现代软件技术概述

吴培稚　牟其铎　朱文林　刘天海　编著

测绘出版社

现代软件技术概述

吴培稚　牟其铎
朱文林　刘天海　编著

测绘出版社
·北京·

内 容 提 要

本书是一本关于现代计算机软件技术的概述性书籍，书中结合应用软件系统开发过程中所涉及的一些共同的问题，采用叙述为主、实现为辅的方式进行介绍。其主要内容有软件设计的思想方法、分析设计、数据、用户界面、智能技术等，同时简单介绍了数据压缩、多媒体、超文本、可视化、动画设计等新技术。期望本书对应用系统的开发者和项目管理者有所帮助。

本书面向具有一定经验的计算机工作者及项目管理人员，也可作为有关学科的高等院校学生和研究生的课外参考书。

图书在版编目(CIP)数据

JS267/17

现代软件技术概述 / 吴培稚等编著. — 北京：测绘出版社，1996.8
ISBN 7-5030-0847-4

Ⅰ.现… Ⅱ.吴… Ⅲ.软件－技术－概述 Ⅳ.TP31

中国版本图书馆CIP数据核字(96)第14858号

测绘出版社出版发行
(100045 北京复外三里河路50号 电话：(010)68523901)
北京交通印务实业公司印刷 · 新华书店总店北京发行所经销
1996年9月 第一版 · 1996年9月第一次印刷
开本：787×1092 1/16 · 印张：8.5
字数：200千字 · 印数0001－1500册
定价：12.00元

前　言

飞速发展的计算机技术和琳琅满目的软件，对于一个没有经验的应用系统开发者，并不总是一件好事。面对具体问题，出现“狗熊掰棒子”的现象就是一例。这种收获小、消耗大、重复劳动的怪现象来源于经验不足，来源于应用系统开发者心中缺乏基本的设计思想，没有一个基本的逻辑框架。这种基本设计思想和基本逻辑框架的取得，主要靠应用系统开发者自身在理论和实践两个方面的修养和积累。本书面向应用系统的开发研制者，期望对软件人员有所帮助。

本书分三部分，共七章。

第一部分即第一至第三章。这部分介绍的内容为概述、思想方法和分析设计，也是初步涉及应用系统开发者容易忽略的内容，它对基本设计思想和基本逻辑框架的取得很有益处，对应用系统开发的成败起着决定性作用，或者说具有战略意义。

第二部分即第四章和第五章。这两章介绍的内容为数据和用户接口，是应用系统中主要部分，也是软件研制中重用率最高的部分，合理使用重用可加快软件开发速度。

第三部分即第六章和第七章。这部分内容介绍有关智能和几项现代技术，对提高应用系统技术水平将有所帮助。应用系统中嵌入智能将越来越普遍，多媒体、系统集成等代表了计算机系统发展中融合和集成的方向。

附录既是本书有关章节的补充，也为读者查阅常见应用系统的名称、概念、术语提供了方便。

从整体而言，本书面向应用系统的开发研制，侧重于有关概念的介绍和软件开发。以叙述为主、实现为辅。书中还介绍了近几年被广泛应用和普遍重视的一些技术，贯穿全书的主线是面向对象的思想、技术和方法。

需要特别声明的是，书中的某些叙述和图件引用了有关学者已发表的论文和书籍，限于本书的性质等原因，书后没有全部列出有关参考文献，在此向他们表示歉意。

本书在编写过程中得到了孙其政、李宣瑚、修济刚、徐京华、成小平、高文海、李谊瑞等同志的指导，苗秀花同志负责全书的录入排版工作，对于上述同志的热情帮助，谨致深切的谢意。

在本书编写期间，牟其铎研究员不幸因病去世，在他生前不能看到本书的出版，值此致以深切哀悼。

由于我们水平有限，经验不足，加之时间仓促，难免出现疏忽谬误之处，敬请读者批评指正。

作者

1996年9月

目　录

第一章　概　述

诚然，要写好一本计算机技术方面的书是很不容易的。原因之一是这方面的书籍已经很多，与其它学科相比可用铺天盖地来形容；原因之二是计算机技术发展太快，先进的技术和优秀软件层出不穷，这种状况会造成写作过程中目标移动、最后成书时作者就已经感到遗憾。本书取名为《现代软件技术概述》，主要着眼于以下四个方面：

