

国家信息产业部电子人才交流中心参与规划
“十一五”高等院校应用型规划教材

软件工程

R u a n J i a n G o n g C h e n g

理论与实践

L i L u n Y u S h i J i a n

理论：系统、全面

案例：工程模板与毕业设计

科海策划

肖 汉 主编

宋玉璞 邵 杰 冯 娜 副主编



科学出版社

国家信息产业部电子人才交流中心参与规划
“十一五”高等院校应用型规划教材

软件工程理论与实践

主编 肖汉

副主编 宋玉璞 邵杰 冯娜

科学出版社

内 容 简 介

软件工程是继程序设计课程之后,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,全面、系统地介绍了软件工程的概 念、技术与方法,内容包括软件开发模型、软件项目管理、需求分析、软件设计、软件维护、软件重用等。书中含有丰富的例题与大量习题,便于老师教学和学生自学。

本书强调理论与实践的紧密结合、软件开发方法与 CASE 工具的相辅相成。全书条理性和逻辑性较强,可作为高等院校计算机专业的软件工程课程的实践型教材,也可作为软件开发人员的参考用书。

图书在版编目(CIP)数据

软件工程理论与实践/肖汉,宋玉璞,邵杰,冯娜编著.

—北京:科学出版社,2006

ISBN 7-03-016768-6

I. 软… II. ①肖…②宋…③邵…④冯… III. 软件工程

IV. TP311.5

中国版本图书馆 CIP 数据核字(2006)第 002000 号

责任编辑:何武 / 责任校对:刘雪莲

责任印刷:科海 / 封面设计:林陶

科学出版社 出版

北京东黄城根北街 16 号

邮政编码:100717

<http://www.sciencep.com>

北京科普瑞印刷有限责任公司印刷

科学出版社发行 各地新华书店经销

*

2006 年 7 月第一版

开本:16 开

2006 年 7 月第一次印刷

印张:18

印数:1-4000

字数:438 千字

定价:28.00 元

(如有印装质量问题,我社负责调换)

前 言

软件工程是计算机学科中一个年轻而充满活力的研究领域。自 20 世纪 60 年代末期以来，人们为克服“软件危机”在这一领域做了大量工作，逐渐形成了系统的软件开发理论、技术和方法，它们在软件开发实践中发挥了重要作用。今天，现代科学技术将人类带入了信息社会，计算机软件扮演着十分重要的角色，软件工程已成为信息社会高技术竞争的关键领域之一。

本书特点

“软件工程”是高等院校计算机教学计划中的一门核心课程，主要内容包括支持软件开发和维护的理论、方法、技术、标准以及计算机辅助工具和环境。这些内容对于软件研制人员、软件项目管理人员都是必需的。本书比较全面、系统地反映了软件工程课程的全貌，既兼顾了传统的、实用的软件开发方法，又介绍了软件工程领域比较新颖的技术和方法，包括面向对象的需求分析与软件设计方法、软件重用技术，并附有较完整的软件工程实践文档以供有兴趣的读者进一步学习和研究。本书的另一重要特点是理论与实践相结合，软件工程的技术与方法相结合。全书内容的选材强调实用价值和可操作性，强调 CASE 工具和环境对软件开发全过程的支持。此外，作者在本书的写作过程中深感软件工程的内容较易流于琐碎、冗繁，往往给读者造成软件工程缺乏深度、缺乏内在逻辑关联等印象，所以，本书力求语言的精练，注重内容的条理性、系统性和逻辑性。

内容组织

全书共 13 章，大致分为 5 个部分。前 3 章构成本书的第一部分，内容包括：软件工程的基本概念、软件开发模型、软件的质量与度量、开发过程管理与质量控制。第 4~6 章构成本书的需求分析部分，内容包括：需求分析的任务与原则，面向数据流、面向对象的需求分析方法，需求规格说明与评审。第 7~9 章构成本书的软件设计部分，内容包括：软件设计过程与一般性技术，面向数据流、面向对象的设计方法，设计规格说明与评审。第 10~12 章构成本书的软件实现与维护部分，内容包括：程序设计语言与编码、软件检验与测试、软件维护。第 13 章为本书最后一章，专门介绍比较新颖、颇具发展潜力的软件开发技术——软件重用。

