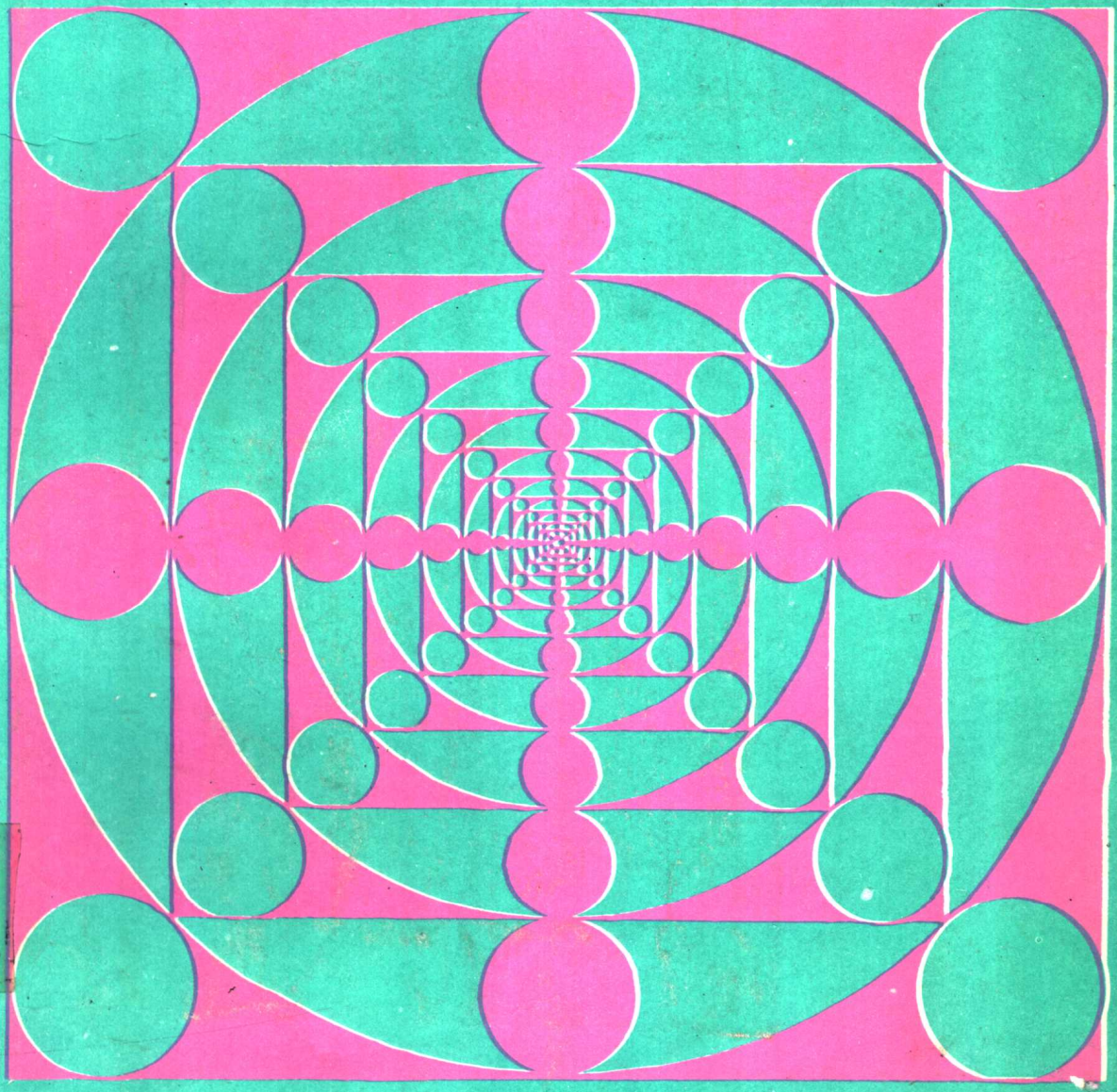
电子计算机应用系列教材

软件工程方法

郑启心 编著



科学出版社

电子计算机应用系列教材

软件工程方法

郑启心 编著

科学出版社

1993

(京)新登字 092 号

内 容 简 介

本书是电子计算机应用系列教材之一，书中全面介绍目前国内外常用的软件工程方法．全书共分八章．第一章简单介绍软件工程的基本概念和主要内容；第二章综述用于软件开发需求分析阶段的 SA 方法及用于设计阶段的 SD 方法；第三章介绍以数据结构为基础的 JACKSON 设计方法；第四章通过实例详细叙述软件蓝图；第五、六章讨论结构化程序设计，逐步求精技术及 PAD 方法；第七章阐述软件测试的概念、方法和步骤；第八章探讨软件工程的发展方向．

本书可作为高等院校计算机类有关专业课程的教材，也可供从事计算机软件工作的科研人员参考．

电子计算机应用系列教材

软件工程方法

[illegible] 编著

责任编辑 鞠丽娜

科学出版社出版

[illegible]

