

高等院校计算机基础教育规划教材

软件工程

主编 管建军 副主编 黄勇 张志勇



WUHAN UNIVERSITY PRESS

武汉大学出版社

内 容 简 介

本书着重从实用角度讲解软件工程的基本概念、基本原理和技术方法,同时也注意了该书的系统型和先进性。本书编写时除保证应用性强,特别注意语言精练、易于理解、可读性好的特点。为利于读者掌握重点和巩固学习内容,每章开始处提出本章的知识点和教学要求,每章最后都有小结和习题。软件工程是一门处于前沿地位的重要学科,并仍然处在迅速发展,学科内的新技术、新方法不断涌现。希望读者通过对本书的学习,能将理论知识应用于软件开发的实践,并在实践中不断创新。

软件工程是一门实践性很强的课程,只有通过软件开发的实践才能真正掌握和应用软件工程的理论知识,为此,在本书最后提供了一个综合性的设计型实验——“软件工程实训指导”,使读者在学习软件工程课程的同时,选择完成一个适当的项目或者自己所承担的实际课题,以软件工程实训的实践来促进软件工程理论的学习。

本书适合作为高等专科学校、高等职业学校、成人高等学校以及高等院校主办的二级职业技术学院计算机及相关专业学生使用的教材。目的在于培养学生的实际动手的能力,使得学生更加适合用人单位的技能要求。

图书在版编目(CIP)数据

软件工程/管建军主编. —武汉:武汉大学出版社,2007.5

高等院校计算机基础教育规划教材

ISBN 978-7-307-05565-0

I. 软… II. 管… 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(2007)第 057665 号

出版发行:武汉大学出版社 (430072 武昌 珞珈山)

(电子邮件:wdp4@whu.edu.cn 网址:www.wdp.com.cn)

印刷:北京市昌平百善印刷厂

开本:787×1092 1/16 印张:16.25

字数:411千字 印数:1~3000册

版次:2007年5月第1版 2007年5月第1次印刷

ISBN 978-7-307-05565-0/TP·247 定价:26.00元

版权所有,不得翻印;凡购买我社的图书,如有缺页、倒页、脱页等质量问题,请与当地图书销售部门联系调换。

高等院校计算机基础教育规划教材

编 委 会

主 任：赵云冲

副主任：涂兰敬 王懿媛 高锐 韦爱荣

编 委：（按姓氏笔画排序）

丁青云	凡大林	王 伟	王艳梅
王海梅	王 娟	王朝晖	王 辉
方美秀	白海波	邢宇飞	向 蕾
刘少华	刘年超	祁昌平	孙怀东
孙贤龙	花庆毅	杨希鹏	杨 涛
李运生	李 琴	沈 丹	张国良
陈 曦	金守兵	赵天广	赵红芬
胡 俊	胡胜利	咎 超	贾云娇
钱 勇	徐 楠	殷洪菊	黄宝龙
黄 磊	梁 辉	韩丽彦	程灵枝
程灵波	程宗米	解 平	熊化武

前言

PREFACE

软件工程是一门指导计算机软件系统开发和维护的工程学科,也是一门实践性很强的课程。它采用工程的概念、原理、技术和方法来开发和维护计算机软件,是各种开发经验的总结和提炼。它把经过时间考验而证明是正确的管理技术和当前能够得到的最好的开发技术和方法结合起来,从经济的角度开发出高质量的软件并加以维护。

本书共有 11 章。第 1 章介绍了软件的概念、发展和软件危机,着重介绍了软件工程的基本概念和基本内容及软件生存周期、软件开发模型等。第 2、3、4、7、8 章是本书的重点,分别论述可行性研究、需求分析、总体设计、详细设计、编码、测试、维护阶段的各种技术和方法,对系统流程图、数据流图、数据字典、层次图、HIPO 图、结构图、N-S 图、PAD 图、PDL 语言、黑盒法、白盒法等,逐一作了详细的介绍,读者开发软件时可根据需要灵活运用。为了保持教材内容的先进性,本书第 5、6 章介绍了面向对象软件工程学和统一建模语言 UML,主要包括面向对象方法的基本概念、面向对象的分析、面向对象的设计、软件复用、UML 的静态建模机制、UML 的动态建模机制等内容。第 9 章介绍了有关软件项目计划、软件项目组织、软件项目人员配备、软件配置管理和能力成熟度模型 CMM 等软件管理方面的内容。第 10 章结合所学课程,通过开发一个小型软件作为软件工程课程实训的示范课题,使读者以此为参照进行软件的设计和实现,完成软件工程实训任务。