1. 现代：近几年来被广泛应用、普遍重视和深入研究的技术；
2. 软件技术：只涉及软件方面内容，除非十分必要才涉及硬件有关知识；
3. 面向应用：面向计算机应用系统的开发，并非介绍软件所有方面的知识；
4. 概述：以讲清软件开发设计的思想方法为目的，以叙述为主、实现为辅。

本书与一般教课书、工具书、程序设计等类的书籍不同，从软件开发的角度来看，也许本书的内容更带有全局性和普遍意义。

现代是一个相对的概念。在人类漫长的历史长河中，现代可以从本世纪算起，至少也有三五十年历史。然而，对于电子计算机科学，其全部历史也只有几十年，计算机科学的现代只能说是近几年。现代软件技术就是近几年被广泛应用、重视和研究的软件技术。现代软件技术与先进软件技术不是同一个概念，二者也许具有时间上的准同步性和内容上的相关性。数据库(Data Base)技术是现代广泛采用的技术，评价某数据库系统是否具有先进性则需要看该系统中采用了什么技术。人工神经网络(Artificial Neural Network—ANN)的提出可追溯到20年前，目前受到普遍的重视，成为一个研究热点，它是一门现代软件技术，具体到某个项目是否具有先进性，也要看系统中采用什么技术。上述的例子仅从单项技术的角度说明了现代软件技术和先进软件技术的关系。对于软件开发的整个过程(或者说工作方式)，存在着过去、现在和将来之分。1989年艾伦·凯关于软件开发的过去、现在和将来有一个说法，它是一张表，现抄录如下(表1.1)。

表 1.1　软件开发的过去、现在和将来

时期 / 项目	过　去	现　在	将　来
地　点	机　房	桌　上	任何地方
人　物	专　家	个　人	协作团体
干什么	编　辑	方案设计	和谐结合
怎么干	记忆和打印	看和指点	提问和讲述
方　式	数据／进程	面向对象	规　则

按照表1.1的说法，我们可以看到，软件开发的工作环境从机房转移到个人办公室桌上；工作重点由编写程序转移到方案设计；思想方法由数据/进程转移到面向对象……，整个软件开发工作的中心为系统分析和系统设计，而不是具体地编写、调试程序，程序的生成大部分由计算机来承担。既使是对将来的协同计算(Collaborative Computing)，近几年也进行了广泛的讨论和研究，同时取得了很大的进展，有的已经初步实现。尽管目前我国计算机开发应用的总体水平还低于国外发达国家，但已具备了良好的开发环境(政治、技术、物质和人员等)，计算机工作者必须不适时机地把握软件开发、应用及发展的这种必然趋势。

软件危机是计算机科学界讨论得很多的一个问题。软件危机主要表现在软件开发跟不上硬件的发展，跟不上用户的需求，跟不上信息处理的需要，成本和计划往往失去控制。目前正是被称为信息爆炸的时代，信息量的增长超过了软件的处理能力，即所谓“被数据淹没”(有些学者称这种现象为数据爆炸、信息饥饿。这也许更为确切)。原来僵硬的软件结构体系越来越不适应环境的变化和不断提高的用户需求，刚刚完成的计算机系统就可能已经是过时的系统。硬件技术的迅速发展，使得新技术、新工艺、新产品不断涌现，由于软件的瓶颈作用，用户不能很快从中得到实惠。大型软件系统开发失去控制，不得不一再宣布推迟交付，造成了巨大的经济损失和信誉损失，这已不再是鲜人为知的现象，即使是有影响的老牌的大公司也在所难免。软件维护费用虽已高达70%，而应用中意想不到的问题仍然接连不断，如此等等。面对软件危机这种难堪的局面，计算机科学界至今还缺乏应付这些难题的十分奏效的对策，目前普遍看好的是面向对象技术。