读者对象

本书主要供高等院校计算机专业学生作为软件工程课程的教材使用，其内容满足教育部颁布的计算机专业软件工程课程教学基本要求。同时，本书也适合于软件开发人员与软件项目管理人员作为技术参考书使用。

肖汉组织了本书的编写工作并撰写了第 1、3、5、7、8、12、13 章，第 2 章由谷国太撰写，第 4 章由邵杰撰写，第 6、9 章由宋玉璞撰写，第 10、11 章由冯娜撰写。本书的附录 A 由冯娜收集整理，附录 B 由谷国太撰写。肖汉同志对全书进行了统稿。

最后，诚恳欢迎各位读者对本书的缺点、错误提出批评。

作者电子函件：xiaohan70@163.com

编 者

2006 年 5 月

目 录

第 1 章 软件工程概述	1
1.1 软件概述.....	1
1.1.1 软件与软件的组成.....	1
1.1.2 软件的特点.....	1
1.1.3 软件分类.....	2
1.1.4 软件的发展历史.....	3
1.1.5 软件危机.....	4
1.1.6 软件工程.....	5
1.2 软件开发方法.....	6
1.3 CASE 工具及环境.....	7
1.3.1 计算机辅助软件工程.....	7
1.3.2 CASE 工具.....	7
1.3.3 集成化的 CASE 环境.....	8
1.4 习题.....	9
第 2 章 过程和生命周期的建模	10
2.1 过程的含义.....	10
2.2 软件过程模型.....	11
2.2.1 瀑布模型.....	11
2.2.2 原型化模型.....	14
2.2.3 渐进式模型.....	15
2.2.4 过程的迭代.....	16
2.2.5 V 模型.....	18
2.3 软件开发标准.....	18
2.3.1 国内外软件开发标准的概述.....	18
2.3.2 软件工程标准的层次.....	20
2.4 习题.....	22
第 3 章 软件项目管理	23
3.1 软件度量.....	23
3.1.1 度量、测量和估算.....	24
3.1.2 面向过程软件度量.....	24
3.1.3 面向对象软件度量.....	26
3.2 软件项目估算.....	28
3.2.1 功能点估算.....	28
3.2.2 COCOMO (COConstructive COst MOde) 模型.....	29

3.3 软件质量度量	30
3.3.1 软件质量定义及三层次度量模型	30
3.3.2 软件质量要素	31
3.4 软件复杂性度量	33
3.4.1 软件复杂性及度量原则	33
3.4.2 结构复杂性度量	34
3.4.3 基于 UML 的类复杂性度量	35
3.5 软件可靠性度量	36
3.5.1 软件可靠性的概念	36
3.5.2 软件可靠性估算	37
3.6 软件开发过程的管理	37
3.6.1 风险分析	37
3.6.2 进度安排	38
3.6.3 软件开发人员的组织与分工	39
3.6.4 软件项目的开发过程管理	40
3.7 软件能力成熟度模型	40
3.7.1 概述	40
3.7.2 能力成熟度模型 CMM	43
3.7.3 能力成熟度集成模型 CMMI	53
3.7.4 我国软件能力评估标准的制定与实施	56
3.8 习题	57
第 4 章 需求分析基础	58
4.1 需求分析各阶段的基本概念	58
4.2 问题分析阶段	59
4.2.1 初步需求获取技术	59
4.2.2 需求建模方法	60
4.3 需求描述阶段	61
4.4 需求评审阶段	66
4.5 习题	67
第 5 章 面向数据流的需求分析	68
5.1 结构化分析概述	68
5.2 数据流图	70
5.2.1 数据流图的基本成分	70
5.2.2 由外向里画数据流图	72
5.2.3 分层数据流图	73
5.2.4 自顶向下画分层数据流图	75
5.2.5 数据流图的改进	77
5.3 数据词典	80
5.3.1 词典条目的各种类型	80

