● 高等学校教材

软件工程——原理、方法与应用

（第二版）

史济民　顾春华　李昌武　苑　荣　编著

高等教育出版社
HIGHER EDUCATION PRESS

高等学校教材

软件工程

——原理、方法与应用

（第二版）

史济民　顾春华　李昌武　苑荣　编著

高等教育出版社

内容提要

本书自第一版出版以来，由于内容全面（以软件开发技术为主体，兼顾软件工程管理和软件工程环境）、注重实用（理论与实践紧密结合）而受到读者欢迎。第一版连续发行12年，重印15次，累计印数近12万册，并于1995年获得上海市优秀教材二等奖。

第二版继承前一版的风格，增加了面向对象、软件复用等大量新内容。全书共15章，在介绍了软件工程的基本概念和软件开发模型后，按照软件开发流程的顺序，依次介绍了需求分析、系统设计、编码、测试、维护的基本概念以及软件管理、质量保证和工程环境等知识。为了说明怎样把软件工程的原理与方法应用于软件开发，在有关各章均有典型案例，从头到尾陆续讲述了一个软件的完整开发过程。

本书可作为高等学校计算机及相关专业本、专科学生软件工程课程的教材，对从事软件开发、软件维护的工程和管理人员也是一本很好的参考书。

图书在版编目(CIP)数据

软件工程/史济民等编著. —2版. —北京：高等教育出版社，2002.12
ISBN 7-04-011561-1

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(2002)第059282号

软件工程——原理、方法与应用(第二版)
史济民 顾春华 李昌武 苑荣 编著

出版发行	高等教育出版社	购书热线	010-64054588
社　　址	北京市东城区沙滩后街55号	免费咨询	800-810-0598
邮政编码	100009	网　　址	http://www.hep.edu.cn
传　　真	010-64014048		http://www.hep.com.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	北京二二〇七工厂		
		版　　次	1990年5月第1版
开　　本	787×1092　1/16		2002年12月第2版
印　　张	22	印　　次	2002年12月第1次印刷
字　　数	470 000	定　　价	23.50元

再版前言

本书第一版自1990年问世以来，已经过了12年。在此期间，软件工程从第一代传统的软件工程发展为第二代面向对象的软件工程，现正向基于软件复用的第三代软件工程发展。在高等学校中，软件工程不仅已成为计算机专业学生的必修课，而且在非计算机专业的“软件技术基础”等公共课中也上升为必学的内容。

作为把软件工程推广到高校非计算机专业的一种尝试，本书第一版曾被收入高等教育出版社的“高校计算机基础教育系列教材”，但实际上多数高校仍把它用作计算机专业的教材。为了适应国内软件工程当前教学的需要，这次改版将本书仍定位为计算机专业的本科教材，并且确定了如下的编写方针。

一、继续保持“注重实践”的风格

软件工程具有很强的实践性，但由于例题难选，多数教材往往偏重于理论。本书第一版参考国际知名教材Pressman著的《软件工程：实践者的方法》的做法，广举实例，注重实践，因而受到读者的欢迎。第一版连续发行12年，累计印数近12万册，并于1995年荣获上海市高校优秀教材二等奖。本版保持了上述风格，使之名副其实地成为原理、方法与应用紧密结合的教材。

二、平行讲解第一、二两代软件工程

经过20世纪90年代的发展，面向对象软件工程已渐趋成熟，在许多应用领域取代了传统的软件工程。但是，传统的、基于结构化程序设计的软件工程并未退出历史舞台，其基本原理与方法有不少仍在新一代软件工程中继续应用。为此，本书将两类软件工程平行讲解，既方便读者对照，又便于展示两者“你中有我、我中有你”的关系。

三、充分反映软件工程近十几年的发展

本次改版增加了许多新方法、新模型的介绍，例如面向对象软件工程的UML语言、软件复用、质量认证标准、净室工程和软件容错技术等。对于代表新的发展方向的软件构件工程、软件过程工程等新技术，均设置专章或专节进行讲解，以突出其重要性。但鉴于本书读者对象主要是高校的本、专科学生，对于还处于形成阶段的有些新技术，本教材着重阐明它们的基本概念、产生背景和主要作用，不展开细节的讨论。

四、重点讲解软件工程技术，兼顾管理与环境

作为软件工程的综合性入门教材，本版同第一版一样主要讲解软件工程技术，兼讲一点管理与环境方面的知识。后者的重点放在质量管理与I-CASE环境上，以便读者对软件工程的整体情况与近期发展有一比较全面的理解。