面向对象的思想方法比较自然，也更接近于人们通常的思维方式。面向对象思想的核心是把客观世界中的事物映射成对象，把事物间的联系映射成消息，以此为出发点，模拟问题空间。对象用数据属性和动作属性来描述，性质相同的对象归为类，同一类中不同的分支称为子类，子类和父类的关系是一种继承关系，对于在不同环境条件下呈现不同状态的现象称为多态性。面向对象的这些基本特征体现了从一般到特殊和特殊到一般的思维方式，也就是演绎——归纳的思想。面向对象的思想已有二十多年历史，近几年得到了广泛的重视，也取得了很大的发展，尤其对于中、大型计算机系统的研制其优势更为明显。有人认为面向对象技术和它在代码重用方面的优势，可能是克服软件危机的重武器，是软件技术的主流。面向对象技术的产品目前发展很快，这为我们广泛采用该技术创造了有利条件，然而对于那些理论上和技术上尚未很好解决就进入市场的某些产品，或那些配套尚不完备的产品，使用中我们必须顾及它们的副作用。应用系统开发的根本目的是解决具体问题，而不要求它在所有方面都具备先进性，发展过快造成一定混乱或导致有关标准的不统一，也是一种负影响，因为不尊重标准化的产品很难适应环境的变化和未来进一步发展。目前关于面向对象技术的标准已经出台，但还不够完善，有的尚有争议，面向对象技术的某些理论问题尚未彻底解决。上述这些问题在进行应用系统开发时必须考虑，尤其是全面采用面向对象技术的应用系统中，软件、硬件、网络支持等各个环节的配套，必须慎重处置严格论证。然而我们也不必为面向对象技术的不够完善而拒绝采用面向对象技术，它是具有很强生命力的新事物。回顾关系数据库的发展历史，我们可以从中得到启发，过分保守的投资，得到衰老的计算机系统，从总体上说是不科学的。

要在应用系统和非应用系统之间划出一条严格界限实际上是比较困难的。例如，管理信

息系统(Management Information System—MIS)早期属于应用系统，后来由于这样系统需求量太大，根据它们的共性形成了一套专用开发方法，逐渐发展成为计算机科学的一门分支，并在课堂上讲授。一般地说，一个部门、一个单位或一个企业需要建立一个应用系统，系统的开发往往需要经过系统分析、设计、实现、测试、维护等阶段，这五个阶段从头到尾顺序地线性地实施，通常称为瀑布式生命周期，这种方法在早期得到广泛的发展，但目前被认为是一种僵硬的方法，也是造成软件危机的重要因素之一。从解题逻辑角度来看，上述五个阶段的思路是合理的、正确的，要改变的是线性的实施方法及传统的分析方法。

本书的章节安排如下，第一章概述部分，介绍本书概况及必要的说明。第二章思想方法，介绍如何用形象思维方法去认识客观世界，用面向对象方法去分析问题世界。第三章分析和设计，通过面向对象与面向过程和面向数据分析方法的对比，指出了面向对象分析法的优越性，叙述了面向对象技术的基本概念和一般原则，同时介绍了当前几种流行的面向对象的分析法。第四章数据和第五章用户界面，这两章的内容涉及计算机软件学科的两个基本领域，数据和用户界面是应用系统中主要的两大部分，也是应用系统开发中重用率最高的两大部分，同时也是面向对象技术最早涉及的两个领域，当前这两个领域得到了广泛重视。这两章采用由简单到复杂，由过去到现在直至将来的方式进行叙述。第六章智能，介绍应用系统中常用的几种实用技术和方法，智能技术已成为或将成为第三个被广泛应用到各种计算机系统中的一门技术。第七章几项现代软件技术，也是本书的最后一章，介绍了应用面较广，专业化、专用性较强的几项技术，如数据压缩技术、动画技术等。有些技术如多媒体技术，代表了计算机学科今后发展的方向。附录以简洁的形式和较小的篇幅，介绍了目前广泛流行的一些软件系统及计算机处理技术，同时还列出了一些常见应用系统的缩写词，从某种意义讲也是本书的一种补充，主要供读者查阅。

应用系统开发中，受制约的因素很多，主要因素有目标、经费、人员和计划进度。这四个因素之间相互联系又相互制约，目标较高时一般需要较多的经费和人员，完成所需的时间较长。在目标、经费、进度不变条件下，人员过多会造成浪费甚至还会产生负作用，人员过少又会使进度放慢。在这里，人的作用并不遵守 $1+1=2$ 的规则，有时或许 $1+1>2$，有时可能 $1+1<2$。开发应用系统需要多方面人员的协同工作，既要有计算机人员、领域专家，又要有决策者、投资者、操作使用人员等。从总体上看，开发应用系统，需要一个良好的环境。所谓环境，即与项目有关的所有方面，正确处理各种因素是很重要、很关键的，有时直接关系到系统开发的成败。

