

138

7P 311.5-43
C 476

现代远程教育计算机信息技术教材

Modern
Software
Engineering

现代软件工程

陈松乔 任胜兵 王国军 编著

北方交通大学出版社

Northern Jiaotong University Press

北京 • BEIJING

内 容 简 介

本书系统地介绍了软件工程的有关概念、原理、方法、技术、标准和相关法律法规。全书共 10 章,以软件生存周期为主线,对软件工程有关的分析、设计、验证、维护和管理等方面内容做了详尽阐述,突出结构化技术、面向对象技术和组件技术在软件开发过程中的运用,强调软件产品质量和软件过程质量的分析与保证,重视软件工程标准化和软件知识产权对软件工程的影响。

全书内容新颖,讲述力求理论联系实际、深入浅出、循序渐进。每章均附有习题。

本书主要用做计算机科学与技术专业本科或研究生“软件工程”课程的远程教育教材,亦可作为高等院校计算机科学与技术专业或其他相关专业的教学用书,或作为从事软件开发的科技人员的参考书、培训教材等。

图书在版编目(CIP)数据

现代软件工程/陈松乔,任胜兵,王国军编著. —北京:北方交通大学出版社,2002.1

现代远程教育计算机信息技术教材

ISBN 7-81082-016-8

I. 现... II. ①陈...②任...③王... III. 软件工程—远距离教育—教材 IV. TP311.5

中国版本图书馆 CIP 数据核字(2001)第 091915 号

丛 书 名: 现代远程教育计算机信息技术教材

书 名: 现代软件工程

编 著 者: 陈松乔 任胜兵 王国军

责任编辑: 孙秀翠

排版制作: 鑫鑫达电脑打印中心

印 刷 者: 李史山胶印厂

装 订 者:

出版发行: 北方交通大学出版社 邮编: 100044 电话: 010-62237564

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 21.75 字数: 543 千字

版 次: 2002 年 1 月第 1 版 2002 年 1 月第 1 次印刷

书 号: ISBN 7-81082-016-8
TP·8

印 数: 5000 册 定价: 28.00 元

现代远程教育计算机信息技术教材

编委会成员名单

主 任：王柯敏

副主任：邱光谊 郑光信

委 员：（以姓氏笔划为序）

王 洪	卢先河	阮秋琦	杨贯中	闵应骅
邹北骥	张大方	张凤祥	张晨曦	陈 庚
罗 安	洪范文	胡峰松	柳军飞	袁开榜

总 序

湖南大学直属教育部，是国家“211工程”立项建设的全国重点大学之一，其渊源可追溯至中国古代著名的四大书院之一的岳麓书院，素有“千年学府”之称。在漫长的办学历程中，湖南大学逐步形成和发展了“爱国务实、严谨勤奋、民主团结、求是创新”的优良传统，造就了一大批经世致用之才。

为了适应社会主义现代化建设对高素质专门人才的需要，湖南大学积极发展多种形式的高等教育。学校1997年开始探索利用计算机网络开展现代远程教育这一全新领域。在多次远程网上教学实验获得成功的基础上，于当年10月与湖南省邮电管理局联合成立了湖南大学多媒体信息教育学院。

现代远程教育是在计算机及网络技术的支持下，允许教师和学生异地，实时或非实时地以文本、图形/图像、音频、视频等形式进行交互式的教学活动，它突破了时间、空间的限制，使教学的规模更为扩大，并为普及高质量的教育提供了一种有效的手段，同时又能满足社会对终身教育的需要。为此教育部启动了国家建设现代远程教育工程，布置了一批高等学校及部分中小学开展远程教育，以推动教育体制进一步改革。

1998年10月，湖南大学获教育部批准，开展现代远程教育试点，同时面向全省招收了第一批网上大学生。招生专业包括计算机科学与技术、英语、计算机应用（专科）、建筑工程（专科）、经济管理（专科）等。通过3年多的探索与实践，湖南大学基本上建立了适合我国国情的现代远程教育管理模式和运行机制，形成了学历与非学历教育相结合，校内的校园网多媒体教学与校外远程教学同时进行并相互融合的开放式办学格局，网上大学已具雏形。