软件工程是一门实践性很强的课程,只有通过软件开发的实践才能真正掌握和应用软件工程的理论知识。为此,在本书最后提供了一个综合性的设计型实验——“实训”,使读者在学习软件工程课程的同时,选择完成一个适当的项目或者自己所承担的实际课题,以软件工程实训的实践来促进软件工程理论的

学习。

本书着重从实用角度讲解软件工程的基本概念、基本原理和技术方法,同时也注意了该书的系统性和先进性。本书编写时除保证应用性强外,特别注意语言精练,具有易于理解、可读性强的特点。

软件工程是一门处于前沿地位的重要学科,并仍然处在迅速发展,学科内的新技术、新方法不断涌现。我们希望读者通过对本书的学习,能将理论知识应用于软件开发的实践,并在实践中不断创新。

本书适合作为高等专科学校、高等职业学校、成人高等学校以及高等院校主办的二级职业技术学院计算机及相关专业学生使用的教材。

本书在编写过程中参考了许多相关的著作,同时也得到有关方面专家和教师的帮助,笔者在此对这些著作的作者和专家一并表示深深的感谢。

由于作者水平有限,时间仓促,书中难免有表达不当和错误之处,衷心希望广大读者批评指正。

编 者

2007 年 3 月

目 录

CONTENTS

第 1 章 软件工程概述	1
1.1 软件	1
1.1.1 软件的定义	2
1.1.2 软件的特点	2
1.1.3 软件的产生与发展	3
1.2 软件危机	4
1.2.1 软件危机的产生	4
1.2.2 软件危机的表现与原因	5
1.3 软件工程	6
1.3.1 软件工程的定义	6
1.3.2 软件工程的目标	6
1.3.3 软件工程的原则	7
1.4 软件生存周期	7
1.5 软件开发模型	9
1.5.1 瀑布模型	10
1.5.2 快速原型模型	11
1.5.3 增量模型	12
1.5.4 喷泉模型	12
1.5.5 螺旋模型	13
1.6 实例解析	14
本章小结	15
习题 1	16
 第 2 章 可行性研究和需求分析	18
2.1 软件的可行性研究	18
2.1.1 可行性研究的任务	18
2.1.2 可行性研究步骤	19

2.1.3 可行性研究的文档	20
2.2 需求分析	21
2.2.1 需求分析的重要性	21
2.2.2 需求分析的任务	21
2.2.3 需求分析的步骤	23
2.3 系统流程图	24
2.4 数据流图	26
2.4.1 数据流图中的符号	26
2.4.2 设计数据流图的步骤	28
2.4.3 数据流图的绘制	29
2.5 数据字典	32
2.5.1 数据字典的内容及格式	32
2.5.2 数据字典的用途	34
2.5.3 数据字典的实现	34
2.6 实例解析	35
本章小结	36
习题 2	37

第 3 章 概要设计	39
3.1 概要设计的任务	39
3.1.1 概要设计的任务	39
3.1.2 概要设计说明书的主要内容	40
3.2 设计过程	40
3.3 设计原理	42
3.3.1 模块化	42
3.3.2 抽象与逐步求精	43
3.3.3 信息隐蔽	44
3.3.4 模块独立性	45
3.4 描绘软件结构的图形工具	47
3.4.1 软件结构图	47
3.4.2 层次图	49
3.5 启发规则	50
3.6 面向数据流的设计方法	52
3.6.1 变换分析	53
3.6.2 事务分析	54
3.6.3 设计优化	55
3.7 实例解析	55
本章小结	60
习题 3	60