目标问题是应用系统开发中不能回避的问题，目标的背后又隐含着一系列问题，在这些问题没有明确之前，其他均不宜草率从事。应用系统要达到什么水平，这是国内项目必须考虑的一个因素，目前应用系统开发似乎存在着水平要求过高的倾向。要求过高的先进性，不仅会给开发增加难度，延长完成时间，增加高水平技术人员的投入，而且不切实际的高要求还可能导致项目的夭折。反之，先进性要求过低又会使项目完成后，难以适应环境的变化和信息社会的客观需求，提早进入“老年期”。有些学者认为，先进性以提前五年为度，即五年后的水平就是立项时的目标。

图 1.1是信息系统发展概况，对于应用系统的开发，确定系统目标有一定的参考价值。应用系统开发者只能根据实际情况，从中作出准确的选择，而不是过分追求先进性(图 1.1中，

下部的方框，先进性一般较高），避免项目开发过程中陷入进退两难的境地。

可行性研究也与目标有关，项目目标确定后必须进行可行性研究，要从人员、经费、市场风险、法律、技术水平、环境条件等进行较详细的实事求是的估计和预测，明确有利条件和不利因素，要充分地估计出成功后的效益和存在的问题，给出不成功的风险概率和相应的补救措施，提出项目实施与否的倾向性意见。至于更详细的问题分析，或者说蓝图的产生将在系统分析和设计阶段进行。项目的系统分析和设计过程一般需要较长的时间，它不仅决定下一步实施的方方面面，也可以对原定目标进行修改和推翻，它要明确说明问题的实质以及实现的每个细节，而不再是一般的预测和估计。对于某些较大的项目，还可以建立模拟系统进行试验。