5.3.2 词典条目的实例.....	83
5.4 小说明.....	84
5.4.1 加工的描述.....	84
5.4.2 结构化语言.....	85
5.4.3 判定表.....	87
5.4.4 判定树.....	88
5.4.5 词典的实现.....	88
5.5 SA 方法小结.....	89
5.6 习题.....	90
第 6 章 面向对象的需求分析.....	91
6.1 面向对象软件工程方法简介.....	91
6.1.1 面向对象方法概述.....	91
6.1.2 面向对象软件开发方法.....	92
6.1.3 面向对象开发方法中的基本概念.....	94
6.2 统一建模语言 UML.....	98
6.2.1 UML 简介.....	98
6.2.2 UML 模型.....	99
6.2.3 UML 软件分析与开发步骤.....	103
6.3 面向对象需求分析.....	104
6.3.1 面向对象需求分析过程简介.....	104
6.3.2 对象与类的识别.....	106
6.3.3 属性的识别.....	110
6.3.4 服务的识别.....	111
6.3.5 定义结构.....	114
6.3.6 对象间通信.....	120
6.3.7 实例关联.....	122
6.4 习题.....	125
第 7 章 软件设计基础.....	126
7.1 软件设计概述.....	126
7.1.1 软件设计的任务和步骤.....	126
7.1.2 概要设计的基本概念.....	126
7.1.3 详细设计的基本概念.....	127
7.2 模块化设计.....	130
7.2.1 模块概述.....	130
7.2.2 块间联系和块内联系.....	132
7.3 结构图的改进.....	138
7.4 详细设计的描述方式.....	141
7.5 习题.....	146

第 8 章 面向数据流的设计方法	147
8.1 基本概念和设计步骤	147
8.2 变换分析	149
8.3 事务分析	151
8.5 综合设计	152
8.6 SD 方法小结	152
8.7 软件设计文档	153
8.7.1 概要设计说明书	153
8.7.2 详细设计说明书	156
8.8 其他设计方法	157
8.9 习题	159
第 9 章 面向对象的设计方法	160
9.1 面向对象设计的基础概念	160
9.2 面向对象设计	161
9.3 面向对象设计的子系统设计	163
9.4 用例分析实例	168
9.4.1 用例图	168
9.4.2 用例模板	168
9.4.3 一个用例模板示例	169
9.4.4 用例的七个要点	170
9.5 习题	172
第 10 章 程序设计语言和编码	173
10.1 程序设计语言	173
10.1.1 语言类别	173
10.1.2 语言选择	175
10.2 SP 方法与编程	176
10.3 程序内部文档	177
10.4 编程风格	178
10.4.1 编码原则	178
10.4.2 编程风格	180
10.5 程序的效率	183
10.6 程序设计自动化	185
10.7 习题	185
第 11 章 检验和测试方法	186
11.1 检验的基本概念	186
11.1.1 概述	186
11.1.2 软件检验手段	187
11.2 软件评审	188
11.2.1 评审过程	188

11.2.2 评审条款	190
11.3 测试的基本概念	191
11.4 白盒法	192
11.4.1 概述	192
11.4.2 语句覆盖	193
11.4.3 判定覆盖	195
11.4.4 条件覆盖	195
11.4.5 判定/条件覆盖	195
11.4.6 条件组合覆盖	196
11.5 黑盒法	196
11.5.1 等价分类法	197
11.5.2 边缘值分析法	199
11.5.3 因果图法	200
11.5.4 错误推测法	201
11.6 综合策略	202
11.7 测试过程	203
11.8 单元测试	204
11.8.1 单元测试概述	204
11.8.2 单元测试的方法	205
11.9 整体测试	206
11.10 有效性测试	209
11.11 系统测试	209
11.12 综合测试文档	210
11.13 习题	212
第 12 章 软件维护	213
12.1 维护的基本概念	213
12.2 维护的种类	214
12.3 维护的步骤与工作	215
12.3.1 维护步骤	215
12.3.2 维护工作	216
12.4 可维护性	217
12.4.1 影响软件可维护性的因素	217
12.4.2 可维护性复审	219
12.5 维护工作的管理	220
12.5.1 维护的管理和组织	220
12.5.2 维护报告	220
12.6 维护的副作用	222
12.7 软件维护文档	223
12.8 习题	227