远程教育与传统的面对面的教育方式不同，它更强调学生以自主的个性化学习为主，因此需要提供更适合于自学的教材，同时还要提供内容丰富的多媒体教学课件、电子教案、学习指导书、学习进度与时间安排表等，以支持远程教学活动。

为进一步推动现代远程教育事业的发展，湖南大学组织了一批具有网上教学经验的年富力强的优秀教师，编写了这套现代远程教育计算机信息技术教材。本套教材是根据教育部审定批准的教学大纲编写的，适合高等教育的教学及学生学习，尤其适合我国现代远程教育的本（专）科学生学习使用。

现代远程教育计算机信息技术教材编委会

2002年1月

前 言

从软件工程概念正式提出至今,已经经历了将近40年的发展,取得了大量的理论和应用研究成果,形成和促进了软件产业的飞速发展,在社会生活中扮演着愈来愈重要的角色。正是由于学术界和产业界的不懈努力,软件工程正逐步发展成为一门成熟的专业学科。

远程教育以网络为依托,以多媒体为手段,以资源共享为目的,努力开展教学思想、教学手段、教学方法和教学内容的改革,为科教兴国提供了一种有力的途径。本教材正是为了适应远程教育的需要,由从事软件工程教学和实践经验丰富的教师编写而成。

本书系统地介绍了软件工程的有关概念、原理、方法、技术、标准和相关法律法规,其目标是使学生理解有关软件生存周期模型概念和软件工程基本原理,掌握建造软件系统的有关方法、技术和标准(规范),了解“软件工程师”职业的有关行为准则。本课程是一门工程性课程,实践性很强,学生在学习时除了概念、原理等的理解之外,更应结合实际,注重方法、技术等理解和运用。

软件工程近年来发展非常迅速,像并行工程、UML建模语言、三层应用体系结构、CORBA/COM/DCOM等组件技术、软件复用技术、CMM模型等,本书均作了详细介绍。在介绍新概念、新方法、新技术时,本书力争与已有的内容相融合,一方面可以加深理解,另一方面可使本书的所有内容能形成一个统一的整体。同时,本书为了突出表现软件工程的工程性质,强调计划、分析、设计、质量保证、标准(规范)和知识产权保护等内容,没有详细介绍编程的有关内容。

全书共分10章。第1章,简述软件工程的产生、软件工程学的研究对象与原则、软件开发方法、软件工程的生存周期模型以及软件工程发展的新方向。第2章,阐述软件需求分析的任务和要求、可行性研究的任务以及系统建模方法。第3~5章,叙述软件开发阶段的任务、过程、方法和技术。第6章,介绍软件质量的有关概念、分析技术和质量保证体系等。第7章,着重讨论软件维护阶段的有关活动、提高软件可维护性的策略等。第8~9章,主要涉及有关软件工程的管理技术。第10章,简要介绍有关软件的法律保护和职业道德规范的问题。如果教学时间有限,则可以只讲授第1~5章和第7~8章,其内容依然是完整的。为了便于学生的学习,每章均附有相当数量的练习题。

