的技术和工具。如果有标准接口完成标准功能的构件块大量存在，也将会大大改进计算机软件的设计；如果能从软件需求规格说明直接生成程序而实现自动程序生成，则显然会大大提高软件的生产效率；如果能出现真正的程序正确性证明器，软件危机就将最终消失。

由此可见，解决软件危机的根本出路在于将软件开发方法学(包括管理、控制、复审等方法)与支持软件开发各个阶段的技术和各种支持工具结合起来。

3. 软件工程

什么是软件工程？许多学者曾从不同的角度给它下过各种定义。例如，1969 年 F. Bauer 定义为："软件工程的目的，在于获得廉价的、能够在实际机器上可靠工作的软件。为此，需要建立并应用牢固的工程准则与方法。"1983 年 IEEE 定义为："软件工程是开发、运行、维护和修复软件的系统方法。"但其核心思想都是"采用工程化的原理与方法对软件进行计划、开发和维护"，以按预期的进度和经费完成软件生产计划，提高软件的生产率和可靠性。

自软件工程出现以后，人们围绕着实现软件优质高产的目标进行了大量的理论研究和实践，逐步形成了软件工程学这一新兴学科，其研究范围从单纯的软件开发方法扩展到对软件工程方法学、软件工程环境和软件工程管理等多个分支。现简述如下。

(1) 软件工程方法学

方法学是研究软件构造技术的学问。一个软件从定义、开发到维护，都需要有适当的方法。对于应用软件的开发人员，首先应该掌握在"软件需求分析"(定义阶段)、"总体设计"、"详细设计"、"编码"和"测试"(以上为开发阶段)等活动中采用的方法与技术。结构程序设计，就是指导软件详细设计和编码活动的常用技术。

早期的程序设计基本上属于个人活动。结构程序设计的兴起，使程序设计从"无法可依"变为"有章可循"。但是，人们稍后就认识到，一个完整的软件开发过程应该包括"需求分析—软件设计—编码测试"等多个阶段的活动，而结构程序设计仅是其中的部分内容。于是在一些学者的倡导下，在 70 年代中、后期又开发了多种把软件分析和设计"一气呵成"的系统开发技术，其中包括由 E. Yourdon 和 L. L. Constantine 等人倡导的"结构化分析与设计"，由 M. Jackson 倡导的"Jackson 系统开发方法(JSD)"，以及 J. D. Warnier 设计的"程序的逻辑构造"(LCP)技术等。这些不同出发点的系统开发技术，加上在软件测试中使用的"黑盒测试"与"白盒测试"等测试技术，共同构成了 70 年代到 80 年代中期软件工程方法学的主要内容。

但是，以上这些在结构程度设计基础上发展起来的开发方法都有一个先天的弱点。由于 80 年代前的大多数高级语言都是面向过程的，缺乏将数据和加于数据的操作封装在一个统一体中的机制，所以当使用传统方法来分析软件需求时，开发人员的注意力，只得或者集中于对数据的操作而忽略数据结构(例如结构化分析和设计方法)，或者集中于数据结构而忽略对数据的操作(例如 JSD 方法)。而在实际的客观世界中，每个实体的内部状态(数据)和运动(操作)总是结合在一起的。由于这一差异，就使得采用传统方法分析得出的软件模型(称为"解空

间模型"),在结构上不同于直接由客观世界的实体及实体间的联系抽象出来的模型(称为"问题空间模型"),从而增加了软件设计、修改和维护的难度。随着软件规模的增长和多媒体信息(图形、图像、声音、语言等)的应用,这一差异造成的设计复杂性,在有些软件中几乎达到传统的程序开发方法无法解决的程度。正是在这种背景下,一种更新的程序设计技术——"面向对象的程序设计"(Object-Oriented Programming,简称 OOP)才脱颖而出,并在近 10 年中取得引人注目的成就。

面向对象程序设计是许多学者工作的成果。D. Parnas,R. Abbott 和 G. Booch 都是这一方法的早期倡导者。在这一方法中,数据和对数据的操作被封装在一个个称为"对象"(Object)的统一体中,对象之间则通过"消息"(Message)相互联系,从而使由软件所描述的系统与客观实际在结构上十分相似,不仅提高了软件的可修改性与可维护性,同时也提高了软件的可重用性,这些都是软件工程多年来所追求的目标。从结构程序设计到面向对象程序设计,是程序设计方法的又一次飞跃。所以有人认为,它对软件工程当前面临的困境是一个有希望的突破口。