本书由史济民、顾春华共同策划。第一章由史济民改写；第二、三、四、十一、十二、十五章由顾春华改写；第五、八、九、十三章由李昌武改写。新增加的4章，分别由顾春华（第六、七章及第14.5、14.6节）、苑荣（第十章）和史济民（第十四章其余各节）编写。全书由史济民统一修改定稿。华东理工大学宋国新教授十分支持本次改版，除承担书稿审阅外，还在经费和编写人员时间安排上给予帮助。对此编者表示由衷的感谢。

由于软件工程覆盖面宽，发展又很迅速，编者水平有限，疏漏难免，诚恳希望读者不吝指正。

编 者

2002年5月于上海

目　录

第一章　绪论 …… 1
1.1　软件与软件危机 …… 1
1.1.1　软件的定义 …… 1
1.1.2　软件的特征 …… 2
1.1.3　软件危机 …… 2
1.2　软件工程学的范畴 …… 4
1.2.1　软件开发方法学 …… 5
1.2.2　软件工具 …… 5
1.2.3　软件工程环境 …… 6
1.2.4　软件工程管理 …… 6
1.3　传统软件工程和面向对象软件工程 …… 6
1.3.1　程序设计方法的两次飞跃 …… 7
1.3.2　面向对象程序设计的优势 …… 7
1.3.3　两类软件工程范型的简单比较 … 8
1.4　软件工程的应用 …… 9
1.4.1　在各种规模软件开发中的应用 … 9
1.4.2　软件工程的成就与局限 …… 10
1.5　软件工程的教学：本书导读 …… 11
小结 …… 13
习题 …… 13
第二章　软件开发模型 …… 15
2.1　软件生存周期 …… 15
2.2　传统软件开发模型 …… 17
2.2.1　瀑布模型 …… 18
2.2.2　快速原型模型 …… 19
2.3　软件演化模型 …… 20
2.3.1　增量模型 …… 20
2.3.2　螺旋模型 …… 21
2.4　面向对象开发模型 …… 22
2.4.1　面向对象的基本概念 …… 23
2.4.2　构件集成模型 …… 24
2.5　形式化方法模型 …… 25
2.5.1　转换模型 …… 25
2.5.2　净室模型 …… 26
2.6　开发模型选用实例 …… 27
2.6.1　瀑布模型实例：教材购销系统（面向过程软件） …… 27
2.6.2　螺旋模型实例：显像管生产监测系统（面向对象软件） …… 28
小结 …… 29
习题 …… 29
第三章　软件需求分析 …… 30
3.1　需求分析的任务与步骤 …… 30
3.1.1　需求分析的任务 …… 30
3.1.2　需求分析的步骤 …… 31
3.2　需求获取的常用方法 …… 33
3.2.1　常规的需求获取方法 …… 34
3.2.2　快速原型法在需求分析中的应用 …… 35
3.3　分析建模 …… 36
3.3.1　两种分析模型 …… 36
3.3.2　分析模型的组成与描述工具 …… 38
3.4　软件需求说明 …… 53
3.5　结构化分析方法 …… 54
3.5.1　画分层数据流图 …… 54
3.5.2　确定数据定义与加工策略 …… 57
3.5.3　需求分析的复审 …… 58