邮政编码 100703

北京市桦星电脑技术应用部激光照排

天津市蓟县燎原印刷厂 印刷

新华书店北京发行所发行　　各地新华书店经售

*

1993 年 5 月第 一 版　　开本：787×1092　1/16

1993 年 5 月第一次印刷　　印张：10 1/4　插页：1

印数：1—3 600　　字数：230 000

ISBN 7-03-001612-2/TP・117

定价：7.90 元

电子计算机应用系列教材主持、组织编著单位

主持编著单位：

国务院电子信息系统推广应用办公室

组织编著单位：

广东、广西、上海、山东、山西、天津、云南、内蒙古、
四川、辽宁、北京、江苏、甘肃、宁夏、江西、安徽、
河北、河南、贵州、浙江、湖北、湖南、黑龙江、福建、
新疆、广州、大连、宁波、西安、沈阳、武汉、青岛、
重庆、哈尔滨、南京等35省、市、自治区、计划单列市

电子振兴
计算机领导小组办公室
科技工作

电子计算机应用系列教材联合编审委员会名单

（以姓氏笔画为序）

主编审委员：

王长胤* 苏世生 何守才 陈有祺 陈莘萌* 邹海明* 郑天健
殷志鹤 童 颍 赖翔飞 （有"*"者为常务主编）

常务编审委员：

于占涛 王一良 冯锡祺 刘大昕 朱维华 陈火旺 陈洪陶 余 俊
李 祥 苏锦祥 佟震亚 张广华 张少润 张吉生 张志浩 张建荣
钟伯刚 胡秉光 高树森 徐洁盘 曹大铸 谢玉光 谢育先 韩兆轩
韩培尧 董继润 程慧霞

编审委员：

王升亮 王伦津 王树人 王振宇 王继青 王翰虎 毛培法 叶以丰
冯鉴生 刘开瑛 刘尚威 刘国靖 刘晓融 刘德镇 孙令举 孙其梅
孙耕田 朱泳岭 许震宇 何文兴 陈凤枝 陈兴业 陈启泉 陈时锦
邱玉辉 吴宇尧 吴意生 李克洪 李迪义 李忠民 迟忠先 沈林兴
肖金声 苏松基 杨润生 吴福德 张志弘 张银明 张 勤 张福源
张翼鹏 郑玉林 郑 重 郑桂林 孟昭光 林俊伯 林钧海 周俊林
赵振玉 赵惠溥 姚卿达 段银田 钟维明 袁玉馨 唐肖光 唐楷全
徐国平 徐拾义 康继昌 高登芳 黄友谦 黄 佩 程锦松 楼朝城
潘正运 潘庆荣

秘 书 组：

秘 书 长：胡茂生
副秘书长：何兴能 林茂荃 易 勤 黄雄才

前　言

随着计算机在各个领域的广泛应用，需要对大量被传输和存取的信息进行保护，这样，计算机安全保密就成了计算机科学中的一个重要的研究课题，特别是在70年代以后，由于出现了强有力的基于加密的协议，并开辟了新的应用领域，密码学得到了迅速的发展. 1977年1月15日，美国国家标准局采纳了一种加密算法作为联邦的标准——数据加密标准(DES)，从而成为密码研究和设计方面的一个新的里程碑. 1980年12月，美国国家标准协会采纳这个算法作为美国的商用加密算法，随后人们又提出了公开密钥密码的新思想，这种密码目前还在研究之中. 人们研究密码术，不仅将它应用于网络通信中的数据加密，还将它应用于存储介质的文件加密等方面，因此，进行计算机安全保密的研究对于保护政治、经济、军事和其他部门的信息具有极为重要的意义. 为了满足广大读者了解和掌握数据安全保密的原理和方法的需要，我们编写了这本书.

本书是电子计算机应用系列教材之一. 全书共分八章，主要介绍计算机安全保密的发展情况和立法情况，阐述密码术的基本原理，讨论常用的加密算法，并以存储介质和通信网络的安全保密为重点，讨论实施计算机安全保密的技术和方法，如存储介质的防复制技术、软件加密、加密系统和加密程序的防跟踪破译措施、网络通信的信息加密，以及微机数据库安全与保密等. 全书各章尽可能结合 IBM PC 微型计算机，列举大量已运行通过的程序实例，以供读者进一步研究时参考. 本教材适用于中级计算机应用科技人员的研修学习，也可作为大专院校高年级学生和研究生以及其他有关专业人员的参考书.

使用本教材时应注意除课堂讲授以外，必须上机实习(可参考教材中列举的实例)，在作习题的基础上开展一些专题讨论和研究工作. 本教材的重点是讲述存储介质和通信网络的安全保密原理及实施技术. 学习本课程要求具备汇编语言程序设计、计算机网络、数据库和操作系统等方面的知识.

本教材由王化文编写第四章和第七章，张德向编写第五章和第六章，吴亮编写第一章和第八章，张能胜编写第二章和第三章，李立参加了第四章和第五章的部分编写工作，由王化文统编全稿. 本书由河北机电学院赵惠溥副教授担任主审，在此谨向他表示诚挚的谢意. 由于编著者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者指正.

编著者

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第一章 概 论

1.1 软件工程的起源

软件工程是在计算机的应用日益广泛、对软件的质量和生产率要求越来越高的情况下发展起来的一门新兴学科.

软件工程的定义有几种表达方式. P. Wegner 和 B. Boehm 的定义是：科学知识在计算机程序的设计和构造，以及开发、运行和维护程序所需文档的编制中的实际应用.