(2) 软件工程环境

环境一词,对不同用户往往有不同的含义。对最终用户(End User)而言,环境就是他们运行程序所使用的计算机系统。这类用户对环境的要求,主要是运行可靠,操作方便,容易学习和使用。对于应用软件开发人员,环境是他们进行开发活动的舞台。在软件工程时代,环境应支持开发人员按照软件工程学的方法,全面完成软件生存期中各个阶段、首先是开发阶段的任务。所以软件工程环境(Software Engineering Environment,简称 SEE)有时也称为软件开发环境(Software Development Environment,简称 SDE)。

软件工具是环境中最活跃的成分。所谓工具,在这里泛指一切帮助开发软件的软件。一台个人微机,一个编译程序,加上编辑、链接、装入等少量实用工具,就构成最简单的软件开发环境。随着计算机辅助软件工程(CASE)的开展,在软件开发的各个方面都研制了许多有效的工具。从零散的单个工具到配套成龙的工具(Tool Box),进而发展为集成于一体的集成化工具(Integrated Tools),标志了环境自动化程度的不断提高。集成化工具的自动切换,可以明显提高软件的生产率。

用户界面的友好和一致,是环境要求的重要品质。菜单、联机帮助和多窗口屏幕,被称为用户界面的三大友好技术。它们不仅为操作带来方便,也有助于提高开发的效率。由于现代环境通常都拥有大量工具,环境的研制人员都十分重视保持工具界面的一致性(Unification),即在同一环境中使用的工具尽可能采用相同或相似的用户界面,借以减少操作人员要记忆的内容,避免混淆。

图形用户界面的诞生,标志着用户界面达到的新水平。就个人微机而言,在 90 年代初期开发的窗口系统,其色彩丰富、图文并茂的图形用户界面,无论在直观性、交互性或一致性方面,均较基于文本的字符用户界面上了一个新的台阶。有些窗口系统还提供便利,支持用户在自己开发的应用程序中实现图形用户界面。

(3) 软件工程管理

软件工程管理的目的,是为了按照软件的预算和进度完成项目计划,实现预期的经济和社会效益。众所周知,即使拥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,软件生产也不例外。软件工程管理包括成本估算、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,现正日益受到工业界和学术界的重视。

1.1.2 软件工程范型

如前所述，软件工程方法学和软件工具都是软件工程的重要内容。方法学告诉你怎样定义和开发软件，而工具则对方法提供自动或半自动的支持。按照软件工程的观点，软件开发既要采用适当的方法和工具，还应遵循某种合理的“过程”(Procedure)。举例说，开发软件时要划分哪些阶段？在不同的阶段宜使用什么方法与工具，需完成哪些文档？怎样才能在所有阶段包括软件更改后保证产品的质量？这些问题都应该在“过程”中一一规定。所谓软件工程范型(SE Paradigm)——有时也称为软件开发范型或软件开发模型，实际上就是方法、工具和过程三者(Pressman 把它们并列为软件工程的“三要素”)的有机结合。

最早的软件工程范型是瀑布模型(Waterfall Model)。它奠定了 80 年代初软件工程学的基础。以后，人们又在实践中创建了快速原型和第四代技术等范型，逐渐形成了当今“多种范型并存”，开发者可按照软件的应用领域、规模和复杂性自由选择的局面。本节将简要介绍几种常用的范型。

1. 传统的软件工程范型——瀑布模型

瀑布模型是 1976 年由 B. W. Boehm 提出的，是基于软件生存周期的一种范型。它将软件生存周期分为定义、开发、维护三个阶段，每个阶段又分为若干个子阶段，各子阶段的工作顺序展开，如自上而下的瀑布，如图 1-1-1 所示。