第 4 章 详细设计	62
4.1 详细设计的任务与原则	62
4.1.1 详细设计的任务	62
4.1.2 详细设计的原则	63
4.2 结构化程序设计	63
4.2.1 结构化设计技术的形成	63
4.2.2 结构化设计技术的概念	64
4.2.3 结构化程序设计的原则	65
4.3 过程设计的工具	65
4.3.1 程序流程图 PFC	65
4.3.2 盒图 N-S	66
4.3.3 问题分析图 PAD	66
4.3.4 过程设计语言 PDL	67
4.3.5 判定表(Decision Table)	68
4.3.6 判定树	69
4.4 用户界面设计	70
4.4.1 用户界面应具备的特性及设计过程	70
4.4.2 用户界面的风格	70
4.4.3 用户界面的基本类型	72
4.4.4 用户界面设计指南	72
4.5 面向数据结构的设计方法	74
4.5.1 Jackson 方法简介	74
4.5.2 Jackson 方法的设计过程	75
4.6 程序复杂程度的定量度量	78
4.6.1 McCabe 度量法	78
4.6.2 Halstead 方法	79
4.7 实例解析	80
本章小结	81
习题 4	82
 第 5 章 面向对象的方法	 84
5.1 面向对象方法概述	84
5.1.1 传统软件工程方法存在的问题	84
5.1.2 面向对象方法概述	85
5.1.3 面向对象的基本概念	88
5.1.4 面向对象的开发方法	92
5.2 面向对象分析	94
5.2.1 面向对象分析基本过程及原则	95
5.2.2 确定对象	98
5.2.3 确定结构	99
5.2.4 确立主题	100

5.2.5 确定属性	100
5.2.6 确定服务和消息	101
5.3 面向对象设计	102
5.3.1 面向对象设计的准则	102
5.3.2 面向对象设计的内容	104
5.3.3 人机交互子系统的设计	105
5.3.4 问题域子系统的设计	106
5.3.5 任务管理子系统的设计	107
5.3.6 数据管理子系统的设计	108
5.4 实例解析	108
本章小结	112
习题 5	113
第 6 章 统一建模语言 UML	115
6.1 UML 概述	115
6.1.1 UML 的产生和发展	115
6.1.2 UML 的组成	116
6.1.3 UML 的主要特点	117
6.1.4 UML 的应用	117
6.2 通用模型元素	118
6.2.1 模型元素	118
6.2.2 约束	119
6.2.3 依赖关系	119
6.2.4 细化	120
6.2.5 注释	120
6.3 UML 的静态建模机制	120
6.3.1 用例图	121
6.3.2 类图和对象图	124
6.3.3 包	130
6.3.4 构件图	131
6.3.5 配置图	132
6.4 UML 的动态建模机制	132
6.4.1 消息	133
6.4.2 顺序图	133
6.4.3 协作图	134
6.4.4 状态图	135
6.4.5 活动图	137
6.5 实例解析	140
本章小结	144
习题 6	145

第7章 软件测试	147
7.1 编码	147
7.1.1 程序设计语言的选择	147
7.1.2 程序设计风格	149
7.2 软件测试基础	151
7.2.1 软件测试的目的	151
7.2.2 软件测试的特点和原则	152
7.2.3 软件测试的基本步骤	153
7.2.4 静态分析与动态测试	153
7.3 白盒测试技术	154
7.3.1 逻辑覆盖	154
7.3.2 基本路径测试	157
7.3.3 条件测试	158
7.3.4 循环测试	158
7.3.5 白盒法测试步骤的总结	159
7.4 黑盒测试技术	159
7.4.1 等价分类法	159
7.4.2 边界值分析法	160
7.4.3 错误推测法	161
7.4.4 因果图法	161
7.4.5 综合策略	161
7.5 软件测试过程	162
7.5.1 单元测试	162
7.5.2 集成测试	163
7.5.3 确认测试	165
7.5.4 系统测试	166
7.6 调试	166
7.6.1 调试的目的	166
7.6.2 调试方法	166
7.6.3 调试原则	167
7.7 实例解析	168
本章小结	171
习题7	171
 第8章 软件维护	 173
8.1 软件维护的内容	173
8.1.1 软件维护的定义	173
8.1.2 软件维护的类型及策略	174
8.2 软件维护的特点	175
8.3 软件维护过程	176
8.3.1 维护组织	176