叶祖尧 (R. T. Yeh) 的定义是：基于工程和科学重要原理的开发软件的规范和系统的方法.

L. Bauer 的定义是：为经济地获得能在计算机上运行的可靠的软件所应确定和使用的健全的工程原理.

这些定义在表达方式上虽不尽相同，然而在实质上却是一样的. 1986 年在由美国通用电气公司出版的《软件工程手册》一书中，对软件工程作了更确切的定义："软件工程是应用于软件定义、开发和维护的一套方法、工具、文件、策略、标准和步骤."为了更深入地理解软件工程的含义，让我们回忆一下计算机的发展过程.

自从第一台电子计算机出现以来，迄今已有 40 多年了. 计算机不仅大大改变了社会的面貌，而且自身也处在不断的剧烈变化之中. 计算机硬件的发展经历了从电子管时代、晶体管时代、集成电路时代直至大规模集成电路时代. 随着计算机应用的日益广泛深入，计算机软件的类型、数量、规模和复杂程度也在不断发生变化.

在计算机系统发展的早期，软件仅仅是作为硬件的附属品而存在着，那时，编制程序一般被看作是一种技巧，一种表示个人聪明才智大小的能力，而在程序的编制中很少使用系统的方法. 因此往往使软件开发过程难以管理，以致造成进度拖延或者成本超出计划范围.

在软件尚未形成商品即所谓产品软件（为了卖给他人而开发的程序）之前，多数软件都是由个人（或某个机构）开发和最终使用的. 软件由谁编写，就由谁运行，出了故障也是由谁排除. 因为工作流动性小，所以管理者可以保证当遇到故障的时候，软件编制人员会在现场. 在这种私有化的软件环境中，设计过程是隐蔽在人们头脑中的，编制的文档资料一般不够完整. 在这期间，人们摸索出了一些实现以计算机为基础的系统的方法，但是仍然缺乏系统的观点.

计算机系统发展的下一个阶段是从 60 年代中期到 70 年代中期的 10 年. 多道程序设计、多用户系统引入了人-机对话的新概念. 交互技术开创了一个计算机应用的新天地，从而使硬件和软件技术进入了一个新阶段，实时系统能够从多个数据源收集、分析和传送数据，能在几毫秒内对过程施加控制并产生输出. 联机辅助存储设备的发展促成了第一代数据库系统的出现. 随着以计算机为基础的应用系统数量的不断增长，计算机软件库在迅速膨胀，程序的规模也日见庞大（有的高达数万条甚至数十万条源程序语句）. 这时问题发生

了：当检测到故障或用户要求变化时，往往要对整个程序进行维护（修改、扩充），为此不得不付出高昂的代价．这与私人化的软件生产环境形成了尖锐的矛盾．与此相联系的问题还有：

（1）项目管理人员难以估计项目所需经费和时间，项目往往不能按期完成；

（2）技术人员难以预料系统的成败；

（3）研制人员各行其是，没有统一的标准可以遵循；

（4）项目有时会失去控制，从而造成失败，如此等等．

这些都是所谓“软件危机”的表现．

针对上述问题，人们思考和探讨解决办法，开始感觉到传统的手工业式的软件生产方式已不能适应要求，而应该仿照硬件的生产方式，采用工程化的方法来生产软件．1968年，在原联邦德国 Garmisch 召开的北大西洋公约组织的一次会议上，软件工程这一新名词被提出来作为会议的名称和主题．从此，软件工程作为一门新兴学科日益活跃在计算机科学领域内．

软件工程从它诞生之日起，就显示了强大的生命力．1971年IBM公司运用一些软件工程技术成功地研制了纽约时报情报库系统和空间实验室的飞行模拟系统．尽管两个系统都很庞大，用户要求又有很多变化，但是在减少人力和削减经费的情况下，由于适当地采用了工程化的技术，最终还是按时、高质量地完成了任务，软件生产率比以前提高了一倍．

由于软件工程的研究成果为软件的研制生产带来了巨大的经济利益，有力地促进了软件产业的形成，因此软件工程日益受到人们的注目．例如，从1975年起，几乎每年要举行一次国际性的软件工程专业会议，讨论软件工程的进展和存在的问题．

面对新技术革命的挑战，一些发达国家纷纷把软件工程的研究列为国家重点发展项目．例如，美国国防部投资4亿美元用于一个名为SIARS的软件技术开发计划，其主要目标是在今后7到10年内将软件生产率提高数十倍．英国政府准备耗资4亿9千万美元推行一个为期5年的全国性计划，以改善英国在国际信息市场上的竞争能力．该计划将在软件工程、超大规模集成电路、人机界面及人工智能4个领域内开展研究工作，欧洲共同体和日本也有类似计划．

我国软件发展虽有30多年的历史，但总的说来，软件开发的技术水平与国外先进国家的水平相比，尚有较大差距．软件技术人员的结构和软件开发管理方式都还不够理想，在软件人员中缺乏高、中级人员（系统分析员、项目管理人员等），软件低水平重复的现象也较多，大型复杂软件除少数领域外尚不多见，软件开发中的管理问题还未引起足够的重视．为了迅速提高我国软件的生产水平，提高我国软件产品在国际上的竞争地位，国内有关部门及机构正在大力开展软件工程的研究并取得了可喜的成绩，同时正在制订适合我国情况的软件开发规范．另一方面，还必须看到，让更多的软件人员掌握软件工程知识，对于提高软件生产和管理水平具有重要的现实意义和战略意义．