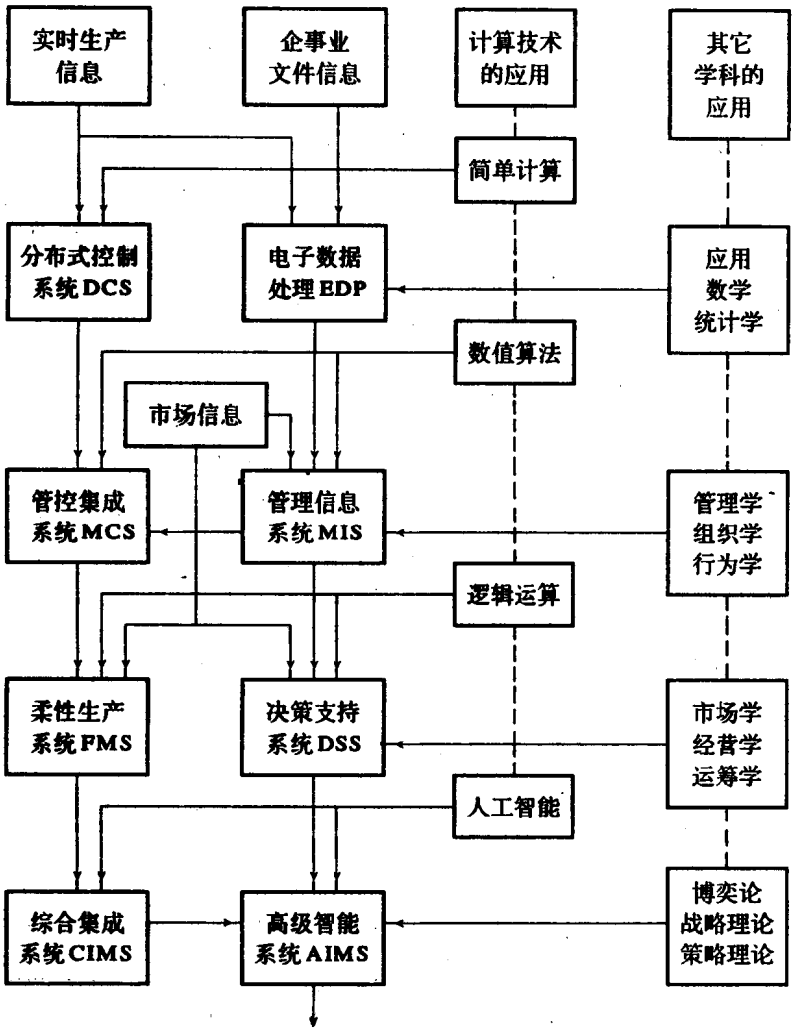

图 1.1 信息系统的发展

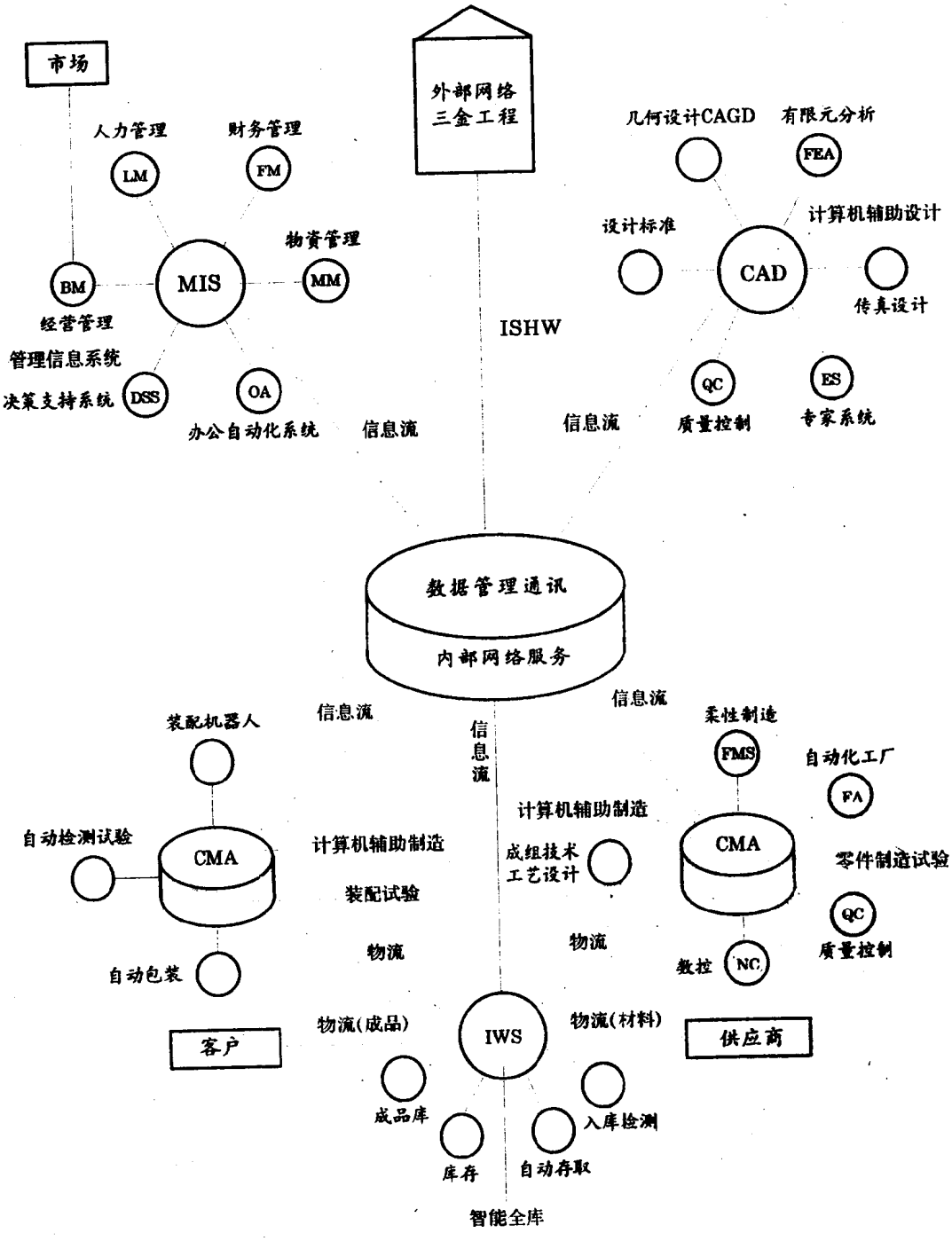

图1.2 工厂信息化示意图

图 1.2是工厂信息化结构示意图，该图的全面实现将是一项十分艰巨的工程，我们只能一部分一部分地分期实现，每期只实现图中一个或几个圆圈。在实现过程中既要考虑眼前的能力、预期效益，又要兼顾长远规划和技术进步等因素。事实上，图 1.2把工厂信息化绝大部分内容包括进去了，或者说，对于一个具体的工厂，图 1.2具有长远规划性质，工厂当前要实现的信息化工程是总体规划的一部分，实施时主要考虑长远规划的约束和未来发展的需要及当前效益。