3.6 面向对象分析方法 …… 60
3.6.1 定义用例 …… 60
3.6.2 领域分析 …… 62
3.6.3 类/对象建模 …… 63
3.6.4 建立对象－关系模型 …… 66
3.6.5 建立对象－行为模型 …… 67
小结 …… 68
习题 …… 68
第四章 软件设计概述 …… 70
4.1 软件设计的任务 …… 70
4.2 软件设计的基本概念 …… 71
4.2.1 模块与构件 …… 71
4.2.2 抽象与细化 …… 71
4.2.3 信息隐藏 …… 72
4.2.4 软件复用 …… 73
4.3 模块化设计 …… 73
4.3.1 分解 …… 73
4.3.2 模块独立性 …… 74
4.3.3 自顶向下与由底向上设计 …… 78
4.4 其他设计问题的处理 …… 79
4.4.1 协同设计 …… 79
4.4.2 用户界面设计 …… 79
4.4.3 并发系统设计 …… 80
4.5 设计文档及其复审 …… 81
4.5.1 软件设计说明书 …… 81
4.5.2 设计复审 …… 81
小结 …… 84
习题 …… 85
第五章 传统的设计方法 …… 86
5.1 概述 …… 86
5.1.1 面向数据流设计和面向数据设计 …… 86
5.1.2 从分析模型导出设计模型 …… 87
5.2 结构化设计方法 …… 88
5.2.1 SC图 …… 88
5.2.2 数据流图的类型与SD方法的步骤 …… 89
5.2.3 变换映射 …… 91
5.2.4 事务映射 …… 95
5.2.5 结构设计的优化规则 …… 97
5.2.6 教材购销系统的结构设计示例 …… 100
5.3 过程设计 …… 105
5.3.1 目的与任务 …… 105
5.3.2 过程设计的原则与方法 …… 106
5.3.3 常用的表达工具 …… 109
5.3.4 过程设计示例 …… 112
5.4 Jackson方法 …… 116
5.4.1 Jackson表示法 …… 116
5.4.2 Jackson方法的设计步骤 …… 117
5.4.3 Jackson方法示例 …… 118
小结 …… 122
习题 …… 122
第六章 面向对象设计方法 …… 125
6.1 面向对象设计概述 …… 125
6.1.1 面向对象设计的任务 …… 125
6.1.2 面向对象的设计模型 …… 126
6.2 系统设计 …… 127
6.2.1 系统设计过程 …… 127
6.2.2 子系统设计 …… 127
6.2.3 人机交互设计 …… 130
6.2.4 任务管理设计 …… 131
6.2.5 数据管理设计 …… 132
6.3 对象设计 …… 133
6.3.1 对象描述 …… 134
6.3.2 算法设计 …… 136
6.3.3 程序构件与接口 …… 137
6.4 领域对象设计 …… 138
6.4.1 领域对象的设计内容 …… 138
6.4.2 领域对象的设计模板 …… 140

小结…… 141
习题…… 142
第七章 统一建模语言 UML …… 143
7.1 UML 的组成、特点与应用 …… 143
7.1.1 UML 的组成 …… 144
7.1.2 UML 的特点 …… 148
7.1.3 UML 的应用 …… 148
7.2 静态建模 …… 149
7.2.1 用例模型 …… 149
7.2.2 类和对象模型 …… 150
7.2.3 包 …… 158
7.3 动态建模 …… 158
7.3.1 消息 …… 158
7.3.2 状态图和时序图 …… 159
7.3.3 协作图和活动图 …… 162
7.3.4 动态图的运用 …… 164
7.4 物理架构建模 …… 165
7.4.1 逻辑架构与物理架构 …… 165
7.4.2 构件图与配置图 …… 167
7.5 基于 UML 的统一建模过程——RUP …… 167
小结…… 169
习题…… 169
第八章 编码和语言选择 …… 171
8.1 编码的目的 …… 171
8.2 编码的风格 …… 172
8.3 编码使用的语言 …… 180
8.3.1 编码语言的发展 …… 180
8.3.2 常用的编码语言 …… 182
8.3.3 编码语言的选择 …… 185
小结…… 186
习题…… 187
第九章 软件测试 …… 188
9.1 测试的基本概念 …… 188
9.1.1 目的与任务 …… 188
9.1.2 测试的特性 …… 189
9.1.3 测试的种类 …… 190
9.1.4 测试的文档 …… 191
9.2 黑盒测试 …… 192
9.2.1 等价分类法 …… 192
9.2.2 边界值分析法 …… 193
9.2.3 错误猜测法 …… 195
9.3 白盒测试 …… 195
9.3.1 逻辑覆盖测试法 …… 196
9.3.2 路径测试法 …… 200
9.4 测试用例设计 …… 205
9.4.1 黑盒测试用例设计 …… 205
9.4.2 白盒测试用例设计 …… 207
9.5 软件的纠错 …… 210
9.5.1 纠错的策略 …… 210
9.5.2 常用的纠错技术 …… 212
9.5.3 纠错举例 …… 215
9.6 多模块程序的测试策略 …… 221
9.6.1 测试的层次性 …… 221
9.6.2 程序错误的类型 …… 222
9.6.3 单元测试 …… 225
9.6.4 集成测试 …… 226
9.6.5 确认测试 …… 229
9.6.6 系统测试 …… 230
9.6.7 终止测试的标准 …… 230
9.7 面向对象系统的测试 …… 230
9.7.1 面向对象软件的测试策略 …… 231
9.7.2 面向对象软件测试用例设计 … 232
小结…… 235
习题…… 235
第十章 软件复用 …… 238
10.1 软件复用的基本概念 …… 238
10.1.1 软件复用的定义 …… 238
10.1.2 软件复用的重要性 …… 239
10.1.3 软件复用的粒度 …… 240