8.3.2 维护工作的流程	177
8.3.3 维护技术	178
8.4 软件的可维护性	179
8.4.1 软件可维护性的定义	179
8.4.2 可维护性的度量	179
8.4.3 提高可维护性的方法	181
8.5 软件再工程过程	183
8.6 实例解析	185
本章小结	186
习题 8	187
 第 9 章 软件项目管理	 189
9.1 度量软件规模	189
9.1.1 面向规模的度量	190
9.1.2 面向功能的度量	190
9.2 成本估算	192
9.2.1 软件开发成本估计方法	192
9.2.2 成本估算模型	193
9.3 进度计划	194
9.3.1 进度安排的方法	194
9.3.2 制定开发进度计划	196
9.4 人员组织	196
9.5 质量保证	198
9.6 软件配置管理	200
9.6.1 软件配置项(简称 SCI)	200
9.6.2 基线	200
9.6.3 软件配置管理的过程	201
9.7 能力成熟度模型	203
9.7.1 CMM 概述	203
9.7.2 CMM 的五个级别详述	204
9.7.3 关键过程域	204
9.7.4 应用软件过程评估	206
9.8 实例解析	206
本章小结	207
习题 9	207
 第 10 章 综合应用	 209
10.1 “图书馆管理信息系统”的立项背景	209
10.2 可行性研究	210
10.2.1 现有系统存在的问题	210
10.2.2 新系统的功能	210

10.2.3	软件开发环境	211
10.2.4	可行性分析	211
10.3	需求分析	212
10.3.1	需求分析概述	212
10.3.2	组织结构调查	212
10.3.3	系统用户分析	213
10.3.4	UML 用例图建模	213
10.3.5	数据流图	213
10.3.6	数据字典	215
10.3.7	IPO 图	218
10.3.8	用户其他需求	219
10.4	概要设计	219
10.4.1	系统结构设计	219
10.4.2	数据库设计	221
10.5	详细设计	226
10.5.1	登录模块详细设计	226
10.5.2	读者管理子系统详细设计	227
10.6	系统实现	231
10.6.1	登录窗体	231
10.6.2	读者管理	233
10.7	系统测试	235
10.7.1	功能测试	235
10.7.2	系统测试	238
10.7.3	测试结论	238
	本章小结	238
第 11 章	实训	239
11.1	实训指导	239
11.2	实训流程	240
11.3	实训内容	242
实训 1	学校排课系统	242
实训 2	学校教材订购系统	242
实训 3	机票预定系统	243
实训 4	学生公寓管理系统	243
实训 5	实训室设备管理系统	243
参考文献		245

第 1 章 软件工程概述

你知道吗?

- 软件发展经历了哪几个阶段?
- 软件危机主要有哪些表现? 其产生的原因是什么?
- 什么是软件工程? 它有哪些原则?
- 软件工程目标和面临的主要问题有哪些?
- 软件生存周期包括哪几个阶段? 常见的软件开发模型有哪些?

学习目标

- 软件工程的定义、目标和原则
- 软件生存周期的概念、各个阶段应解决的问题及相应的输出
- 常见的软件开发方法

1946 年世界上第一台电子计算机的诞生,标志着人类由工业化社会进入了信息化社会,以计算机产业和计算机应用服务业为支柱的信息工业,成了信息化社会的主要基础之一。而软件又是信息化的核心,软件产业是增长最快的朝阳产业,是高投入/高产出、无污染、低能耗的绿色产业。软件产业关系到国家经济发展和文化安全,体现了国家的综合实力,是决定 21 世纪国际竞争地位的战略性的产业。

随着计算机的日益普及和广泛应用,软件系统的规模和复杂度与日俱增,软件技术面临着新的挑战。大型复杂软件的开发是一项特殊的工程,不仅与传统工程一样,需要按照工程化的方法去组织管理软件的开发,而且软件开发更具特殊性、复杂性。软件工程是在克服 20 世纪 60 年代末所出现的“软件危机”的过程中逐渐形成与发展的。自 1968 年在北大西洋公约组织(NATO)举行软件可靠性的学术会议上正式提出软件工程(Software Engineering,简称为 SE)的概念以来,在不到 40 年的时间里,软件工程在理论和实践两方面都取得了长足的进步。

1.1 软件

软件是一种产品,同时又是开发和运行产品的载体。作为一种产品,它表达了由计算机硬件体现的计算潜能。不管它是驻留在设备中,还是在主机中,软件是一个信息转换器,能够产生、管理、获取、修改、显示或转换信息。这些信息可以很简单,如一个 bit,也可以很复杂,如多媒体信息。作为开发和运行产品的载体,软件是计算机工作的基础、信息通信的基础,也是创建和控制其他程序的基础。