第 13 章 软件重用技术	228
13.1 软件重用.....	228
13.1.1 软件重用的概念.....	228
13.1.2 软件重用的发展历史和重用过程.....	228
13.1.3 软件重用的层次和要素.....	229
13.2 软构件技术.....	231
13.2.1 构件技术的概念.....	231
13.2.2 可复用构件的设计准则.....	232
13.2.3 JavaBeans 构件模型.....	232
13.3 域工程.....	233
13.3.1 域工程概述.....	233
13.3.2 域分析.....	234
13.3.3 域模型.....	235
13.3.4 域分析方法.....	235
13.3.5 域分析过程.....	236
13.3.6 域分析过程的 UML 描述.....	238
13.3.7 域设计.....	238
13.3.8 域实现.....	238
13.4 软构件库的开发.....	239
13.4.1 构件库的基本概念.....	239
13.4.2 构件库设计与实现.....	239
13.5 面向对象的软件重用技术.....	243
13.6 软件重用度量.....	244
13.7 习题.....	245
附录 A 文档格式	246
A.1 用户手册.....	246
A.2 操作手册.....	249
A.3 C 程序编程工程化约束.....	252
附录 B 课程实习项目文档实例	257
B.1 可行性研究报告.....	257
B.2 项目开发计划任务书.....	260
B.3 需求规格说明书.....	262
B.4 概要设计说明书.....	264
B.5 详细设计说明书.....	271
B.6 用户操作手册.....	275
参考文献	277

第 1 章

软件工程概述

1.1 软件概述

1.1.1 软件与软件的组成

计算机软件 (software) 是指与计算机系统操作有关的程序、规程、规则及任何与之有关的数据和文档资料。它由两部分组成：一是使计算机硬件能完成计算和控制功能的有关计算机指令和计算机数据定义的组合，或机器可执行的程序及有关数据；二是机器不可执行的，与软件开发、运行、维护、使用和培训有关的文档。

程序 (program) 是用程序设计语言描述的、适合于计算机处理的语句序列。它是软件开发人员根据用户需求开发出来的。程序设计语言编译器可以将程序翻译成一组机器可执行的指令。这组指令亦称机器语言程序，它将根据用户的需求，控制计算机硬件的运行、处理用户提供的或机器运行过程中产生的各类数据并输出结果。

文档 (document) 是一种数据媒体和其中所记录的技术数据和信息，包括计算机的列表和打印输出。文档用来记录计算机软件的要求、设计或细节，解释软件的能力和限制条件，或提供在软件运行期中使用或保障计算机软件的操作命令。文档记录软件开发的活动和阶段成果，它具有永久性并能供人或机器阅读。文档不仅可以用于专业人员和用户之间的通信和交流，而且还可以用于软件开发过程的管理和运行阶段的维护。

程序开发是创造性的脑力劳动，最终结果是产生能够正确运行的程序。文档是脑力劳动过程的真实的记录，是从事软件开发和维护活动的依据，是软件生命周期不可分割的一部分，是用自然语言对思想的非（半）结构化的描述。即便是用完全形式化的体系描述进行软件的开发，也需要编写必要的文档。

1.1.2 软件的特点

- 软件是逻辑产品而不是物理产品。因此，软件在开发、生产、维护和使用等方面与硬件相比均存在明显的差异。
- 软件是逻辑实体，始终不会自然变化，只是其载体可变。而物理实体同规格产品的质量特性之间有散布，会随时间和使用而老化、磨损以至失效。
- 软件是一种创造性的思维活动。即给定一个任务后，可有不同的方法，结果也可