10.2 领域工程 …… 241
10.2.1 领域工程的活动内容 …… 241
10.2.2 实施领域分析 …… 242
10.2.3 开发可复用构件 …… 242
10.2.4 建立可复用构件库 …… 245
10.3 基于构件的软件开发 …… 246
10.3.1 构件集成的过程 …… 246
10.3.2 应用系统工程 …… 247
10.4 面向对象与软件复用 …… 247
10.4.1 面向对象方法对软件复用的支持 …… 248
10.4.2 复用技术对面向对象方法的支持 …… 249
小结 …… 251
习题 …… 251
第十一章 软件维护 …… 252
11.1 软件维护的种类 …… 252
11.2 软件可维护性 …… 253
11.3 软件维护的实施 …… 256
11.4 软件维护的管理 …… 258
11.5 软件再工程 …… 260
小结 …… 262
习题 …… 262
第十二章 软件项目计划 …… 264
12.1 问题定义 …… 265
12.2 可行性研究 …… 266
12.3 软件风险分析 …… 270
12.4 项目实施计划 …… 273
小结 …… 274
习题 …… 275
第十三章 软件工程管理 …… 276
13.1 管理的目的与内容 …… 276
13.2 软件估算模型 …… 277
13.2.1 资源估算模型 …… 277
13.2.2 COCOMO 模型 …… 279
13.3 软件成本估计 …… 281
13.4 人员的分配与组织 …… 287
13.5 项目进度安排 …… 290
13.6 软件知识产权保护 …… 294
13.6.1 软件著作权 …… 294
13.6.2 软件侵权及法律保护 …… 295
13.6.3 软件工程师的职业道德规范 …… 296
小结 …… 299
习题 …… 300
第十四章 软件质量管理 …… 301
14.1 从质量保证到质量认证 …… 301
14.2 质量保证 …… 302
14.2.1 软件的质量属性 …… 302
14.2.2 质量保证的活动内容 …… 303
14.3 软件可靠性 …… 304
14.3.1 可靠性的定义和分级 …… 304
14.3.2 可靠性模型 …… 306
14.3.3 软件容错技术 …… 308
14.4 程序正确性证明 …… 311
14.5 CMM 软件能力成熟度模型 …… 313
14.5.1 CMM 的基本概念 …… 313
14.5.2 软件能力成熟度等级 …… 314
14.5.3 CMM 的应用 …… 315
14.5.4 CMM 评估的实施 …… 316
14.5.5 软件过程评估的 SPICE 国际标准 …… 316
14.6 ISO 9000 国际标准 …… 317
14.6.1 ISO 9001 和 ISO 9000－3 …… 317
14.6.2 ISO 9000 标准对软件企业的重要性 …… 318
14.6.3 怎样在软件企业中实施 ISO 9000 标准 …… 318
14.7 软件度量 …… 319

14.7.1　项目度量 …… 319
14.7.2　过程度量 …… 321
小结 …… 322
习题 …… 323
第十五章　软件工程环境 …… 324
15.1　什么是软件工程环境 …… 324
15.1.1　软件开发环境的特点 …… 324
15.1.2　理想环境的模型 …… 326
15.1.3　CASE 环境 …… 327
15.2　CASE 环境的组成与结构 …… 327
15.2.1　CASE 的组成构件 …… 327
15.2.2　CASE 的一般结构 …… 329
15.3　CASE 环境实例 …… 331
15.3.1　SUITE 企业开发环境 …… 331
15.3.2　青鸟系统 …… 336
小结 …… 337
习题 …… 337
附录　缩略语中英文对照表 …… 338
参考文献 …… 340

第一章 绪 论

1969 年，美国 IBM 公司首次宣布除操作系统继续随计算机配送外，其余软件一律计价出售，从此开创了软件成为独立商品的先河。短短 30 余年间，计算机软件的重要性与日俱增。从 PC 机到笔记本电脑，从因特网到移动电话，从先进的武器到现代的家电，计算机软件几乎无处不在，无时不在。世界上最大的软件公司微软公司及其创始人，已成为全球知名度最高的企业和人物之一。在很多发达国家，软件产业已成为社会的支柱产业，软件工程师也成为最受青睐的一种职业。