信息是 21 世纪最重要的产品,软件充分地体现了这一点。软件处理数据,使得这些数据更为有用;软件管理商业信息,增强了商业竞争力;软件提供了通往全球信息网络的途径,

也提供了以各种形式获取信息的手段。

1.1.1 软件的定义

“软件”一词是在 20 世纪 60 年代出现的,它的定义是:计算机程序及其说明程序的各种文档。其中,“程序”是计算任务的处理对象和处理规则的描述,即按既定算法,用某种语言规定的指令或语句编写的指令或语句的集合;“文档”是有关计算机程序功能、设计、编制、使用的文字或图形资料。软件与硬件一起构成完整的计算机系统,它们相互依存、缺一不可,计算机系统是通过运行程序来实现各种不同的应用。

随着计算机应用的日益普及,软件变得越来越复杂,规模也越来越大,这就使得人与人、人与机器间相互沟通,保证软件开发与维护工作的顺利进行显得特别重要。因此,文档(即各种报告、说明、手册的总称)是不可缺少的。特别是在软件日益成为产品的今天,文档的作用就更加重要。

1.1.2 软件的特点

在计算机系统中,软件是一个逻辑部件,而硬件是一个物理部件。因此,软件相对硬件而言有以下特点:

1. 软件区别于物质产品

硬件是看得见摸得着的,而软件产品是看不见摸不着的,具有无形性。它是脑力劳动的结晶,以程序和文档的形式出现,保存在计算机存储器的磁盘和光盘介质上。软件正确与否,要在机器上运行才能知道,只能通过机器的执行才能体现它的功能和作用。

2. 软件与硬件的生产方式不同

软件产品的生产过程是研制,软件产品的成本主要体现在软件的开发和研制上。软件是通过人们的智力活动,把知识与技术转化成信息的一种产品,是在研制、开发中被创造出来的。一旦某一软件项目研制成功,以后就可以大量地复制同一内容的副本,所以其研制成本远远大于其生产成本。软件故障往往是在开发时产生而在测试时没有发现的问题,所以要保证软件的质量,必须着重于软件开发过程,加强管理和减少故障。

3. 软件故障的产生与硬件不同

在软件的运行和使用期间,没有硬件那样的机械磨损、老化问题。软件维护比硬件维护要复杂得多,与硬件的维护有着本质的差别。如图 1-1 和图 1-2 所示的是硬件和软件故障率随时间变化的曲线。

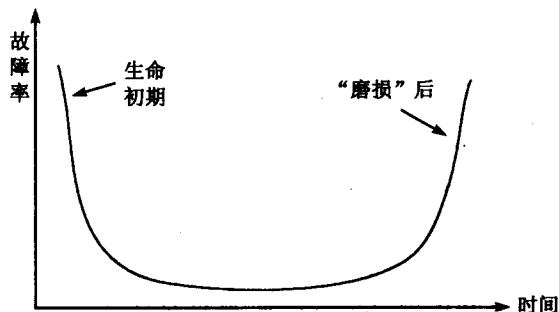


图 1-1 硬件故障率曲线

图 1-1 给出了硬件的故障率曲线,它是一个 U 型曲线(即浴盆曲线),说明硬件随着使用

时间的增加故障率急剧上升。

图 1-2 所描述的软件故障率曲线,它没有 U 型曲线的右半翼,表明软件随着使用时间的增加故障率降低。软件不存在磨损和老化问题,然而却存在退化问题。这是由于软件开发时的需求环境、软硬件环境在不断变化,必须多次修改软件,而修改软件则不可避免地会引入新的错误,导致软件的故障率升高,从而使得软件可靠性下降。当修改的成本变得难以接受时,软件就被抛弃。

