

面向 21 世纪

高职高专计算机专业教材

软件工程

冷英男 ◀ 主编



人民交通出版社

面向21世纪



高职高专计算机专业教材

Ruanjian Gongcheng

软件工程

冷英男 主编



人民交通出版社

内 容 提 要

本书介绍了软件工程的基本概念、技术与方法,内容包括:软件开发模型,系统分析,需求分析,软件设计,编码,软件测试,维护,软件配置管理,软件度量和软件体系结构与软件设计模式等。本书取材精练,各章配有习题并有比较具体的示例,便于学习和自学。

本书注重原理,结合实践,兼顾传统技术与新技术,繁简适当,逻辑清楚,可作为高职高专院校计算机专业和相关专业的教材,也可作为软件技术人员的参考书。

图书在版编目(CIP)数据

软件工程/冷英男主编. —北京:人民交通出版社,
2004.1

ISBN 7-114-04932-3

I. 软... II. 冷... III. 软件工程 IV. TP311.5

中国版本图书馆 CIP 数据核字 (2003) 第 126958 号

面向 21 世纪高职高专计算机专业教材 软 件 工 程

冷英男、主编

正文设计:姚亚妮 责任校对:张莹 责任印制:张恺

人民交通出版社出版发行

(100013 北京和平里东街 10 号 010 64216602)

各地新华书店经销

三河市宝日龙印务有限公司印刷

开本:787×1092 1/16 印张:14.25 字数:345 千

2004 年 2 月,第 1 版

2004 年 2 月,第 1 版,第 1 次印刷

印数:0001—3000 册 定价:23.00 元

ISBN 7-114-04932-3

编写人员名单

主 编：冷英男（宁波高等专科学校）

副 主 编：从凤燕（天津交通职业学校）

王亚平（广东交通职业技术学校）

本书策划组成员名单

白 峻 翁志新 张 景 黄景宇

前言

FOREWORD

根据 21 世纪高等职业教育的新趋势和计算机专业学科建设的要求,结合目前众多高职高专院校的教学计划,人民交通出版社组织全国十几所高职高专院校的多年从事一线教学、实践能力强且具有丰富教材编写经验的教师,编写了这套“面向 21 世纪高职高专计算机专业教材”,共 21 本(书目附后),涵盖了高职高专计算机及相关专业的主要课程。在编写过程中认真贯彻了教育部《关于加强高职高专教育人才培养工作的意见》的精神。内容以必需、够用为度,既注重基础知识的讲解,又注意从实际应用出发,满足社会对计算机类专业人才的需求,突出以能力为本位的高等职业教育的特色。

应当说明的是,凡是高等职业教育、高等专科学校和成人高等院校的计算机及其相关专业的师生均可使用本套教材。各学校可以根据实际需要,在教学中适当增删一些内容,从而更有针对性地帮助学生掌握计算机专业知识,并形成相关应用能力。

本套教材的出版,将促进高等职业教育的教材建设,对我国高等职业教育的发展产生积极的影响。同时,我们也希望在今后的使用中不断改进、完善此套教材,更好地为高等职业教育服务。

编 者

目

录

CONTENTS

第1章 绪论	1
1.1 软件的基本概念	1
1.1.1 什么是软件	1
1.1.2 软件的特点和分类	2
1.1.3 软件开发涉及的问题	4
1.2 软件危机与软件工程	5
1.2.1 软件危机	5
1.2.2 软件工程学及其基本原则	7
1.2.3 软件生存周期	9
1.3 软件开发模型	10
1.3.1 瀑布模型	10
1.3.2 原型化模型	10
1.3.3 渐增模型	12
1.3.4 螺旋模型	13
1.3.5 喷泉模型	14
1.3.6 构件集成模型	14
1.4 小结	15
练习题	15
第2章 系统分析	17
2.1 基于计算机的系统	17
2.1.1 基于计算机系统的层次及元素	17
2.1.2 计算机系统工程	18
2.1.3 软件和软件工程	19
2.2 系统分析	20
2.2.1 系统分析的目标	21
2.2.2 系统分析的过程	21
2.3 可行性分析	21
2.3.1 可行性研究的内容	22
2.3.2 可行性研究的步骤	23
2.3.3 可行性研究报告	24
2.4 技术分析	25
2.4.1 系统模型	25
2.4.2 模型工具	25
2.5 成本/效益分析	27