应用系统的实施和工厂式的施工不同，工厂式施工模式——盖房子、装设备、培训人员、开工，最终目标是源源不断地生产某些规格产品。该模式不适用于应用软件系统的开发，从本质上看，这是两种根本不同的事情。尤其是项目刚开始，在有关问题分析尚未清楚的条件下，就急于购置设备、建立工作环境等，这是计算机应用系统开发中所不允许的，应用系统开发则更重视前期的“纸上谈兵”（将在第二章中作介绍）。

一切自己动手的思想往往影响着应用系统的开发，不仅技术人员有这种思想，决策者也有这种思想倾向，总感到这样做最保险，是我们自己的东西，别人抢不去夺不走。以这种思想来指导应用系统的开发一般会造成研制周期长、项目水平低、投资效益差、低层次的重复开发、维护工作量增加、修改和扩展困难、对环境适应能力下降等诸多弊病。其实一切自己动手本身就不成立，因为机器、操作系统、支撑软件等一般不是自己开发的，应用系统开发中一切自己动手做也是很困难的。问题的正确提法应该是：建设一个计算机系统，应首先确立哪些应该购买？哪些应请别人做？哪些应自己做？更进一步的提法是系统开发中是尽可能自己开发，还是尽可能利用现成的商品？正确的答案不是前者，而是后者——尽可能利用现成的商品，包括硬件商品和软件商品，从整体上来衡量是“合算”的。这种合算除经费因素外，还包括时间、人员、技术、可靠性、难度等因素。尤其是软件，市场上的商品化软件一般要比自己开发的软件更经得起时间和用户的考验。重视硬件忽视软件的倾向大量地存在着，它使相当多的计算机系统建成后失去了应有的效益，过几年甚至成为单位的一种负担。

本书作为概述，以介绍设计思想、设计方法、面向应用为目的，不涉及软件的所有方面，也不像教科书那样把所有细节完全交待清楚。例如操作系统是软件系统的重要方面，随着多媒体技术的引入和网络技术的发展，与之相适应的面向对象的操作系统不久将面世，这在本书中并未作深入的阐述。硬件的介绍也只是在不得已的情况下简单地提一下。为了帮助读者理解数据和用户界面等内容，本书在第四章及第五章中提供了部分程序，这些程序使用C++语言编写，所以选择C++的原因是考虑到该语言全面支将面向对象技术，尽管目前Smattall、Fortran、Basic、Lisp、Pascal等语言都有了面向对象的功能，然而C++语言的表达能力及使用的普遍性等是其它语言不能比拟的，C++已成为面向对象语言的主流。

本书作为概述，不能像有关学术著作那样引经据典，详尽地列出参考文选，作者姓名等，成书过程中参阅了大量的书籍和论文，有些图件来自有关论文。书后虽然列出了不少参考文献，遗漏之处在所难免，在此谨向有关作者表示歉意。

作为概述，本书中引用到很多名词和概念，对它们的解释也不像一般专业书籍那样严密。对于使用频率较高带有基础性的概念，尽可能给出了较普遍和公认的定义和解释；对于一般性概念，视情况作了相应的说明；对于使用频率较低的概念，采用直接引用，未作进一步的解释；对于一些翻译过来的概念，尽可能给出英文原词，避免引起不必要的误解。另一方面，

由于计算机科学发展十分迅速，应用领域越来越广泛，有些概念虽然多次出现，大家也在不同程度地引用，但尚无确切的统一的定义；有些概念只能按特征描述方式给予说明，给读者一个只能意会的感觉；有些概念甚至在不同场合具有不同的含义。因此，读者在阅读本书时，应从上下文的联系中去理解问题的本质。

由于我们水平有限及客观条件的限制，有些现代软件技术未能写入本书中，书中的某些叙述及说法可能不甚确切，错误之处在所难免，值此恳请读者批评指正。

第二章　思想方法

软件的开发和研制与科学研究一样，需要思想方法。正确的思想方法往往是顺利解题的关键，早期软件系统研制的思想方法受科学研究中常采用的抽象思维(abstract thought)方法的影响较大，采用面象过程的方法研制软件系统就是这种思想的一种反映。随着计算机处理信息的多样化，软件系统的大型化、复杂化，抽象思维的思想方法越来越不适应软件系统研制的需要，形象思维(figure thought)的方法逐渐为计算机学界所接受，这几年得到很大发展的面向对象(object-oriented)的方法就是这种转变的一种反映。