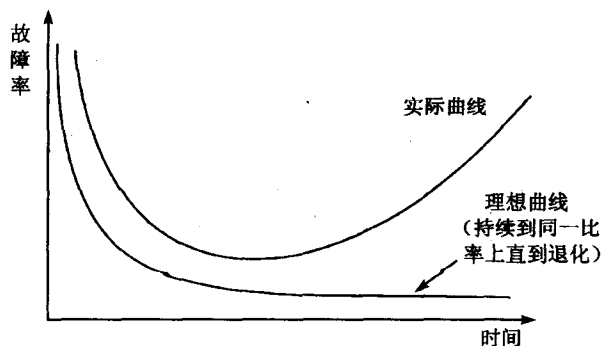


图 1-2 软件故障率曲线

4. 软件的复杂性

软件的复杂性一方面来自它所反映实际问题的复杂性;另一方面,也来自程序结构的复杂性。软件技术的发展落后于复杂的软件需求,并且随着时间的推移,这个差距日益加大。

5. 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,研制成本是比较高的。在 20 世纪 50 年代末,软件开销约占总开销的百分之十几,大部分成本花费在硬件上,但在这个比例完全颠倒过来,软件的开销大大超过硬件的开销。

1.1.3 软件的产生与发展

随着计算机硬件性能的极大提高和计算机体系结构的不断变化,同其他事物的发展规律一样,计算机软件系统也经历了从产生、发展到成熟的过程,从而促使计算机软件的角色发生了巨大的变化,其发展历史大致可以分为四个阶段。

1. 程序设计阶段

计算机发展早期阶段(20 世纪 50 年代初期至 20 世纪 60 年代中期)为程序设计阶段。在这个阶段硬件已经通用化,而软件的生产却是个体化的。这时,由于程序规模小,几乎没有什么系统化的标准方法可遵循。对软件的开发没有任何管理方法,一旦任务超时或者成本提高,程序员才开始弥补。在通用的硬件已经非常普遍的时候,软件却相反,对每一类应用均需自行再设计,应用范围很有限。软件产品处在初级阶段,大多数软件都是由使用者自己开发,因为是个人化的软件环境。设计往往仅是人们头脑中的一种模糊想法,而根本就不存在文档。

2. 程序系统阶段

计算机系统发展的第二阶段(20 世纪 60 年代中期到 70 年代末期)为程序系统阶段。多道程序设计、多用户系统引入了人机交互的新概念。交互技术打开了计算机应用的新世界以及硬件和软件配合的新层次,实时系统和第一代数据库管理系统出现了。这个阶段的另

一个特点就是软件产品的使用和“软件作坊”的出现。开发出的软件可以在较宽广的范围中应用,主机和微机上的程序能够有数百甚至上千个用户。

在软件的使用中,当发现错误时需要纠正程序源代码;当用户需求发生变化时需要修改软件;当硬件环境变化时需要适应。这些活动统称为软件维护。在软件维护上所花费的精力以惊人的速度消耗资源。更为严重的是,许多程序的个性化特性使得根本不能维护它们。于是,“软件危机”出现了。

3. 软件工程阶段

计算机系统发展的第三阶段始于 20 世纪 70 年代中期并跨越了近十年,称为软件工程阶段。在这一阶段,以软件的产品化、系列化、工程化、标准化为特征的软件产业发展起来,打破了软件生产的个体化特征,有了软件工程化的设计原则、方法、标准可以遵循。在分布式系统中,各台计算机同时执行某些功能,并与其他计算机通信,极大地提高了计算机系统的复杂性。广域网、局域网、高带宽数字通信以及对即时数据访问需求的增加都对软件开发提出了更高的要求。

4. 第四阶段

计算机发展的第四阶段已经不再着重于单台计算机系统和程序,而是面向计算机的综合应用。由复杂操作系统控制的强大的桌面机、广域网络和局域网络,配以先进的软件应用已成为标准。计算机体系结构迅速地集中从集中的主机环境转变为分布的客户/服务器环境。世界范围的信息网提供了一个基本结构,信息高速公路和网际空间连通已成为令人关注的热点问题。事实上,Internet 可以看作是能够被单个用户访问的软件,计算机发展正朝着社会信息化和软件产业化方向发展,从技术的软件工程阶段过渡到社会信息化的计算机系统阶段。随着第四阶段的发展,一些新技术开始出现。面向对象技术将在许多领域中迅速取代传统软件开发方法。