正是由于软件的发展，使计算机应用逐步渗透到社会生活的各个角落，使各行各业都发生很大的变化。这同时也促使人们对软件的品种、数量、功能和质量等提出了越来越高的要求。然而，软件的规模越大、越复杂，人们的软件开发能力越显得力不从心。于是，人们开始重视软件开发过程、方法、工具和环境的研究，软件工程应运而生。

本章介绍软件和软件工程的基本概念，包括软件、软件危机、软件工程学、传统软件工程与面向对象软件工程以及它们的应用等。章末讨论了与软件工程教学有关的几个观点，也可视为本书的导读。

1.1 软件与软件危机

和计算机硬件一样，从 20 世纪 60 年代以来，软件也从规模、功能等方面得到了很大的发展，同时人们对软件质量的要求也越来越高。那么，究竟什么是软件，软件有哪些特征呢？

1.1.1 软件的定义

众所周知，计算机硬件的发展基本上遵循了摩尔定律，即每 18 个月芯片的性能与速度均提高一倍。软件的发展也十分惊人：例如就体系结构而言，它经历了从主机结构到文件服务器结构，从客户/服务器系统到基于 Internet 的服务器/浏览器结构等变化；从编码语言来讲，它经历了从机器代码到汇编代码，从高级程序语言到人工智能语言等变化；从开发工具来看，它经历了从分离的开发工具（如代码编辑器，中间代码生成器和连接器）到集成的可视化开发系统，从简单的命令行调试器到方便的多功能的调试器等变化；等等。

但是在过去 30 余年中，软件的基本定义却并未改变。有些初学者认为软件就是程序，这个理解是不完全的。这里引用著名的美国软件工程教材作者 R. S. Pressman 的定义："软

件是能够完成预定功能和性能的可执行的计算机程序和使程序正常执行所需要的数据，加上描述程序的操作和使用的文档。"简明地表述，可以写作"软件 = 程序 + 文档"。

程序是为了解决某个特定问题而用程序设计语言描述的适合计算机处理的语句序列。它是由软件开发人员设计和编码的，通常要经过编译程序，才能编译成计算机可执行的机器语言指令序列。程序执行时一般要输入一定的数据，也会输出运行的结果。而文档则是软件开发活动的记录，主要供人们阅读，既可用于专业人员和用户之间的通信和交流，也可以用于软件开发过程的管理和运行阶段的维护。为了提高软件开发的效率和方便软件产品的维护，现在软件人员越来越重视文档的作用及其标准化工作。我国国家标准局已参照国际标准，陆续颁布了《计算机软件开发规范》、《计算机软件需求说明编制指南》、《计算机软件测试文件编制规范》和《计算机软件配置管理计划规范》等文档规范。

1.1.2 软件的特征

要对软件有一个全面的理解，首先要了解软件的特征。当生产硬件时，生产的结果能转换成物理的形式。如果要建造一台新的计算机，从设计图纸、生产部件（VLSI 芯片、线路板、面板等）到装配原型，每一步都演化成物理的产品。而软件却是逻辑的而不是物理的，在开发、生产、维护和使用等方面，都具有与硬件完全不同的特征。

1. 软件开发不同于硬件设计

与硬件设计相比，软件更依赖于开发人员的业务素质、智力，以及人员的组织、合作和管理，而硬件设计与人的关系要相对小一些。对硬件而言，设计成本往往只占整个产品成本的一小部分，而软件开发的成本很难估算，通常占整个产品成本的大部分，这意味着软件开发项目不能像硬件设计项目那样来管理。

2. 软件生产不同于硬件制造

硬件设计完成后就投入批量制造，制造也是一个复杂的过程，其间仍可能引入质量问题；而软件成为产品之后，其制造只是简单的拷贝而已。

3. 软件维护不同于硬件维修

硬件在运行的初期有较高的故障率（主要来源于设计或制造的缺陷），在缺陷修正后的一段时间中，故障率会降到一个较低和稳定的水平上。随着时间的改变，故障率将再次升高，这是因为硬件会受到磨损等损害，达到一定程度后就只能报废。软件是逻辑的而不是物理的，虽然不会磨损和老化，但在使用过程中的维护却比硬件复杂得多。如果软件内部的逻辑关系比较复杂，在维护过程中还可能产生新的错误。