下面，我们介绍软件工程中的几个重要的基本概念．

1.2 软件工程中的基本概念——生存期

在软件工程中，软件与程序有着明显不同的含义．所谓程序是指：为了使计算机实现

预期目的（计算某个问题或控制某个过程）而编排的一系列步骤．程序可用机器指令编写，也可以用程序设计语言来编写．而所谓软件指的是：计算机的程序以及开发、使用和维护程序所需的所有文档．软件的这种定义是为了强调在研制过程中及时地按一定规格产生各种文档，这乃是研制工作的有机组成部分，必须充分地重视．

软件的概念是程序概念的拓广与延伸，因此对软件的研究也就不能仅仅局限于程序的编制，而应当着眼于软件的形成、发展乃至失效的全过程．这就自然地导出了生存期的概念．生存期是工程中的重要概念，也是软件工程 中的重要的基本概念之一．软件工程中的方法、技术与工具都是针对生存期各个阶段给出的．每个阶段各有其确定的任务，每个阶段的完成都产生相应的文档并作为下一阶段工作的依据．在机械工程中，一台机器的生存期要经过分析要求、设计、制造、测试、运行等几个阶段．为了用工程化的方式有效地管理研制软件的全过程，软件的生存期也可以分为以下几个阶段：

- 需求分析；
- 设计；
- 编码；
- 测试；
- 运行与维护．

我们可以用图 1.1 的方式表达上述 5 个阶段．

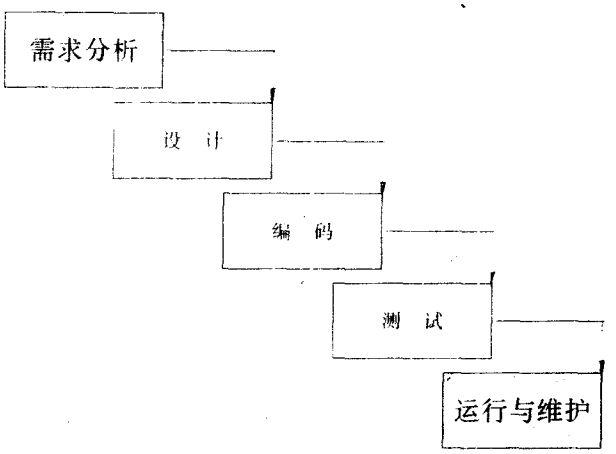


图 1.1

需求分析阶段的任务是理解与弄清用户对软件的要求．参加这一阶段工作的软件人员是系统分析员．系统分析员通过对用户现场的调查、了解及与用户的讨论、访问搞清用户的意图和要求．通常，用户对软件的要求是用不严格的形式提出的，例如，“速度越高越好”，“结果越准确越好”等类，有时，甚至会提出自相矛盾的要求．因此，在需求分析阶段需要把这些要求明确化、精确化，能用数字形式规定的就用数字形式规定，例如“响应时间不得大于 3 秒”等．这个阶段末尾产生的文档是规格说明书，它以文档的形式把用户对软件的要求固定下来，它是用户与软件承包单位共同达成的协议，是下一阶段——设计阶段的出发点和依据，它对以后各阶段的工作影响极大．经验证明，为更改需求分析阶段

发生的错误所付出的代价要比更改以后阶段发生的错误高得多.

设计阶段通常又分为初步设计阶段和详细设计阶段.

初步设计阶段的目的是将一组定义好的需求转换成一种可工作的软件结构，而软件结构是指确定模块间的关系并指明软件控制的层次. 初步设计的主要任务是：定义通过软件子系统的数据流；为软件确定一个逻辑上一致的模块结构；验证每个模块的功能划分；建立模块间的接口关系；以及说明约束限制条件. 初步设计也定义数据项、格式、长度和存取机构. 初步设计工作还应集中工具有“全局”性质的那些方面，即用于多个而不是一个模块的数据项. 初步设计的结果是初步设计文件说明. 为了保证对需求的可跟踪性，以及技术的透明性和正确性，初步设计文件说明应接受初步设计评审.

在初步设计评审中，对初步设计作了必要修改并认为满意之后，就可以开始详细设计了. 在初步设计阶段已经建立了软件结构和接口说明，现在的工作应集中于各个模块的规格说明上. 在详细设计阶段，初步设计的结果被进一步充实，对每个模块都作过程描述. 详细设计的成果是一份以书面形式提供的关于每个模块的完整的说明. 它是编码和单元测试的基础.

接下来进入编码阶段. 由于在详细设计过程中已经明确地给出了模块内部结构，因此编码的任务就是把设计用适当的程序设计语言表达出来. 每个模块使用的编码语言，在没有严格的性能限制时，都应当选用高级语言（如 FORTRAN，COBOL，PASCAL，Ada 等）. 当高级语言不能达到性能要求时，可采用汇编语言. 但是，一般来说，通过改变算法来改进性能比用汇编语言重新编码强得多. 编码过程本身应遵守结构化编码原则，即尽量使用顺序、循环和选择三种基本程序结构，非必要时不要使用 GO TO 语句. 这样能明显改进软件的可读性、可测试性和可维护性.