能大相径庭，无现成的东西可以直接借鉴，与一些传统的生产工艺是不一样的。

- 软件是可以长期运行的，不会因时间长短而磨损、老化。只要需要，一个久经考验的优质软件是可以长期使用下去的，而这一点硬件是做不到的。软件的淘汰属于技术上的淘汰，即需要更优良的软件来替代。
- 软件的研制过程主要是脑力劳动过程，在本质上是无形的，不可见的，难以控制的。而硬件研制过程不只是脑力劳动过程，还有体力劳动过程，其过程有形，便于测控。
- 程序是指令序列，在执行时指令的逻辑组合状态千变万化，因此即使每条指令都正确，结果也不一定完全正确。而硬件的不可靠问题不只是设计问题，在生产和使用过程中也会产生新的故障。
- 软件中系统的数学模型是离散型的，其在合理范围内输入的微小变化可能引起输出的巨大变化，故障的形成无物理原因，失效的发展取决于输入值和运行状态的组合，无前兆。而硬件系统在正常工作条件下其行为是渐变的，故障的形成和失效的发生一般都有物理原因，有前兆。
- 对软件的生产过程进行严格的控制，可得到完全一致的产品。而对硬件生产过程进行严格的控制，可将产品的容差控制在可接受的范围内。
- 软件中不可靠问题基本是由于开发过程中的人为差错所造成的缺陷而引起的。而硬件失效通常是由其零部件或其结合的故障所引起的。
- 软件在使用过程中出现故障后，必须修改原产品以解决问题；只要在修改时未引入新问题，其可靠性就会增长。而硬件在使用过程中出现故障后，一般只需更换或修复失效的部件，使产品恢复良好状态，其可靠性一般不会提高。
- 软件维护通常涉及软件更改，会对其他部分造成影响。而硬件维修通常涉及零部件更换，一般不会对其他部分造成影响。
- 软件的冗余设计应确保冗余软件相异，否则不仅不能提高可靠性，反而会增加复杂性，降低可靠性。而硬件相同部件之间是独立的，适当的冗余可以提高可靠性。

1.1.3 软件的分类

20 世纪 40 年代以来，人们开发了大量的软件，积累了丰富的软件资源并使之广泛应用于科学研究、教育、工农业生产、事务处理、国防和家庭等。下面介绍几种在各个领域中常用的计算机软件分类方法。

1. 按功能

(1) 系统软件：管理、控制和维护计算机系统中的各种资源（包括硬件和软件），并使这些资源充分发挥作用，提高计算机的工作效率，方便用户使用计算机。如操作系统、语言处理程序等。

(2) 支撑软件：旨在帮助编制应用软件。如 WPS、Office 等。

(3) 应用软件：为计算机的特定应用提供惟一的特定功能，一般来讲就是为了解决科学技术、生产、生活等许多实际问题而编写的程序。如信息系统、通信软件等。

2. 按工作方式

(1) 实时处理软件：对于给定的一个时间约束量 $\varepsilon > 0$ ，如果系统 S 在 T_1 时刻接受输入，在 T_2 时刻给出合理的输出，且使 $T_2 - T_1 < \varepsilon$ ，则称系统 S 满足要求的时间 ε 的实时性，通常称系统 S 为实时系统。实时处理软件一般应用于控制系统上，永远不能停下来。如交通管理系统等。

(2) 嵌入式系统：由软件配置项和硬件技术状态相互嵌入组成的系统。嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分，控制该系统的运行，进而实现一个特定的物理过程。例如，汽车的刹车控制、电视机和洗衣机的自动控制等。

3. 按规模

(1) 小型程序：包括工程师们用于解答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，以及在校的课程设计中学生编写的程序。这类程序的长度一般不超过 2000 行，与其他程序也没有什么联系。

(2) 中型程序：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。这类程序可能与其他程序有少量联系，也可能没有。

(3) 大型编译程序、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。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。极大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，如空中交通管制系统、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，开发周期可长达 10 年，并要求有极高的软件可靠性。

4. 按使用的频度

常用软件和不常用软件。如人口普查软件每隔 4~5 年用 1 次，而财务报账系统、银行系统等是常用的，常年运行。

5. 按服务对象

专用软件和通用软件。专用软件一般用于解决某个特定问题，如人事管理、财务管理等。通用软件则适用于解决多个领域中的一般问题，适用面较广，如文字处理软件、病毒软件等。

6. 按软件失效性

一般性软件和高可靠性软件，如核电站的软件、卫星发射控制系统等为高可靠性软件。

1.1.4 软件的发展历史

软件是由计算机程序和程序设计的概念发展演化而来的，是在程序和程序设计发展到一定规模并且逐步商品化的过程中形成的。19 世纪初，法国人约瑟夫·雅各（Joseph Marie Jacquard）设计的织布机，就能够通过“读”穿孔卡上的信息完成预定的任务。英国