1.1.3 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达 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 元用于计算机投资总额中，软件将花费 90 元。这一预计早已为实践所证实。

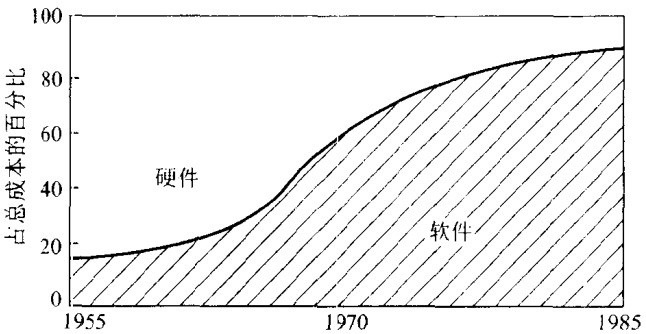

图 1.1 硬件/软件成本变化趋势

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。

以下再从维护和生产两个方面，进一步说明出现软件危机的原因。

（1）软件维护费用急剧上升，直接威胁计算机应用的扩大。

根据一些大公司的统计，软件维护费用大约占到软件总花费的 2/3，比开发费用高出一倍。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，称为“纠错性维护”。在此后的运行过程中，还常常要为完善功能、适应环境变更等原因对软件修改，即进行所谓“完善性维护”和“适应性维护”（详见第 11.1 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工作中出现的问题比做冰海中横在前进航道上的冰山，或直称之为维护墙（Maintenance Wall），视之为软件生产和维护中难以逾越的障碍。

（2）软件生产技术进步缓慢，是加剧这一软件危机的重要原因。

有人统计，硬件的性能价格比在过去 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。一方面，软件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，创建新软件的能力提高得十分缓慢（如图 1.2 所示）。另一方面，很多在早期用“自由化”方法开发的、带有很强“个人化”特征的程序，因缺乏文档而根本不能维护，更加剧了供需之间的矛盾。结构化程序设计的出现，使许多产业界人士认识到必须把软件生产从个人化方式改变为工程化方式，从而导致了软件工程的诞生。

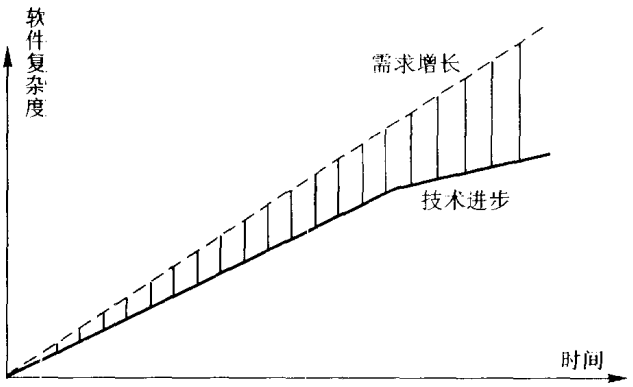

图 1.2 软件技术进步落后于需求的增长

1.2 软件工程学的范畴

“软件工程”一词，首先是 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 30 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”这一计算机新学科，图 1.3 列举了它所包含的主要内容。

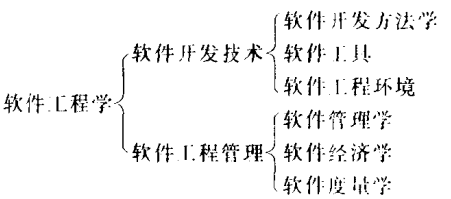

图 1.3 软件工程学的范畴

1.2.1 软件开发方法学

软件的发展,大体上经历了程序、软件和软件产品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,并且将前者称为软件。但是,无论是软件或程序,在开发过程中都很少考虑到对它们的维护。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须完成的"文档"(Documents)。

与上述3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。20世纪60年代后期兴起的结构化程序设计,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"等多个阶段。把结构化的思想扩展到分析阶段和设计阶段,于是形成了"结构化分析"与"结构化设计"等传统的软件开发技术。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,但它们都遵循结构化程序设计的原则,都对软件开发步骤和文档格式提出了规范的要求。软件生产已经摆脱了过去随心所欲的"个人化"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。