图 1-3 给出了四个发展阶段典型技术的比较。

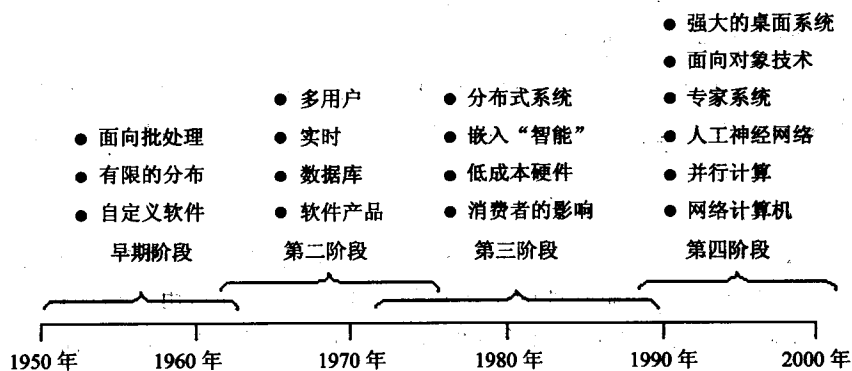


图 1-3 四个阶段典型技术

1.2 软件危机

1.2.1 软件危机的产生

软件发展到第二阶段末期,由于计算机硬件技术的进步,计算机运行速度、容量、可靠性显著提高,生产成本显著下降,这就为计算机的广泛应用创造了条件。一些复杂的、大型的

软件开发项目提了出来,但是软件开发技术一直不能跟上硬件技术的进步,不能满足发展的需求。在软件开发过程中遇到的问题找不到解决的办法,使问题积累起来,形成了尖锐的矛盾,因而导致了软件危机。

1962年6月,美国飞往金星的第一个空间探测器(水手I号),因其飞舱中计算机导航程序的一条语句出错,致使空间探测器偏离航线无法取得成功。还有,可以称为上世纪世界上最精心设计,并花费了巨额投资的美国阿波罗登月飞行计划的软件,也仍然没有避免出错。例如,阿波罗8号由于太空飞船的一个计算机软件错误,造成存储器的一部分信息丢失;阿波罗14号在飞行的10天中,出现了18个软件错误。

1.2.2 软件危机的表现与原因

1. 软件危机的表现

软件危机指的是软件开发和维护过程中遇到的一系列严重问题。软件危机包含下述两方面的问题:如何开发软件,怎样满足对软件的日益增长的需求;如何维护数量不断膨胀的已有软件。具体地说,软件危机主要有下列表现:

(1) 经费预算经常突破,完成时间一再拖延

由于缺乏软件开发的经验和软件开发数据的积累,使得开发工作的计划很难制定。主观盲目制定的计划,执行起来和实际情况有很大差距,使得开发经费一再突破。由于对工作量和开发难度估计不足,进度计划无法按时完成,开发时间一再拖延。

(2) 开发的软件不能满足用户要求

开发初期对用户的要求了解不够,未能得到明确表达。开发工作开始后,软件人员和用户又未能及时交换意见,使得一些问题不能及时解决,导致开发的软件不能满足用户的要求,使开发失败。

(3) 开发的软件可维护性差

开发过程没有统一的、公认规范,软件开发人员按各自的风格工作,各行其事。开发过程无完整、规范的文档,发现问题后进行杂乱无章的修改。程序结构不好,运行时发现错误也很难修改,导致维护性差。

(4) 开发的软件可靠性差

由于在开发过程中,没有确保软件质量的体系和措施,在软件测试时,又没有严格的、充分的、完全的测试,提交给用户的软件质量差,在运行中暴露出大量的问题。这种不可靠的软件,轻者会影响系统正常工作,重者会发生事故。

2. 软件危机的原因

造成上述软件危机的原因是由于软件产品本身的特点以及开发软件的方式、方法、技术和人员引起的。

(1) 软件的规模越来越大、结构越来越复杂。随着计算机应用的日益广泛,需要开发的软件规模日益庞大,软件结构也日益复杂。1968年美国航空公司订票系统达到30万条指令;IBM 360 OS第16版达到100万条指令,花了5000人/年;1973年美国阿波罗计划达到1000万条指令。这些庞大软件的功能非常复杂,体现在处理功能的多样性和运行环境的多样性。有人曾估计,软件设计与硬件设计相比,其逻辑量要多达10~100倍。对于这种大规模软件的复杂性,其调用关系和接口信息复杂,数据结构也复杂,这种复杂程度超过了人能接受的程度。

(2) 软件开发管理困难。由于软件规模大,结构复杂,又具有无形性,因此导致管理困