软件测试是软件开发中最容易引起误解、最麻烦、最不可预测和最费钱的阶段. 软件测试的任务是：

（1）通过测定结构化的接口和程序中的元素来验证软件设计；

（2）软件需求的确认；

（3）提供一套系统化的软件装配方法；

（4）制订质量基准，以便于维护的跟踪检验.

所有测试任务的制定都是为了要达到一个主要目的，即引起软件可能的失败，以便发现错误. 换句话说，一个成功的测试应能找出潜在的错误，发现不出任何错误的测试不是好的测试，而人们常常在这一点上产生误解.

对于典型的软件项目，测试往往要占开发工作量的 40－50%，而对于一个纯系人工开发的软件系统，测试可占其全部开发工作量的 80%.

实际上初步设计评审、详细设计评审和代码预查都可以算作软件测试. 这些人工的、实现前的测试可以找出软件中占有很大比例的错误. 而且，采用这些前期测试（如评审）其代价比基于计算机的测试代价低得多.

一旦完成了代码的预查和认可，就可进行模块的单元测试. 单元测试可以验证模块的输入和输出，还可以查明模块功能的有效性以及是否反映了设计的要求. 成功地进行了模块单元测试之后，就可进行集成测试，以使每个模块结合成整个软件系统. 集成测试需完成两项任务：

(1) 提供组装软件的方法；

(2) 在组装的每一步中，检查新装入的模块引起的错误.

测试阶段的最后一步是验证组装起来的软件系统是否都能满足软件需求规格说明书所提出的要求. 这一步称为验收测试.

软件交付运行使用后，其生存期并未终止，而是进入了一个新的重要阶段——运行与维护阶段. 人们常常忽视这一阶段的重要性. 事实上，软件投入运行之后，还可能存在大量潜伏的问题. 对于一个大型软件系统，不可能设想通过测试会把它们全揭露出来，在使用期间会不断发现错误，我们把这种诊断和校正错误的活动叫做校正性维护. 另外，由于硬件的更换，操作系统版本的更新，以及外围设备的补充与更新等都必然要求软件系统作相应的修改. 这种维护活动叫做适应性维护. 当用户使用软件以后，可能还会提出增加新的功能，修改现有功能及扩大通用性之类的建议，为此需要进行完善性维护. 据统计现有软件的维护将占一个开发机构全部工作量的60%以上. 这个比例将随着生产出更多的软件而持续上升. 这样发展下去，在将来的某一天，软件开发机构势必被束缚在旧软件的维护上，以致无法再生产新的软件. 有人把这种现象称为“软件墙”，这的确是需要认真对待的问题.

以上，我们简单介绍了程序、软件及软件生存期等软件工程中的基本概念.

软件工程涉及的范围大体上可以划分为方法、工具（包括环境）和管理这三个方面. 下面，我们对它们作一概述.

1.3 软件工程的主要内容

一项软件产品的诞生必然要涉及到人，这包括用户和软件开发单位两方面，每一方都是由许多人构成的. 每人对问题的理解可能不同，对问题的处理方式也可能不同；在漫长的软件开发期内，即使是同一个人，在不同时期，对问题的理解和处理还有可能发生变化. 而且，随着计算技术的发展，处理问题的选择余地又更加扩大了. 在软件开发过程中，缺乏强有力的内部纪律，各行其是，则是造成软件危机的主要原因.

为了使软件研制走上工程化的轨道，我们必须寻找一些标准的规程，以便为开发人员给出指导和约束，使他们遵照一定的方式来理解和处理问题，这是使开发获得成功的关键.

软件工程方法就是指导研制软件的某种标准规程，它告诉人们什么时候做什么和怎样做. 由于软件研制过程的复杂性，不存在一种“包治百病”的软件工程方法，各种方法都有自己的侧重面，分别适用于不同的具体情况. 通常一个软件工程方法要对以下三个方面作出规定：

(1) 明确的工作步骤

研制一个软件系统要考虑并解决许多问题，如果同时处理这些问题，将会顾此失彼，应接不暇，或者造成混乱. 正确的方式是将这些问题分成先后次序，每一步集中精力解决一个问题. 就像为加工机械产品规定一道道工序那样，软件工程方法提出了处理问题的基本步骤，这包括每一步的目的是什么，每一步应产生什么结果，每一步需具备的条件以及注意事项等.

(2) 具体的文档格式

此为试读，需要完整PDF请访问：www.ertongbook.com

工程化必须强调文档化，即每人必须将每一步的工作结果以一定的书面形式记录下来，这将保证开发人员之间有效地进行交流，方便地追溯软件故障产生的根源，也有利于维护工作的顺利进行，文档化也是研究软件工具的前提．软件工程方法规定了描述软件产品的文档格式，文档格式包括每步应产生什么文档，文档中应记录哪些内容，采用哪些图形、符号等．