20世纪80年代出现的smalltalk、C++等语言,促进了面向对象程序设计的广泛流行。但是,人们继而发现,仅仅使用面向对象程序设计不会产生最好的效果。只有在软件开发的早期乃至全过程都采用面向对象技术,才能更好地发挥该技术的固有优势。于是,包括"面向对象需求分析—面向对象设计—面向对象编码"在内的软件开发方法学开始形成,并且正在逐步取代传统的软件开发方法,成为许多软件工程师的首选方法。面向对象技术还促进了"软件复用"技术的发展。复用加快了开发,开发又产生更多的可复用软件构件,使软件复用最终成为软件开发方法学的一个重要组成部分。

1.2.2 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。

试设想在微机上用PASCAL语言开发一个应用软件的过程。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,再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。众多的工具组合起来，成龙配套，可以组成“工具箱(Tool Box)”或“集成工具(Integrated Tool)”，供开发人员在不同的阶段按需选用。

1.2.3 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。创建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20年来出现的 UNIX 环境，Windows 环境，以及形式繁多的网络环境等，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。

1.2.4 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。不少软件在生产中不能按质按时完成计划，管理混乱往往是其中的重要原因。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开发计划，实现预期的经济和社会效益。它包括成本估算，进度安排、人员组织和质量保证等多方面的内容，还涉及管理学、度量学和经济学等多项学科。

显然，软件管理也可以借助计算机来实现。一些帮助管理人员估算成本、制定进度、生成报告等工具现在已研制出来。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 传统软件工程和面向对象软件工程

正如程序设计可区分为“面向过程程序设计”与“面向对象程序设计”一样，软件工程也可区分为“传统软件工程”和“面向对象软件工程”。前者以结构化程序设计为基础，后者以

面向对象程序设计为基础。本书将平行讲解这两种软件工程。作为导引，本章仅对它们作概略的说明。

1.3.1 程序设计方法的两次飞跃

曾被称誉为“程序设计方法的革命”的结构化程序设计，使程序设计从主要依赖于程序员个人的自由活动变成为有章可循的一门科学。它的主要贡献，是推动了程序设计风格从“追求技巧与效率”到“清晰第一、效率第二”的转变，从而提高了程序的易读性和可靠性。20世纪70年代以后，结构化的技术从编码阶段扩展到设计与分析阶段，逐步形成了结构化的软件开发范型，即今天人们常说的传统(Conventional)软件工程或经典(Classical)软件工程。

但是从早期的FORTRAN到现代的C语言，传统软件工程使用的编码语言全是面向过程的，都存在一个先天的弱点：当对软件进行分析或设计时，开发人员总是遵循“程序 = 数据结构 + 算法”的思路，把程序理解为由一组被动的数据和一组能动的过程所构成。可是在客观事物中，实体的内部“状态”(可用数据表示)和“运动”(加于数据的操作)却是结合在一起的，这就使采用传统范型开发的软件模型(Solution Domain，“解空间”)被人为地偏离客观实体本身的模型(Problem Domain，“问题空间”)。结构化程序设计的普及促进了软件生产的工业化，也缓解了当时的软件危机，然而它并未改变面向过程的程序设计思路。实践表明，用结构化技术处理50 000行以下代码的软件的确是十分有效的，但面对当今的大规模软件产品的复杂性，却仍旧无能为力。

在面向对象的程序设计中，数据及其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，“对象 + 消息”的机制取代了“数据结构 + 算法”的思路，因而较好地实现了“解空间”与“问题空间”的一致性，为解决软件危机带来了新的希望。从结构化程序设计到面向对象的程序设计，是程序设计方法的又一次飞跃，在软件开发和维护中后者正日益显露其优越性。

1.3.2 面向对象程序设计的优势

两种程序设计的不同，首先在于程序设计的思路。面向过程的程序设计认为“程序 = 数据结构 + 算法”，面向对象的程序设计则遵循“程序 = 对象 + 消息”。

为便于对照，先看一个简单的例子——银行储蓄处理事务。这一事务包含1个数据(账户余额)和3个对数据的操作——存款、取款与利息结算(每年度一次)。图1.4显示了使用两类不同编码语言实现这一事务的模型。

在采用结构化范型的图1.4(a)中，数据“账户余额”与施加在其上的操作是分离的，存款、取款与利息结算等3个过程模块分别将相应的操作施加于“账余额”，使数据获得更新。图1.4(b)则采用面向对象范型，存款、取款与利息结算作为对象内部的3个事件过程，与“账户余额”一起封装在称为“银行账户”的对象中，通过来自对象外部的3种不同的消息(图中用粗线箭头表示)，可以启动相应的操作。