诗人拜伦 (Byron) 的女儿, 数学家爱达·奥古斯塔·拉夫拉斯伯爵夫人 (Ada Augusta Lovelace) 在帮助巴贝奇研究分析机时, 指出分析机可以像织布机一样进行编程, 并发现进行程序设计和编程的基本要素, 被认为是有史以来的第一位程序员, 而著名的计算机语言 Ada 就是以她的名字命名的。

在计算机系统发展的早期 (20 世纪 60 年代中期以前), 计算机硬件已相当普遍, 软件只是为每个具体应用而专门编写的。这时的软件通常是规模较小的程序, 编写者和使用者往往是同一个人。在这种个体化的软件环境下, 软件设计通常只是在人们头脑中进行的一个隐含的过程, 除了程序清单之外, 没有其他文档资料保存下来。到了 20 世纪 70 年代, 出现了“软件作坊”, 开始广泛使用产品软件。但是, “软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。随着计算机应用的日益普及, 软件规模急剧膨胀。在程序运行时发现的错误必须设法改正; 用户有了新的需求时必须相应地修改程序; 硬件或操作系统更新时, 通常需要修改程序以适应新的环境。上面的种种软件维护工作, 使得资源耗费十分严重。

1.1.5 软件危机

1. 软件危机的表现

软件危机最早出现于 20 世纪 60 年代末期, 主要表现是: 软件质量差, 可靠性不能保证; 软件成本的增长难以控制, 在成本预算内往往不能完成任务; 软件开发进度不易控制, 周期延长; 软件维护困难, 维护人员和维护费用不断增加等。例如, IBM 公司开发的 OS/360 系统, 虽然耗资数千万美元, 花费五千多人年, 拖延了几年才交付使用, 但之后还是不断发现错误; 有的软件在耗费了大量的人力物力后, 无果而终; 更为严重的是, 用错误方法开发出来的许多大型软件几乎无法维护, 只好提前报废, 造成大量人力、物力、财力的浪费。同时人们还发现, 在研制软件系统期间也需要投入大量的人力、物力, 但系统的质量却还是难以保证。这样, 在软件开发所需的高成本与产品的低质量之间存在着尖锐的矛盾, 这就是所谓的“软件危机”。概括地说, 软件危机包含下述两方面的问题: 如何开发软件, 怎样满足人们对软件的日益增长的需求; 如何维护规模不断膨胀的已有软件。具体地说, 软件危机主要有下述一些表现:

- (1) 对软件开发成本和进度的估计常常不准确。
- (2) 用户对“已完成的”软件系统不满意的现象经常发生。
- (3) 软件产品的质量往往靠不住。
- (4) 软件常常是不可维护的。
- (5) 软件通常没有适当的文档资料。
- (6) 软件成本在计算机系统总成本中所占的比例逐年上升。
- (7) 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。

2. 软件危机产生的原因

从软件危机的种种表现和软件作为逻辑产品的特殊性, 可以发现软件危机产生的原因:

- (1) 用户对软件需求的描述不精确, 可能有遗漏、有二义性、有错误; 甚至在软件开

发过程中，用户还提出修改软件功能、界面、支撑环境等方面的要求。

(2) 软件开发人员对用户需求的理解与用户的本来愿望有差异，这种差异必然导致开发出来的软件产品与用户要求不一致。

(3) 大型软件项目需要组织一定的人力共同完成。多数管理人员缺乏开发大型软件系统的经验，而多数软件开发人员又缺乏管理方面的经验。各类人员的信息交流不及时、不准确、有时还会产生误解。

(4) 软件项目开发人员不能有效地、独立自主地处理大型软件的全部关系和各个分支，容易产生疏漏和错误。

(5) 缺乏有力的方法学和工具支持，过分地依靠程序设计人员在软件开发过程中的技巧和创造性，加剧软件产品的个性化。

(6) 软件产品的特殊性和人类智力的局限性，导致人们无力处理“复杂问题”。

3. 解决软件危机的途径

软件开发不应该是某种个体劳动的神秘技巧，而应该是一种组织良好、管理严密、各类人员协同配合来共同完成的工程项目。软件开发必须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，特别要吸取人类几十年来从事计算机软件研究和开发的经验教训。