（3）确定的评价标准

对于同一个问题，其解决方案往往不是唯一的，选取哪一个方案好呢？有些软件工程方法提出了比较确定的评价标准，因而可以指导人们对各个具体方案作出比较评价．

近十几年来，软件工作者陆续研究总结出多种软件工程方法．70年代初，出现了编写程序的一些方法；70年代中期，人们认识到编写程序仅仅是软件开发的一个环节，而合理地建立系统的结构比编写程序更重要，所以重点前移到设计阶段，出现了设计阶段的软件工程方法；70年代后期，人们又进一步认识到在设计阶段之前还应当对用户的要求进行分析，因此，重点又转到分析阶段，于是又出现了分析阶段的软件工程方法．目前，除了运行与维护阶段以外，对于其它几个阶段都有各种不同的软件工程方法可供使用．而在运行与维护阶段，尚未形成成熟的方法．

本书第二章到第七章分别介绍了生存期各阶段中常用的软件工程方法，了解并掌握这些方法的基本思想和特征，结合具体情况加以灵活运用，这就是学习软件工程方法的目的．

下面介绍一下软件工具和软件环境．工具是人们很熟悉的概念．一提到工具，大家就会想到刀子、斧头、螺丝起子等等．工具是人类的帮手，它帮助人类扩大能力，提高工作效率．所谓软件工具，其实是一种特殊目的程序系统，用来帮助人们高效率地开发软件．和计算机打过交道的人都知道，要使计算机工作，必须按要求编制相应的程序，而程序是通过键盘一个字符一个字符输入进去的，在键入一个很长的程序时，免不了会出错，这就要进行修改，通常，在编辑状态下可以对程序进行修改，还可以查询、补充、删除．当处于编辑状态时，实际上是在运行编辑程序，而编辑程序就是一种软件工具，它帮助人们提高编制程序的效率．编辑程序是在编码阶段使用的软件工具之一．在其它阶段也有各自的软件工具．例如，在需求分析阶段有需求表达工具、规格描述工具；在设计阶段有各种辅助设计工具；在测试阶段有测试数据产生器、测试分析器等工具．随着计算机软件的商品化，为了高效率高质量地生产软件，各种不同类型、不同用途的软件工具如雨后春笋般涌现．许多通用性强、性能好的软件工具本身也商品化了，人们已可以在市场上买到．

前面我们已经说过，对应于生存期的不同阶段有不同的软件工具．人们自然会想到，既然一个软件的开发要贯彻软件生存周期的始终，那么，我们何不把各阶段的软件工具串联起来，并使它们互相衔接匹配，从而全面地提高软件生产效率呢？这种想法是对的，而且这一系列配套软件工具还有一个专门的名字——软件开发环境（或简称为软件环境）．软件环境具有下列特点：

（1）连续性　能支撑从需求分析、设计、实现到维护的各个阶段，换句话说，就是能实现生存期的支撑．

（2）综合性　能支撑软件的开发、维护、管理和质量控制各个方面．

（3）广泛性　从各方面向软件人员（任务组）完成软件任务提供全面的、有力的支撑．

(4) 友善性　必须提供有力的、直接的、适宜的支撑. 不应强制用户去适应环境，而应使环境适应用户（如使用“菜单”方式，多“窗口”功能等).

(5) 紧密性　一个环境中各工具必须能密切地配合工作.

(6) 坚定性　一个环境可以保护自己不受来自用户与系统错误的影响，可以从意外中恢复，可对用户提供有效的诊断信息.

(7) 可适应性　必须能灵活调节以满足不同的、经常变化的用户要求，环境中各工具可以随时修改，随时扩充功能，工具可以增减.

(8) 可移植性　应能较方便地移植到不同类型的计算机上.

本书第八章将对软件工程的发展方向作一简介.

在现代社会中，管理起着越来越重要的作用. 在软件开发中，情形也一样. 当软件的规模增大时，投入开发的人力财力都要增多. 显然，必须加强对人财的管理. 而且不像物质产品的生产，软件的生产在其前期只存在于人脑中，是不可见的. 这种不可见性更增加了软件开发管理的复杂性. 为了使软件生产的全过程可见，可被跟踪，要求在生存期各阶段产生相应的文档. 通过这些文档，对软件开发实现有效的管理.

以上我们对软件工程方法、软件工具（环境）及软件开发管理的概念作了介绍. 需要强调的是，这三个主要内容不是彼此无关的. 软件工程方法为软件工具提供理论基础，软件工具是软件工程方法的体现，而软件开发管理是软件工程方法和软件工具得以有效使用和运行的保证. 三者紧密配合，协调运用才能高效优质地生产软件.

第二章　SA/SD 方法

SA（Structured Analysis）方法是需求分析阶段中使用的方法. SD（Structured Design）方法是设计阶段中使用的方法. 这两种方法常常衔接使用，它们在美国较为流行. 在本章中，我们先介绍 SA 方法，然后介绍 SD 方法.

在 SA 中，使用下列工具：

- 数据流程图
- 数据字典
- 结构化英语
- 判定表
- 判定树

数据流程图是用图示的方法表示数据处理系统的功能分解及各成分间的界面. 数据字典是用于对系统中涉及的数据进行定义和描述. 结构化英语、判断表和判断树是以严格的方式刻划数据流程图中的各个处理环节.

利用这些工具可以形象而准确地描述数据处理系统，以此作为系统的规格说明. 下面，我们逐一介绍这些工具.

2.1　数据流程图

我们可以把数据处理系统想象为数据加工厂. 从数据发源地流出的数据进入系统，在系统内部的不同“车间”被加工，有的被保存起来，有的以“成品”形式流向数据的终点站. 利用数据流程图可以把这些过程形象地表示出来. 数据流程图由四个基本要素组成. 它们是：数据流（用箭头表示），文件（用直线段表示），加工（用圆表示），数据源点或终点（用方框表示），见图 2.1.