AJ5276/01

2.5.1 成本估计	27
2.5.2 成本/效益分析的内容	28
2.6 系统规格说明与评审	30
2.7 小结	31
练习题	31
第3章 需求分析	33
3.1 需求分析的任务和过程	33
3.1.1 软件需求分析的任务与原则	33
3.1.2 软件需求的内容	34
3.1.3 软件需求分析的过程与步骤	34
3.1.4 常用需求分析方法	36
3.2 结构化分析方法(Structured Analysis ,SA)	36
3.2.1 结构化分析方法使用的工具	36
3.2.2 结构化分析的过程	45
3.2.3 原型化方法在需求分析中的作用	47
3.3 面向对象分析方法(OOA, Object-Oriented Analysis)	48
3.3.1 面向对象的基本概念	49
3.3.2 面向对象方法常用的图形工具	50
3.3.3 面向对象的分析方法学	51
3.3.4 面向对象分析过程	53
3.4 软件需求分析的验证	59
3.4.1 软件需求分析的评审	59
3.4.2 软件需求规格说明	61
3.5 小结	62
练习题	62
第4章 软件设计基础	64
4.1 软件设计的目的和任务	64
4.1.1 概要设计的任务	64
4.1.2 详细设计的任务	65
4.2 软件设计的基本概念	65
4.2.1 抽象与细化	65
4.2.2 模块化与信息隐蔽	67
4.2.3 复用	72
4.3 软件架构设计	72
4.4 过程设计的技术与工具	74
4.4.1 结构化程序设计	74
4.4.2 图形工具	75
4.5 设计规格说明书与评审	77

4.6 小结	78
练习题	79
第5章 面向数据流的软件设计方法	80
5.1 基本概念与工具	80
5.1.1 SC图	80
5.1.2 数据流图 DFD 的类型	81
5.1.3 SD 方法的步骤	82
5.2 变换分析	83
5.3 事务分析	86
5.4 结构设计优化的启发式策略	87
5.5 结构设计的示例	91
5.6 过程设计	93
5.6.1 过程设计的原则	93
5.6.2 过程设计的工具	94
5.6.3 过程设计的例	96
5.7 小结	98
练习题	99
第6章 面向对象的设计方法	101
6.1 面向对象设计概述	101
6.1.1 面向对象设计任务	101
6.1.2 OOD 方法的描述机制	101
6.2 面向对象设计方法	103
6.2.1 系统设计	103
6.2.2 对象设计	110
6.3 小结	112
练习题	112
第7章 编码与程序设计语言选择	114
7.1 程序设计语言	114
7.1.1 影响编码的程序设计语言的特点	114
7.1.2 程序设计语言的类别	115
7.1.3 程序设计语言的选择	117
7.2 编码标准	119
7.3 编码风格	120
7.3.1 编码风格指导原则	120
7.3.2 编码问题的例	122
7.4 软件设计支撑工具	125
7.5 小结	125
练习题	126