全书由博士生导师陈松乔教授组织编著和定稿,前4章的大部分内容 by 王国军博士编著,其余各章节由任胜兵博士编著。

由于作者水平所限,加之时间仓促,书中错误和不当之处在所难免,敬请各位读者批评指正。

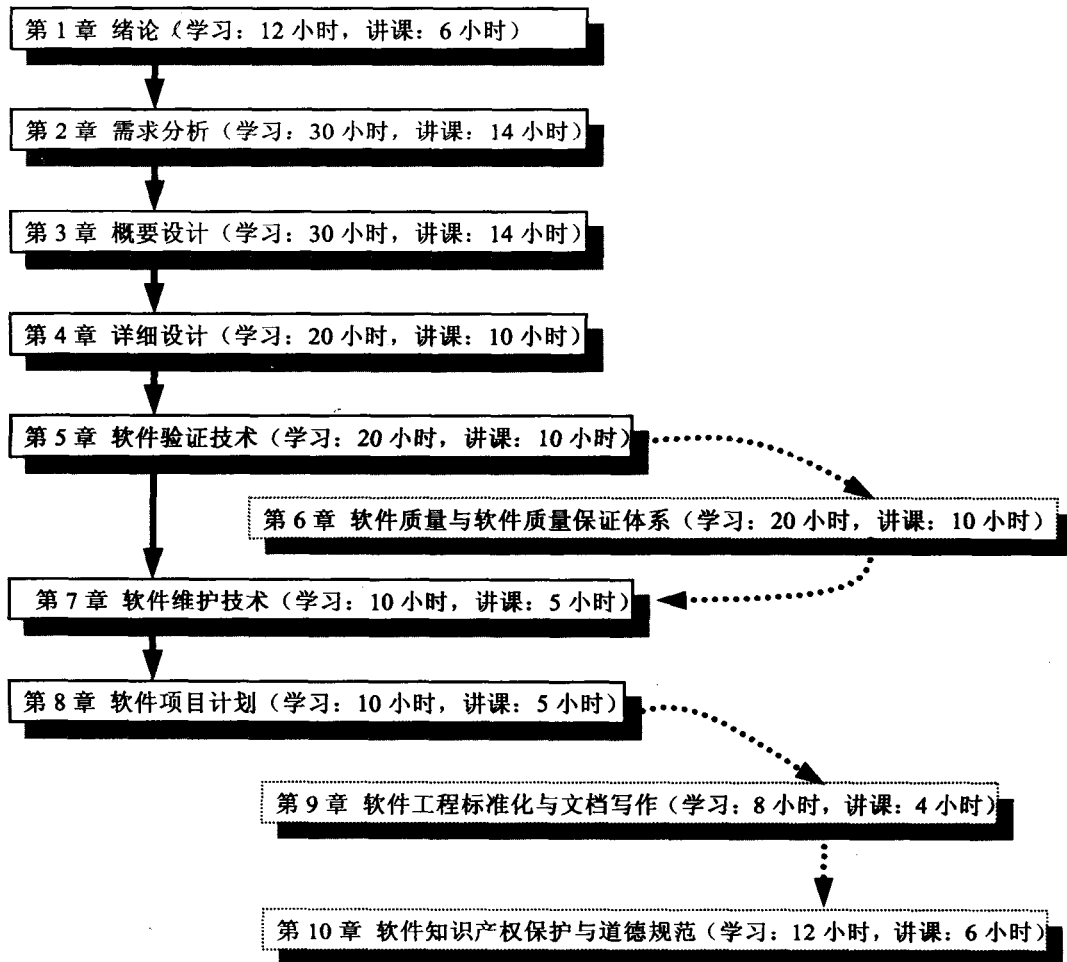
作 者

2002年1月于中南大学

学 习 指 南

1. 先修课程：数据结构、程序设计语言、操作系统、数据库系统、计算机网络等；
2. 参考学时与学分：建议学习时限约为 172 小时，其中讲课 84 小时，本课程计 3 学分；
3. 学习进度安排与学习流程：

如果时间有限，可以根据需要只学习第 1 至第 5 章和第 7、8 章，讲课需 64 小时。



目 录

第 1 章 绪论	(1)
1.1 软件工程的产生	(1)
1.1.1 软件的特征	(1)
1.1.2 软件危机的出现	(3)
1.1.3 软件工程的提出	(3)
1.2 软件工程的研究对象与基本原理	(3)
1.2.1 软件工程的研究对象	(4)
1.2.2 软件工程的层次结构	(4)
1.2.3 软件工程的基本原理	(5)
1.3 软件开发方法	(5)
1.3.1 结构化方法	(5)
1.3.2 面向对象方法	(6)
1.3.3 形式化方法	(7)
1.3.4 结构化方法、面向对象方法和形式化方法的结合研究	(7)
1.4 软件工程工具和环境	(8)
1.4.1 软件工具	(8)
1.4.2 CASE 集成环境	(9)
1.5 软件工程常用生存周期模型	(10)
1.5.1 瀑布模型	(11)
1.5.2 快速原型化模型	(12)
1.5.3 快速应用开发模型	(14)
1.5.4 渐增模型	(15)
1.5.5 螺旋模型	(16)
1.6 软件工程的并行工程	(17)
1.6.1 簇的基本概念	(18)
1.6.2 并行工程的提出	(18)
1.6.3 簇的基本步骤和任务	(19)
1.6.4 簇模型与并行工程	(19)
1.6.5 无缝开发和可逆性的基本概念	(21)
1.7 小结	(21)
习题	(22)