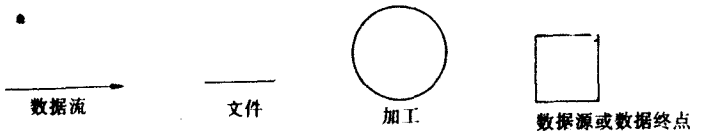


图 2.1

图 2.2 是一个简单的数据流程图. 在该图中，从数据源流出两股数据流：数据流 1 和数据流 2，数据流 2 经过加工 2 加工后转变为数据流 3，加工 1 对数据流 1 和数据流 3 进行加工，产生数据流 4 和数据流 5，前者流入数据终点，而后者则被保存在文件中.

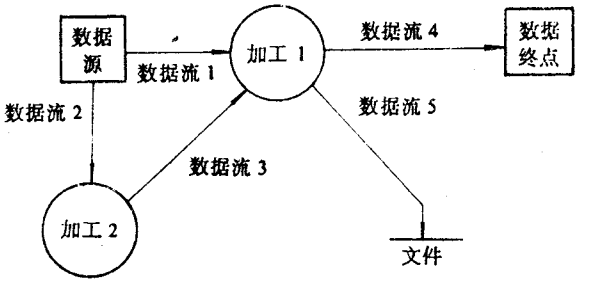


图 2.2

实际的数据处理系统要远远比上面这个例子复杂得多. 显然，如果我们巨细不分地把它们用数据流程图画出来，势必构成一幅高度错综复杂的图形，甚至无法看明白. 为了解决这个问题，我们采用分层的方法. 也就是说，我们从宏观开始，首先勾画出系统的轮廓，然后一层一层地向微观方向发展，并越来越细致地描述各个细节. 通过这一套分层的数据流图来完整而细致地描述系统.

图 2.3 是一个分层的数据流程图的例子. 在图 2.3 (a) 中，系统被看作是一个加工 X，它接受一个数据流，输出两个数据流. 图 2.3 (b) 是它的下一层，在这一层中 X 被分解

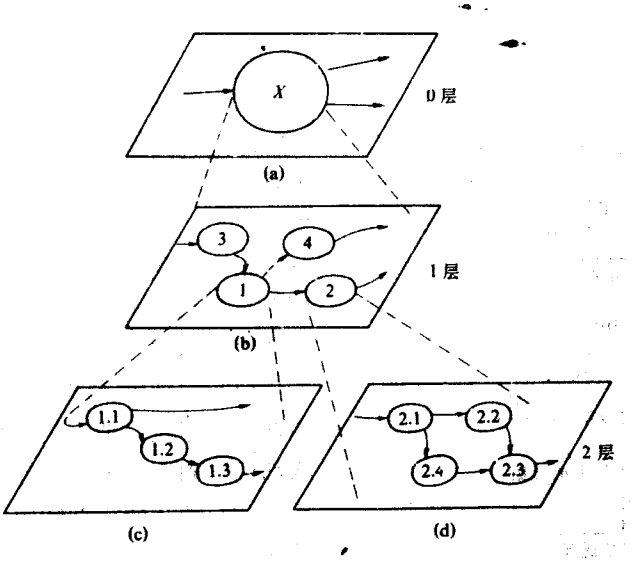


图 2.3

为 4 个加工. 在图 2.3 (c)，(d) 中，加工 1 和加工 2 被进一步分解.

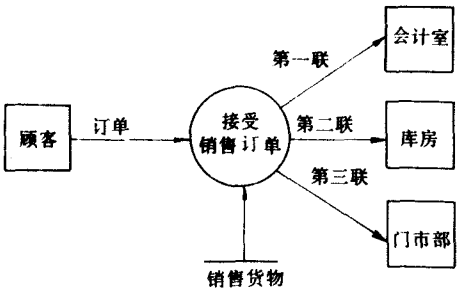
在绘制分层数据流程图时，要注意到数据流的平衡，也就是说，流入的数据和流出的数据必须与上一层相一致. 例如，在图 2.3 的 0 层，输入数据流有一条，输出数据流有两条，在 1 层，尽管系统内部增加了数据流，但输入的数据依流仍然只有一条，输出的数据

流仍为两条，0层与1层是平衡的．读者可以检验一下，1层与2层数据流的平衡性．

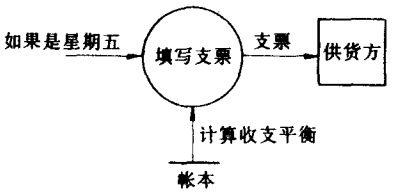
在绘制层次数据流程图时，各层的编号宜采用加点表示法，例如．1层为1，2，3，…，2层为1.1，1.2，1.3，…，2.1，2.2，2.3，…，3.1，3.2，….这样通过编号就可以明显地看出层次性．

值得注意的是，数据流程图中不应包括具体的控制信息．图2.4中给出了几种常见的错误，在图2.4（a）中，订单的第一联，第二联，第三联都是票据的编号而不是数据流的名称；在图2.4（b）中，“如果是星期五”也不是数据流的名称而是加工的触发条件（在这里是时间），在图2.4（c）中，用“填金额”，“取下一个发货通知”等过程性描述代替了数据流名称．