§2.1　软件开发的四个阶段

软件系统的开发研制一般可分为以下四个阶段，它们是：

系统分析 (analysis)

系统设计 (design)

系统实现 (implementation)

系统测试 (test)

这四个阶段和系统维护一起被称为软件的生命周期(lift cycle)。用通俗的语言来叙述，这四个阶段也可称为：

做什么

怎么做

实现它

对不对

实际上，这四个阶段反应了一个求解过程，有些作者也把软件开发过程称为求解过程。一般地说，系统分析阶段深入到研究的问题空间，通过反复分析、抽象，产生系统模型，完成系统任务描述；系统设计阶段在求解空间中进一步扩充，细化模型，选择适当的程序设计语言用于实现系统；系统实现阶段按照设计要求进行实施、编码、调试，要求有良好的设计风格，充分考虑程序的可维护性和易扩展性等；系统测试阶段则采用“破坏”的思想方法，发现错误，排除错误，同时对整个系统进行复审。系统模型是这几个阶段的核心和联系的纽带，在上述每个阶段上，软件人员的思想方法和工作方式既存在着联系又存在着差异。

一般地说，上述四个阶段，后一阶段是在前一阶段的结果上开展工作的，这是一种链条式工作方式，继而产生了“瀑布式”生命周期(waterfall life cycle)。用瀑布式工作方法开发应用系统，软件供需双方的界面是合同书以及随合同书而附的一系列说明。相当多的软件产品都是这样产生的。

§2.2 瀑布式开发遇到了困难

人们都有过这样的经历，解决比较简单的问题，对于做什么、怎么做、实现它、对不对这四步，可以按步就班地一步步进行。解决复杂一些的问题，这种直线式的解决方法就行不通了。也许在实现阶段要求修改模型，在设计阶段需要作模拟实验，用来进一步证实设计的正确性，这些都是解题的需要。软件人员对于能一步步按四阶段实现的软件系统，常采用瀑布式开发法。对于完全不能按四阶段顺序实现的软件系统，则采用渐进式开发法(increments)，有些软件系统甚至需要软件人员跟随需求者一起工作，直到项目完成。上述两类软件开发，可以说是两种极端的情况，绝大多数软件系统的开发处于二者之间。处于二者之间的软件系统开发则应权衡利弊，妥当选择，也可以选择某种改进型的开发方法。方法的选择主要取决于人们对项目的理解程度，它包括委托者、操作者、领域专家、计算机专家等有关人员，以及人员之间相互沟通。实际上完全不理解和完全理解项目的人员只是绝少数，大部分都是有所了解，只是在了解的程度上差别较大。

瀑布式开发方法对项目有关人员的要求是：

1. 委托方

对委托方，软件系统开始设计后不能提出变动要求(需求稳定)，系统分析期间要能准确地表述项目的目的、要求、功能等(描述准确)，项目结束之前只能看到纸上的合同、计划、图表、说明等，不能看到大致模样，也不能进行试运行以便更深入地分析研究，期望得到更理想的效果(不支持原型开发)。

2. 软件人员

要求软件人员能准确地理解委托方的种种需求，准确地用专门语言表述系统模型，描述系统任务。设计人员忠实地分析、完善、细化模型、完成解题方法，产生大量说明、表格、图形，选择语言工具。系统实现按设计要求实现解题，程序员可以利用原有软件资源，或进行必要的改造后使用。测试阶段则以“破坏”思想为主导，采取一定方法(如黑盒法、白盒法)检查并纠正错误。在项目进行过程中，必须严格把握项目的开支和进度、组织规模与项目规模相适应，充分估计因错误和变动引起的连锁反应所造成的影响。

图2.1是瀑布式开发模型示意图。由图可见，在开发过程的各个阶段之间，都要形成一系列文档材料，有关人员要对上一阶段工作结果进行评审，这很像工厂里的流水线，每道工序的产品都要受到检验，然后进入下一道工序。瀑布式模型是一种直线式链条式开发方法，软件产品不同于工厂里流水线上单一规格产品，软件产品被用来解决客观世界中各种各样千变万化的问题，面对复杂的客观世界，面对计算机越来越广泛的应用，瀑布模型遇到了困难。