第8章 软件测试	128
8.1 测试的基本概念	128
8.1.1 软件测试的目的与特性	128
8.1.2 几个与测试相关的概念	129
8.1.3 Myers 测试经验	129
8.1.4 软件的错误	130
8.2 黑盒测试	131
8.2.1 等价分类法	132
8.2.2 边界值分析(Boundary Value Analysis)	134
8.2.3 错误猜测法	134
8.3 白盒测试	134
8.3.1 逻辑覆盖测试	135
8.3.2 基本路径测试	138
8.4 软件的测试策略	141
8.4.1 测试的层次性	141
8.4.2 单元测试	142
8.4.3 集成测试	144
8.4.4 确认测试	146
8.4.5 系统测试	147
8.5 调试(纠错)	148
8.5.1 调试的特点	148
8.5.2 调试的方法	149
8.6 面向对象的测试	149
8.6.1 面向对象软件的测试策略	150
8.6.2 类的测试用例设计	151
8.7 软件测试计划与测试分析报告	152
8.8 小结	153
练习题	154
第9章 软件维护	156
9.1 软件维护的基本概念	156
9.1.1 软件维护的定义与分类	156
9.1.2 维护代价的分布	157
9.1.3 一个维护的例子	157
9.2 维护的特点	158
9.2.1 软件工程方法对维护的影响	158
9.2.2 维护的代价	158
9.2.3 软件维护中的一些典型问题	159
9.3 可维护性	159

9.3.1	影响可维护性的因素	160
9.3.2	可维护性的若干量化的测度	160
9.3.3	保证可维护性的复审	161
9.4	维护过程与维护活动	161
9.4.1	软件维护工作的内容	161
9.4.2	建立维护机构	161
9.4.3	维护报告	162
9.4.4	维护活动的模型	163
9.4.5	维护管理文档	164
9.4.6	评价	165
9.5	软件修改的副作用	165
9.6	逆向工程与重构工程	166
9.6.1	软件重构工程模型	167
9.6.2	恢复信息的级别	167
9.6.3	恢复信息的方法	168
9.7	小结	169
	练习题	169
第10章	配置管理	171
10.1	软件配置管理的基本概念	171
10.1.1	基线技术	172
10.1.2	软件配置项	172
10.2	软件配置管理的任务	173
10.2.1	标识软件配置中各种对象	174
10.2.2	版本控制	175
10.2.3	变化控制	176
10.2.4	状态报告	178
10.2.5	配置审计	178
10.2.6	配置管理的标准和 CASE 工具	179
10.3	小结	180
	练习题	180
第11章	软件项目的度量	181
11.1	软件度量的基本概念	181
11.1.1	度量,测量和估算	181
11.1.2	面向规模的度量	182
11.1.3	面向功能的度量	183
11.1.4	软件度量体系标准	185
11.2	软件项目的估算	186
11.3	经验估算模型	189

11.3.1	COCOMO 模型	189
11.3.2	Putnam 模型	191
11.4	软件的质量度量	192
11.4.1	软件质量和影响软件质量的要素	192
11.4.2	软件质量要素评价标准	193
11.5	软件复杂性度量	195
11.5.1	软件复杂性	195
11.5.2	控制结构的复杂性度量	196
11.5.3	源代码复杂性度量	197
11.6	软件可靠性估算	198
11.7	面向对象系统的技术度量	199
11.7.1	面向对象系统在度量方面的区别特征	199
11.7.2	面向类的度量	200
11.7.3	面向操作的度量	201
11.8	小结	201
	练习题	202
第 12 章	软件体系结构与设计模式	203
12.1	软件体系结构	203
12.1.1	软件体系结构的研究现状	203
12.1.2	常用的软件体系结构	205
12.2	面向对象中的模式	207
12.2.1	设计模式的概念	207
12.2.2	设计模式的组成与描述	208
12.2.3	模式的分类	209
12.2.4	Proxy 模式	210
12.2.5	如何使用模式	214
12.3	小结	215
	练习题	215
	参考文献	216