第2章 需求分析	(23)
2.1 需求分析的目标	(23)
2.2 可行性分析	(23)
2.2.1 可行性分析的内容	(24)
2.2.2 可行性分析的基本步骤	(25)
2.2.3 系统流程图	(25)
2.3 需求收集	(27)
2.3.1 需求收集的内容	(27)
2.3.2 需求收集的方法	(27)
2.4 需求规格说明	(28)
2.5 数据流建模	(29)
2.5.1 数据流图中的常用符号	(29)
2.5.2 数据流图的用途	(32)
2.5.3 数据词典和加工说明	(33)
2.6 实体-关系建模	(36)
2.6.1 实体-关系图	(37)
2.6.2 基数和模态	(38)
2.6.3 实体-关系图的例子	(39)
2.7 系统行为建模	(40)
2.7.1 处理模型和控制模型的关系	(41)
2.7.2 一个控制流图的例子	(42)
2.7.3 一个状态转移图的例子	(43)
2.8 IDEF0 功能建模	(44)
2.8.1 IDEF0 图	(44)
2.8.2 IDEF0 建模步骤	(49)
2.8.3 绘制活动图形的注意事项	(49)
2.9 IDEF1X 数据建模	(50)
2.9.1 IDEF1X 图	(50)
2.9.2 IDEF1X 建模步骤	(55)
2.10 面向对象建模	(56)
2.10.1 面向对象建模的发展	(56)
2.10.2 面向对象基础	(58)
2.10.3 面向对象分析模型	(64)
2.10.4 对象模型的建立	(65)
2.10.5 行为模型的建立	(67)
2.10.6 功能模型的建立	(72)
2.11 统一建模语言 UML	(75)

2.11.1	UML 的基本实体	(76)
2.11.2	UML 的目标	(77)
2.11.3	UML 的范畴	(77)
2.11.4	UML 图的使用实例	(77)
2.12	小结	(83)
	习题	(83)
第 3 章	概要设计	(85)
3.1	概要设计的任务和过程	(85)
3.1.1	数据设计(对象设计)	(86)
3.1.2	软件结构设计(子系统设计)	(87)
3.1.3	接口设计(消息设计)	(87)
3.1.4	过程设计(方法设计)	(87)
3.2	软件设计基本原则	(87)
3.2.1	抽象与求精	(87)
3.2.2	模块化和信息隐藏	(89)
3.2.3	模块独立性	(90)
3.3	面向数据流图的设计方法	(93)
3.3.1	软件结构的图形表示	(93)
3.3.2	数据流分析	(95)
3.3.3	事务分析	(98)
3.3.4	软件结构的改进	(100)
3.4	面向对象的设计方法	(101)
3.4.1	软件复用	(101)
3.4.2	面向对象设计	(104)
3.5	小结	(110)
	习题	(110)
第 4 章	详细设计	(112)
4.1	详细设计的目标与任务	(112)
4.1.1	详细设计的目标	(112)
4.1.2	详细设计的任务	(112)
4.2	结构化程序的详细设计技术与工具	(113)
4.2.1	结构化程序的详细设计技术	(113)
4.2.2	结构化程序的详细设计工具	(114)
4.3	Jackson 程序设计方法	(119)
4.3.1	Jackson 图	(119)

4.3.2	Jackson 伪代码	(120)
4.3.3	Jackson 程序设计方法的步骤	(121)
4.4	Warnier 程序设计方法	(125)
4.4.1	Wariner 图	(126)
4.4.2	Warnier 程序设计步骤	(126)
4.5	面向对象程序的详细设计	(131)
4.5.1	面向对象程序的特性	(131)
4.5.2	设计原则	(138)
4.6	基于组件的程序设计方法	(140)
4.6.1	三层应用体系结构	(140)
4.6.2	CORBA 对象总线技术	(143)
4.6.3	COM 对象总线技术	(146)
4.6.4	组件库技术	(148)
4.7	小结	(152)
	习题	(152)

第 5 章 软件验证技术 (154)