定义阶段的主要任务是分析用户需求，分析软件系统所追求的目标，分析开发该系统的可行性。这一时期需要用户与系统分析员的交互和配合。

- 问题定义　收集、分析、理解、确定用户的要求，系统分析员在与用户讨论的基础上共同提出“软件系统目标与范围说明书”。

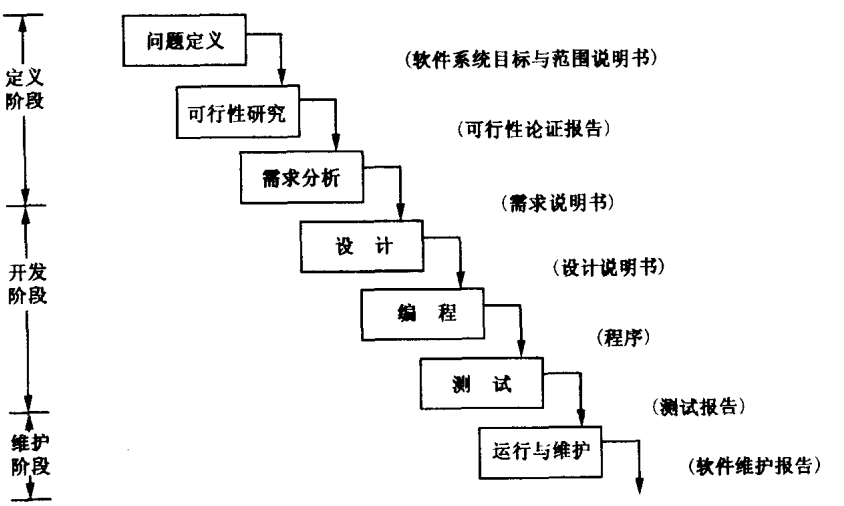

图 1-1-1　瀑布模型

- 可行性研究　确立对问题定义阶段确定的问题是否有可行的解决办法，并对各种可能方案做出成本/效益分析，系统分析员据此提出“可行性论证报告”，作为使用部门负责人决定是否继续进行这项工程的依据。
- 需求分析　确定用户对软件系统的全部需求，并以“需求说明书”的形式表达，其目的是

明确软件系统“做什么”。

- 设计　设计软件系统的模块层次结构，设计数据库的结构，设计模块的控制流程，其目的是明确软件系统“如何做”。这个阶段通常可分两步：概要设计和详细设计。概要设计完成软件系统的模块划分和模块的层次结构以及数据库设计。详细设计完成每个模块的控制流程的设计。这个阶段结束时应交付“设计说明书”。
- 编程　按照选定的程序设计语言将设计说明书中每个模块的控制流程编出相应的程序，得到软件系统的源程序。
- 测试　检查并排除软件中的错误，提高软件的可靠性。测试分为如下三个步骤：模块测试，测试程序的每个模块是否有错误；组装测试，测试模块之间的接口是否正确；确认测试，测试整个软件系统是否满足用户功能性要求。本阶段结束时应交付“测试报告”，说明测试的对象，测试数据的选择，测试结果是否符合预期结果。如果测试发现问题，经过调试，找出出错原因，然后进行改正。
- 运行与维护　主要任务是软件维护。其任务是修改软件系统在使用过程中发现的隐含错误，扩充在使用过程中用户提出的新功能要求，目的是维持软件系统的正常运行。本阶段的文档是“软件维护报告”。

软件生存周期虽然按瀑布模型可划分为若干阶段，事实上各阶段不可能理想化地进行分割，在实际开发过程中充满了回溯、反复和交叉过程。例如，在设计阶段发现需求说明书有不完整或不正确之处，就必须进行“再分析”；测试阶段发现模块界面有错误，就必须进行“再设计”；在运行阶段为了扩大系统的功能又必须进行“再分析”、“再设计”、“再编程”等。

2. 快速原型模型

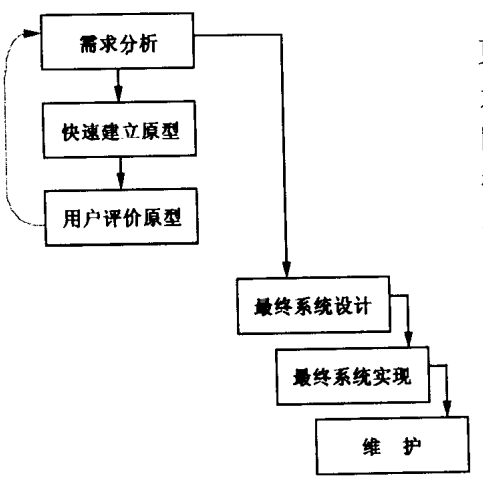

图 1-1-2　快速原型范型

为了弥补生存期范型的缺陷，使需求分析的结果更接近于待开发软件的真实需要，人们创建了一种称为“快速原型法”(Rapid Prototyping)的软件工程范型。图 1-1-2 显示了这一范型的典型过程。其主要活动包括：首先建立一个模拟待开发软件的原型，经过用户评价后提出对软件需求的修改(这种修改可能会反复多次)，然后根据用户和开发者一致认定的软件需求，设计和实现所需要的软件(即最终系统)。

原型是简化了的目标系统。根据不同的情况，又可区分为以下两类。

(1) 功能性原型(Functional Prototype)

这类原型着重模拟目标系统的主要功能。在多数情况下，原型的功能是目标系统功能的一个子集。

(2) 行为性原型(Behavioral Prototype)

着重于模拟目标系统的人机界面，包括输入方法和输出格式等，使用户了解目标系统将怎样进行人机交互(Interaction)。

实际使用的原型常常兼顾功能与界面两方面的要求，以便更好地显示目标系统的轮廓。

原型法具有以下特点。