第1章 绪论

软件工程是一门指导计算机软件开发和维护的工程学科。它借助传统的工程学原则,利用计算机科学、数学、管理科学的知识指导软件的开发,以便保障软件开发的低成本、高质量。计算机在现代社会中的作用越来越大,而计算机软件是计算机系统不可缺少的组成部分,随着计算机应用需求的不断扩大,如何更快、更好、更多地开发出好用的软件已经成为一个重要的问题。软件工程就是为解决这类问题而提出的。历史上,计算机软件的发展经历了几个不同的时期。在最初,计算机软件只注重编程经验和技巧,随着计算机应用的普及,计算机软件的开发人员开始注重程序设计风格。随后,由于软件的规模进一步增大,软件的复杂程度也在不断增加,软件的开发和维护遇到了一系列严重问题,使得软件开发人员必须从更高的角度来对待软件,于是便提出了软件工程。软件工程在20世纪60年代提出以来,已有三十多年的历史,由于它强调软件开发的科学管理和规范化,已经对软件领域的发展起了很大作用。本章简要介绍软件工程的有关基本概念,包括软件,软件危机,软件工程的原则等。

1.1 软件的基本概念

软件概念的内涵有一个随着计算机科学与技术的发展而逐步扩充的过程。本节将介绍软件的有关基本概念、软件的特点、软件的分类以及软件开发涉及的问题。

1.1.1 什么是软件

“软件”是英文 software 的中译。而 software 一词于20世纪60年代初用于计算机领域。英文 software 一词是 soft 和 ware 两字的组合。有人译为“软制品”,也有人译为“软体”,现在人们统称它为软件。软件这一概念的一种公认的解释为,软件是计算机系统中与硬件相互依存的另一部分,它是包括程序、数据及其相关文档的完整集合。其中:

- 程序是按事先设计的功能和性能要求编写的指令序列,这些在执行时能够提供期望的功能和性能;程序通常用程序设计语言描述,由程序设计人员开发,由程序设计语言翻译程序转换成机器指令,在计算机上运行。
- 数据是使程序能正常操纵信息的数据结构,这些数据结构使得程序在运行时能够完全操纵必要的信息。
- 文档是与程序开发、维护和使用有关的图文材料,这些文档描述了程序的分析、设计、实现和维护的细节及使用说明。一般地,文档可以存储在各种媒体上,具有永久性。文档的制定有一定的规范。例如,我国颁布的《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》等。



通常,人们用“软件 = 程序 + 数据 + 文档”这一公式来说明软件的组成。

1.1.2 软件的特点和分类

和硬件相比,软件有其独有的特点及复杂的系统分类。

1. 软件的特点

软件在开发、制造、维护过程以及其自身属性诸方面与硬件都不相同。

1) 软件是一种逻辑实体,不是具体的物理实体

软件具有抽象性。这个特点使它和计算机硬件,或是其他工程对象有着明显的差别。人们可以把它记录在纸面上,保存在计算机的存储器内部,也可以保留在磁盘、磁带和光盘上,但除了软件的代码外,无法看到软件本身的形态,必须通过观察、分析、思考、判断,去了解它的功能、性能及其他特性。

2) 软件的构造过程是“开发”,而不是“制造”

软件的构造过程取决于开发人员的素质、组织和智力劳动。开发成功后的“制造”只是简单的拷贝,因而软件的质量主要取决于软件的“开发”过程。

3) 软件的维护与硬件不同

软件从被提交给用户使用开始,只存在维护问题,而不存在使用过程中被“用坏”的问题。而硬件则不同,用户购买到硬件并开始使用硬件后,随着时间的过去,硬件会因为各种原因(如灰尘、振动、高温、磨损、滥用等)造成某些部件失效,从而导致硬件被“用坏”了。

软件的使用过程中出错的可能性与硬件不同。图 1-1 是软件使用过程中随时间变化的错误率曲线的示意图。在软件使用的早期,错误率是比较大的,这主要是因为软件开发时的错误或缺陷造成的。随着时间的推移和错误的改正,软件中潜在的错误越来越少,因而曲线变得越来越平坦。换句话说,软件不会被“用坏”。当然,随着时间的不断推移,软件最终会由于环境或需求的变化而不能满足用户的需要被“废弃”。