过于严密、要求过高的瀑布式开发法，虽然适用于工厂式的软件开发，但越来越不适应大、中型计算机系统的研制。瀑布式开发法的封闭性、直线性造成了它的局限性，其实质是把人们认识世界多次逐步深化的过程变成了单次认识过程。逐步深化过程是在继承原有的知识基础上多次迭代往复的深化过程，其中既包括了从一般到特殊的演绎过程，也包括了从特殊到一般的归纳过程。软件人员采用瀑布式开发法完成一个软件后，总结一下本项目的经验教训，往往认为本项目是一个遗憾工程。遗憾之处可能是：项目是完成了，也能运行了，运行也

没有错，但某部分结构不合理，某些地方没有采用更先进的技术，某些用户界面使用不方便，经费开支和项目进度不理想等等。造成遗憾的原因很可能是由于瀑布式链条式施工造成的。

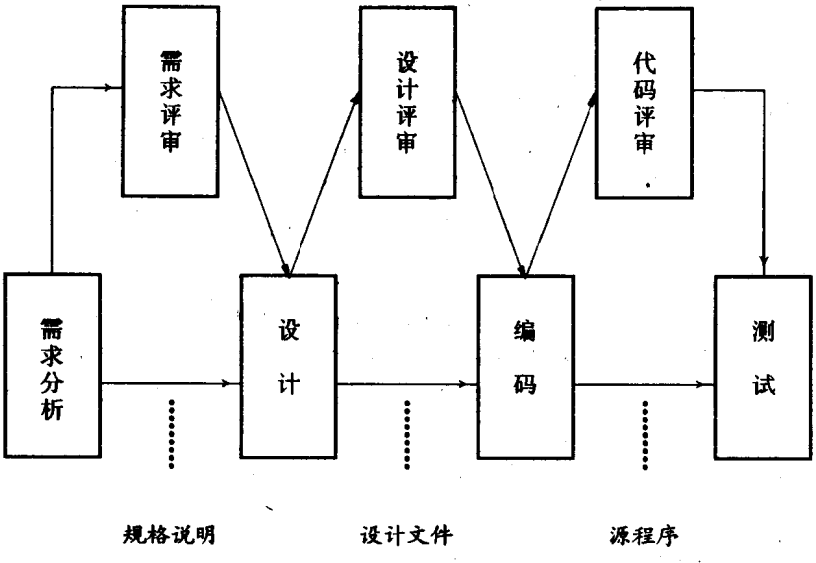

图2.1 瀑布式开发模型示意图

§2.3 系统模型是核心

面对一个项目，按照一般解题规律，首先分析题目本身，明确那些是已知的，那些是未知的。对于已知部分可以用已有的方法、工具、软件先行解决。对于未知的部分，可以分为简单的和复杂的两大类：简单的问题采用上述的四个阶段过程加以实现；复杂的问题则可通过多次的四阶段过程方能解决。从整体上看，这里已不再是瀑布式的直线式的解题过程，而是把整体分成了许多块，一块块地实现的。当然整体不是这些块块的杂乱无章的堆积，块与块之间不仅互相联系，而且互相作用，由块块及块块间的联系组成的整体就是解题模型，习惯上人们称之谓系统模型(system model)。

从人们的认识过程来看，开始阶段的模型可以是粗糙的、不完善的，甚至可能是整体中的一个局部。对于已知的部分是清晰的，未知部分是框架式的。对于简单部分是清楚的，对于复杂部分是模糊的。软件人员往往从简单的框架出发，不断地与项目有关人员进行交流，分析研究，在实际工作中不断深化，变未知为已知，使模型从上下左右各个方面上得到完善。因此系统模型应该说是由项目全体有关人员共同完成的，同时它将又对全体人员具有联系和约束作用。在这里委托者、领域专家、操作者仍按照他们习惯的工作方式和熟悉的领域知识审查模型、完善模型。软件人员如系统分析员、设计员、程序员、测试员均按照一定的工作方式和思想方法审查模型、完善模型，在模型规定的内容下工作。当然作为系统分析员，他们是模型的主要生产者。当前一个软件系统往往需要几十人乃至几百人共同完成已很常见，要