5.1	软件测试基础	(154)
5.1.1	测试观点	(154)
5.1.2	测试原则	(155)
5.1.3	测试工具	(155)
5.1.4	测试组织	(156)
5.1.5	测试与调试	(157)
5.1.6	动态测试步骤	(157)
5.2	代码复审	(158)
5.2.1	代码复审内容	(158)
5.2.2	代码会审	(159)
5.2.3	走查	(159)
5.2.4	办公桌检查	(159)
5.3	白盒测试	(160)
5.3.1	逻辑覆盖法	(160)
5.3.2	基本路径覆盖法	(162)
5.3.3	循环覆盖法	(165)
5.4	黑盒测试	(166)
5.4.1	等价分类法	(166)
5.4.2	边界值分析法	(168)
5.4.3	猜错法	(169)

5.4.4 因果图法	(169)
5.5 单元测试	(172)
5.5.1 测试策略	(173)
5.5.2 测试内容	(173)
5.5.3 测试的阶段及活动	(174)
5.5.4 测试软件	(175)
5.6 集成测试	(176)
5.6.1 测试内容	(176)
5.6.2 测试策略	(177)
5.6.3 非渐增式测试	(177)
5.6.4 渐增式测试	(177)
5.6.5 回归测试	(180)
5.7 确认测试	(180)
5.7.1 确认测试内容	(180)
5.7.2 α 测试和 β 测试	(181)
5.8 系统测试	(182)
5.8.1 恢复测试	(182)
5.8.2 安全性测试	(182)
5.8.3 可用性测试	(183)
5.8.4 安装测试	(183)
5.8.5 互连测试	(183)
5.9 程序正确性证明	(183)
5.9.1 程序正确性定义	(184)
5.9.2 Floyd 不变式断言法	(184)
5.9.3 Floyd 良序集方法	(187)
5.9.4 程序正确性证明的局限性	(189)
5.10 调试	(189)
5.10.1 调试的步骤	(189)
5.10.2 调试的策略	(190)
5.10.3 调试的原则	(191)
5.11 小结	(191)
习题	(192)

第 6 章 软件质量和软件质量保证体系

6.1 软件质量	(195)
6.1.1 软件质量特性	(195)
6.1.2 软件质量评价	(196)

6.1.3	软件质量保证	(197)
6.1.4	软件质量管理	(198)
6.2	软件复杂性分析	(199)
6.2.1	基于需求分析的复杂性分析	(199)
6.2.2	基于软件设计的复杂性分析	(203)
6.2.3	基于源程序代码的复杂性分析	(206)
6.2.4	基于软件维护的复杂性分析	(208)
6.3	软件可靠性分析	(209)
6.3.1	软件可靠性三要素	(209)
6.3.2	软件可靠性模型	(211)
6.3.3	软件可靠性工程	(213)
6.4	ISO 9000 软件质量体系	(215)
6.4.1	ISO 9000 族国际标准	(216)
6.4.2	企业软件质量体系的建立和实施	(218)
6.5	软件配置管理	(221)
6.5.1	软件配置项	(221)
6.5.2	软件配置管理	(222)
6.6	软件过程能力成熟度模型简介	(225)
6.6.1	基本概念	(225)
6.6.2	软件过程能力成熟度等级	(226)
6.6.3	关键过程域	(228)
6.6.4	关键实践	(230)
6.6.5	软件过程能力成熟度模型的应用	(231)
6.7	小结	(232)
	习题	(233)

第 7 章	软件维护技术	(235)
7.1	软件维护的基本概念	(235)
7.1.1	软件维护的类型	(235)
7.1.2	软件维护的策略	(236)
7.1.3	软件维护的成本	(236)
7.1.4	软件维护的副作用	(238)
7.1.5	软件维护的困难	(238)
7.2	软件维护的过程	(239)
7.2.1	维护组织	(239)
7.2.2	维护申请	(240)
7.2.3	维护工作的流程	(240)

7.2.4	维护记录与评价	(241)
7.3	软件可维护性	(242)
7.3.1	影响软件可维护性的软件属性	(242)
7.3.2	软件可维护性的定量度量	(243)
7.3.3	提高软件可维护性的方法	(243)
7.4	软件再工程技术	(244)
7.4.1	逆向工程	(244)
7.4.2	重构	(245)
7.4.3	正向工程	(245)
7.4.4	再工程的成本/效益分析	(245)
7.4.5	再工程的风险分析	(246)
7.5	小结	(246)
	习题	(247)