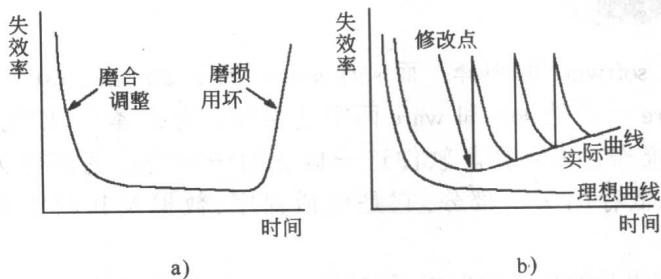


图 1-1 软件随时间变化的错误率曲线

a) 硬件失效率曲线; b) 软件失效率曲线

4) 软件的开发和运行常常受到计算机系统的限制

软件的开发和运行对计算机系统有着不同程度的依赖性,软件不能完全摆脱硬件单独活动。有的软件的这种依赖性大些,常常为某个型号的计算机所专用。有的软件依赖于某个操作系统。为了解除这种依赖性,在软件开发中提出了软件移植的问题。

5) 软件本身是复杂的

软件的复杂性可能来自它所反映的实际问题的复杂性,例如它所反映的自然规律,或是

人类社会的事务,或是一项需要高智力因素的工程,都具有一定的复杂性;另一方面,也可能来自程序逻辑结构的复杂性。软件开发,特别是应用软件的开发,常常涉及到其他领域的专门知识,这对软件人员提出了很高的要求。软件的复杂性与软件技术的发展不相适应的状况越来越明显。

6) 软件成本相当昂贵

软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动,它的成本比较高。在基于计算机的系统中,软件的开销大大超过硬件的开销,如图 1-2 所示。

7) 相当多的软件工作涉及到社会因素

许多软件的开发和运行涉及,法律,机构、体制及管理方式等问题,甚至涉及到人的观念和人们的心理。对于这些人的因素重视得不够,常常是软件工作遇到的问题之一。

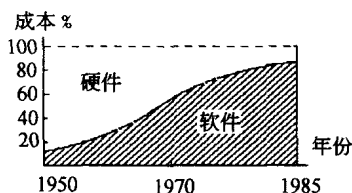


图 1-2 软件和硬件成本对比

2. 软件的分类

要给软件做出科学的分类很难,而且从不同的角度出发,有对软件的不同分类方法。下面是对软件的几种分类。

(1) 按软件的功能进行划分,软件可以分为系统软件、支撑软件和应用软件。其中:

- 系统软件是计算机管理自身资源(如 CPU、内存、外存、外部设备等),提高计算机使用效率并为用户提供各种服务的基础软件。它依赖于机器指令系统、中断系统及控制、运算、存储等使计算机系统各个部件、相关的软件和数据协调、高效地工作,为用户提供尽可能标准、方便的服务。系统软件如操作系统、设备驱动程序以及通信处理程序等。系统软件是计算机系统必不可少的一部分。

- 支撑软件是协助用户开发软件的工具软件,其中包括帮助程序人员开发软件产品的工具,也包括帮助管理人员控制开发进程的工具。表 1-1 给出了一些支撑软件的实例。

支撑软件举例

表 1-1

一般类型	支持需求分析
文本编辑程序 文件格式化程序 磁盘向磁带向数据传输的程序 程序库系统	PSL/PSA 问题描述语言、问题描述分析程序 关系数据库系统 一致性检验程序 CARA 计算机辅助需求分析程序
支持设计	支持实现
图形软件包 结构化流程图绘图程序 设计分析程序 程序结构图编辑程序	编辑程序 交叉编辑程序 预编译程序 连接编辑程序
支持测试	支持管理
静态分析程序 符号执行程序 模拟程序 测试覆盖检验程序	PERT 进度计划评审方法绘图程序



● 应用软件是在特定领域内开发,为特定目的服务的一类软件。例如商业数据处理软件、工程与科学计算软件、计算机辅助设计(CAD)/计算机辅助制造(CAM)软件、系统仿真软件、智能产品嵌入软件(如汽车油耗控制、仪表盘数字显示、刹车系统),以及人工智能软件(如专家系统、模式识别)等。