人们应该开发和使用更好的软件工具。因为在软件开发的每个阶段都有许多繁琐重复的工作需要做，而在适当的软件工具辅助下，开发人员可以把这类工作做得既快又好。如果把各个阶段使用的软件工具有机地集成成一个整体支持软件开发的全过程，则称为软件工程支撑环境。

总之，为了解决软件危机，既要有技术措施，又要有必要的组织管理措施。软件工程正是从管理和技术两方面，研究如何更好地开发和维护计算机软件的一门新兴学科。

1.1.6 软件工程

1968年，北大西洋公约组织(NATO)召开计算机科学会议，弗里兹·鲍尔(Fritz Bauer)首先提出了“软件工程”的概念，试图建立并使用正确的工程方法开发出成本低、可靠性好，并在机器上能高效运行的软件，从而解决或缓解软件危机。

软件工程是指导计算机软件开发和维护的工程学科。其中具有代表性的定义有如下几个：

(1) Fritz Bauer 在 NATO 会议上给出的定义：“软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件而确立和使用的健全的工程原理(方法)。”

(2) IEEE【IEEE83】给出的软件工程定义：“软件工程是开发、运行、维护和修复软件的系统方法。”

(3) IEEE【IEEE93】给出了一个更加综合的定义：“软件工程是将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中的方法的研究。”

软件工程的研究内容是采用工程化的概念、原理、技术和方法，把经过时间考验而证

明正确的管理技术和当前能够得到的最好的技术方法结合起来开发与维护软件。它支持项目计划和估算、系统和软件需求分析、软件设计、编码、测试和维护。软件工程的方法、工具、过程构成了软件工程的三要素。软件工程所包含的内容不是一成不变的，而是随着人们对软件系统的研制开发和生产的理解，应用发展的眼光看待它的。

软件工程的目的是成功地建造一个大型软件系统，所谓成功是要达到以下 8 个目标：

- (1) 付出较低的开发成本。
- (2) 达到要求的软件功能。
- (3) 取得较好的软件性能。
- (4) 开发的软件易于移植。
- (5) 需要较低的维护费用。
- (6) 能按时完成开发任务。
- (7) 及时交付使用。
- (8) 开发的软件可靠性高。

软件工程技术两个明显特点：强调规范化，强调文档化。

1.2 软件开发方法

为了得到“高产优质”的软件，我们有必要研究一下软件开发方法和软件工具。研究软件方法的目的是使开发过程“纪律化”，使开发工作能够有计划、有步骤地进行。研究软件工具的目的是使开发过程“自动化”，就是使开发过程中的某些工作用计算机来完成。

软件开发方法是指导研制软件的某种标准规程，它告诉开发人员：“什么时候做什么以及怎么做”。具体可参照下面过程。

- (1) 明确工作目标与步骤。
- (2) 具体的描述方式。对每项任务要有具体描述即计划任务书。
- (3) 确定评价标准。就是要有一个系统的评估标准来对几种预选的方案进行筛选。
- (4) 软件开发标准规程。有两种：
 - 结构化分析——结构化设计——结构化编程
 - 面向对象分析——面向对象设计——面向对象编程
- (5) 纪律化，指技术上的纪律性。
- (6) 尽量采用自动化工具，都是用一定标准编写出的支持软件。

近年来，人们陆续研究总结出多种软件开发方法。20 世纪 70 年代初，出现了一些编写程序的方法，主要是“结构化程序设计”；70 年代中期人们认识到编程仅仅是软件开发的一个环节，合理地建立软件结构比编写程序更为重要，所以研究重点前移到设计阶段，出现了用于设计阶段的“结构化设计”和 Jackson 方法等；70 年代后期，人们又意识到在设计阶段之前必须先对用户的要求进行分析，所以研究重点又前移到分析阶段，出现了用于分析阶段的“结构化分析”、SADT、SREM 等方法。80 年代，出现了面向对象的软件