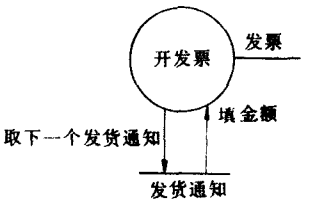
数据流的命名必须具有一定含义，它应当反映数据的特征，不能用数据所驻留的文件的名称作为数据流的名称．也不能把传递数据的介质作为数据流的名称．例如，一张发票上有发票号、开票日期、开票人等数据，它们连同发票一起流动，尽管如此，发票名称不应作为数据流名称．因为，一旦建立了计算机化的数据系统，发票的物质形式也许就会消失，而发票号、开票日期等数据依然存在．就是说，数据的传递介质会改变，而数据却是长存的．



(a)



(b)



(c)

图2.4

加工的命名必须能刻画加工活动的本质．例如，把“盘存”、“付款”、“销售”等含义较一般的概念作为加工的名字就不如用“清点存货数量”、“填写发货单”、“开发票”等具有较明确含义的概念作为加工命名来得恰当．

以上，我们叙述了数据流程图的一些性概念．那么，下一步究竟应该怎样画数据流程图呢？通常，按下列步骤进行：

（1）识别出系统所有的纯输入数据流和纯输出数据流，把它们画在数据流程图的外缘部位以确定系统的边界．

（2）按从输入到输出，或从输出倒推到输入，或“中心开花”的顺序考察系统，凡有数据流转换处画上加工圆圈．

（3）对系统内部数据流进行命名．

（4）对加工命名．

（5）消除初始化和终止条件．

（6）略去出错数据流的细化．

（7）消除控制流或控制信息．

(8) 反复检查与修改数据流程图.

现在，对上述步骤作一些说明. 在步骤 (1) 中，我们要做的事实际上是确定系统的边界，就是说，哪些数据是系统应该处理的，哪些不是系统处理的. 我们可以大致上把数据分成三类：第一类是系统外生成而由系统进行加工的数据，即系统的纯输入数据；第二类是系统内生成又由系统进行加工和保留的数据；第三类是系统内生成和加工但不保留在系统内的数据，即系统的纯输出数据.

第二步是考察系统的工作，凡是需要把一个（或一组）数据转换成另外一个（或一组）数据的地方，都要画上一个空心圆，以表示这里有一个加工. 值得注意的是，这样画出来的图只能算是草图，还必须把它交给用户审查. 如果一个空心圆内部存在数据流，那就说明原先的加工还要进一步分成子加工，然后根据用户的实际情况或要求画上文件，这时必须准确地知道流入或流出文件的数据流.

第三步是对数据流命名，这里要注意的是，命名要准确可靠不要重名，也不要把不相干的数据拉在一起取一个长长的名字. 如果发生这种情况，就说明数据流程图的划分有问题，必须重新进行划分.

第四步是对加工命名. 按照我们的步骤，先对数据流命名后对加工命名，这是符合自顶向下原则的，否则就会造成麻烦. 让我们来看图 2.5 所示的例子. 在图 2.5 (a) 中，加

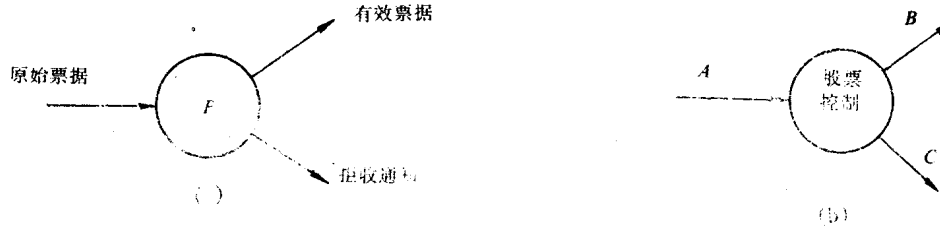


图 2.5

工 P 的输入数据流为原始票据，输出数据流为有效票据和拒收通知（对无效票据）. 根据这些数据流的名字，我们很容易地把加工 P 命名为票据整理. 但是在图 2.5 (b) 中加工已先被命名为股票控制，从这个命名中，我们很难对其输入输出数据流 A，B 和 C 赋以名称. 对加工的命名要求除了准确可靠以外，还应当符合谓宾结构，即每个加工应当只有一个动作（由一个动词表示）和一个动作对象（由一个宾语表示）.

第五步是消除初始化和终止条件. 数据流所描写的是系统进入正常运行后的情况，因此有关系统如何起动和终止的条件，应不在考虑之列. 它们应当从数据流程图中消去.

第六步是略去出错数据流的细化. 当发现了出错数据流时，不必再进一步分析出错数据流的处理.

第七步是消除控制流或控制信息. 习惯于画流程图的人，往往把控制流与数据流混为一谈. 为了将二者明确地区分开来，可以考察其用途. 如果该信息本身需要改变或处理，则为标准的数据流，如果是为了触发或控制某个过程，则为控制流.

第八步的目的和必要性是不言而喻的，在此不必多作解释.

此为试读，需要完整PDF请访问：www.ertongbook.com