(2)按软件规模进行划分,软件可分为微型、小型、中型、大型、甚大型和极大型六种,见表 1-2。

软件规模的分类

表 1-2

类别	参加人员数	研制期限	产品规模(源程序行数)
微型	1	1~4 周	0.5k
小型	1	1~6 月	1k~2k
中型	2~5	1~2 年	5k~50k
大型	5~20	2~3 年	50k~100k
甚大型	100~1000	4~5 年	1M(=1000k)
极大型	2000~5000	5~10 年	1M~10M

(3)按软件工作方式划分,软件可分为实时处理软件、交互式软件和批处理软件,其中:

● 实时处理软件:监视、控制、分析现实世界发生的事件,在事件或数据产生时,以足够快的速度立即予以处理,并在规定时间内做出反应的软件。其内容主要包括数据采集、分析、输出三部分,其处理时间是被严格限定的。支持实时软件的操作系统叫做实时操作系统。

● 交互式软件:能实现人机通信的软件。这类软件接收用户给出的信息,但在时间上没有严格的限定。这种工作方式给用户很大的灵活性。近年来,随着终端设备更加普及,交互式软件几乎到处可见。显得日益突出的一个重要问题就是用户界面设计。良好的用户界面设计将给用户带来极大的方便。

● 批处理软件:把一组输入作业或一批数据以成批处理的方式一次运行,按顺序逐个处理完的软件。这是最传统的工作方式。

1.1.3 软件开发涉及的问题

软件开发方法是软件开发过程所遵循的办法和步骤,其目的在于有效地得到一些工作产物,也就是一个运行的系统及其支持文档,并且满足有关的质量要求。软件开发和系统演化有着各种方法,这些方法都典型地包含了分析、设计、实现、确认(验收测试)、演化(维护)等活动。

软件开发方法有些是针对某一活动的,属于局部性的软件开发方法;但实践表明,针对分析和设计活动的软件开发方法更为重要。除此之外,还有覆盖开发全过程的全局性方法,这是人们注意的重点。

那么如何评价一种具体的软件开发方法呢?一般地说,要看四个方面的特征:

● 技术特征:即支持各种技术概念的方法特色,如层次性、抽象性、并行性、安全性、正确

性等。其中抽象性包括数据抽象和过程抽象。

- 使用特征:即应用于具体开发时的有关特色,如易理解性、易转移性、易复用性、工具的支持、任务范围、使用的广度、活动过渡的可行性、产品的易修改性、对正确性的支持等。

- 管理特征:即增强对软件开发活动管理的能力方面的特色,如易管理性、支持或阻碍集体工作的程度、中间阶段的确定、工作产物、配置管理、阶段结束准则、费用估计等。

- 经济特征:即给软件机构产生的在质量和生产力方面的可见效益,如分析活动的局部效益、完全生存周期效益、获得该开发方法的代价、使用它的代价、管理的代价等。

不管怎样,在一切方面都好的开发方法并不存在;反之,也没有一种开发方法能适应于所有软件开发之需。当你需要选用一种开发方法时,会考虑如下的几个因素:

- 对该开发方法是否已具有经验,或者已有受过训练的人员。
- 开发项目进度安排、人员组成情况。
- 为开发项目提供的资源,包括软件硬件环境、可能使用的工具。
- 计划、组织、管理的可行性。
- 开发项目的领域知识准备情况。

1.2 软件危机与软件工程

软件危机的出现是计算机系统发展的产物,而软件工程是软件工作者为解决软件危机的一种努力结果。本节通过软件的发展历程介绍软件危机与软件工程的有关概念,包括:软件工程的定义及其基本原则,软件工程过程和软件生存周期。

1.2.1 软件危机

1. 软件的发展

软件经历了几十年的发展,人们对它有了更深刻的认识。在这几十年中,计算机软件经历了三个发展阶段:

- 程序设计阶段,约为 20 世纪 50 至 60 年代。
- 程序系统阶段,约为 20 世纪 60 至 70 年代。
- 软件工程阶段,约为 20 世纪 70 年代以后。

从表 1-3 中可看到三个发展时期主要特征的对比。

计算机软件发展的三个时期及其特点

表 1-3

特 点 \ 时 期	程 序 设 计	程 序 系 统	软 件 工 程
软件所指	程序	程序及规格说明书	程序、文档、数据
主要程序设计语言	汇编及机器语言	高级语言	软件语言 ^①
软件工作范围	编写程序	包括设计和测试	软件生存期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大中型软件开发机构
软件规模	小型	中、小型	大、中、小型



续上表

特 点 \ 时 期	程 序 设 计	程 序 系 统	软 件 工 程
决定质量的因素	个人程序设计技术	开发小组技术水平	管理水平
开发技术和手段	子程序 程序库	结构化程序设计	数据库、开发工具、开发 环境、工程化开发方法、标准和 规范、网络和分布式开发、对象技术
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高、存储容量 小、工作可靠性差	降价、速度、容量及工作 可靠性明显提高	向超高速、大容量、微型化 及网格化方向发展
软件特片	完全不受重视	软件技术的发展不能满足 需要、出现软件危机。	开发技术有进步,但未获突破性进展, 价格高,未完全摆脱软件危机。

注:①这里软件语言包括需求定义语言、软件功能语言、软件设计语言、程序设计语言等。

2. 软件危机

软件是计算机系统不可分割的重要组成部分,软件危机的出现则是计算机系统发展的产物。20 世纪 40 年代中期到 60 年代中期是计算机系统发展的第一个时期。在这一时期中,计算机几乎完全用于科学和工程计算。编程语言只有机器语言和后来出现的汇编语言,没有专职的程序员,这一时期的软件通常是规模比较小的程序,编写者和使用者往往是同一个人或同一组人。编程没有公共的规则可遵循。

20 世纪 60 年代中期到 70 年代中期为计算机系统发展的第二个时期,在这一时期出现了高级程序设计语言,操作系统已经用于计算机,使得脱机操作得以实现。并且,数据处理功能成为计算机应用的更广领域,软件数量不断膨胀,出现了许多始料不及的问题:要花费大量的时间和精力去修改程序中的错误,要忙于修改程序以满足用户提出的新的需求;为了跟上硬件平台的改变或操作系统的更新而不得不修改程序以适应新的环境。凡此种维护工作在整个计算机系统中所占的比重越来越大。更为严重的是:许多软件的个体化特征使得它们最终成为不可维护的。这样,大型软件的开发成本急剧上升,而质量越来越得不到保证,便开始出现了软件危机(Software Crisis)。

软件危机是计算机软件开发和维护过程中遇到的一系列严重的问题。这些问题主要表现在软件开发不能在规定的时间内、规定的成本预算内完成,并且质量没有保证。这些问题不仅仅限于所开发出的软件不能正常工作,还包括如何开发软件,如何维护越来越多的现有软件,以及如何满足不断增长的软件需求。软件危机的出现,究其原因:一方面是由软件本身的特点引起的,如软件越来越复杂、软件故障难于检测、工作量难以估计等;另一方面则是因为软件开发和维护方法不当所造成的,如软件的个体化特征太强等。与经济危机时期的“商品供过于求”的特点相比,软件危机却是“软件供不应求”。那么,如何摆脱软件危机呢?此时就提出了软件工程的观念。

为了摆脱软件危机所造成的困境,北大西洋公约组织(NATO)的科学委员会于 1968 年在联邦德国召开的有关研讨会上,首次提出了“软件工程”(Software Engineering)的概念,其主要思路是:要把人类长期以来从事各种工程项目所积累起来的行之有效的原理、概念、技术和方法,