第 8 章	软件项目计划	(248)
8.1	成本估计	(248)
8.1.1	自顶向下成本估计	(248)
8.1.2	自底向上成本分析	(249)
8.1.3	基于经验模型的成本估计	(250)
8.2	效益分析	(255)
8.2.1	度量方法	(255)
8.2.2	效益分析	(256)
8.3	风险分析	(257)
8.3.1	风险标识	(258)
8.3.2	风险估计	(259)
8.3.3	风险评价	(259)
8.3.4	风险管理与监控	(260)
8.4	进度安排	(260)
8.4.1	进度安排原则	(261)
8.4.2	工作量分配	(261)
8.4.3	进度安排方法	(262)
8.5	项目组织与计划	(265)
8.5.1	人员组织规律	(265)
8.5.2	人员组织形式	(267)
8.5.3	项目计划	(269)
8.6	小结	(270)
	习题	(270)

第 9 章 软件工程标准化与文档写作	(272)
9.1 软件工程标准	(272)
9.1.1 软件工程标准的类型	(272)
9.1.2 软件工程标准的层次	(273)
9.1.3 软件工程国家标准	(274)
9.2 软件文档的编写步骤	(275)
9.2.1 软件文档的含义及要求	(275)
9.2.2 软件文档的种类	(275)
9.2.3 软件文档的编写步骤	(277)
9.3 软件文档的主要内容及写作要求	(278)
9.3.1 可行性研究报告的主要内容及写作要求	(278)
9.3.2 项目开发计划的主要内容及写作要求	(279)
9.3.3 软件需求说明书的主要内容及写作要求	(280)
9.3.4 概要设计说明书的主要内容及写作要求	(281)
9.3.5 详细设计说明书的主要内容及写作要求	(281)
9.3.6 其他文档	(282)
9.4 软件文档的管理	(283)
9.4.1 软件文档编写过程的管理	(283)
9.4.2 软件文档使用过程的管理	(284)
9.5 小结	(285)
习题	(285)
第 10 章 软件知识产权保护与道德规范	(286)
10.1 知识产权	(286)
10.1.1 知识产权的概念	(286)
10.1.2 知识产权的法律体系	(287)
10.1.3 知识产权的主要特点	(287)
10.1.4 知识产权的国际保护	(288)
10.1.5 计算机软件知识产权的保护	(288)
10.2 软件著作权	(288)
10.2.1 计算机软件著作权法的保护	(289)
10.2.2 软件著作权的客体	(289)
10.2.3 软件著作权的主体	(290)
10.2.4 软件著作权的权利内容	(290)
10.2.5 软件著作权的权利限制	(291)
10.2.6 软件著作权的取得	(292)

10.2.7 软件著作权的归属·····	(292)
10.2.8 软件著作权的行使·····	(292)
10.2.9 案例分析·····	(293)
10.3 软件著作权登记·····	(293)
10.3.1 实施计算机软件著作权登记的作用·····	(294)
10.3.2 软件著作权登记的形式·····	(294)
10.3.3 软件著作权登记程序·····	(295)
10.4 软件商业秘密·····	(296)
10.4.1 商业秘密·····	(296)
10.4.2 计算机软件的商业秘密·····	(297)
10.5 软件工程师的职业道德规范·····	(298)
10.5.1 职业道德·····	(298)
10.5.2 软件工程师的职业道德规范·····	(298)
10.6 小结·····	(300)
习题·····	(300)
 附录 A·····	 (302)
 附录 B·····	 (325)
 参考文献·····	 (331)

第1章 绪 论

计算机软件的发展经历了若干个不同的时期，反映了人们对软件认识不断加深的历程。在计算机出现之初，计算机软件只是专家个人的“艺术品”，注重编程的经验和技巧。随着计算机在商业领域应用的普及，计算机软件的开发人员开始注重程序设计风格，典型的代表是结构化程序设计方法与模块化设计思想。随后，由于软件的规模愈来愈大，软件的复杂程度愈来愈高，给软件的开发和维护带来了一系列严重问题，如生产率低、成本高、工作量难于估计、质量不尽人意、维护困难等，使得软件开发人员必须从更高的角度来看待软件，这便提出了软件工程，强调软件开发的科学管理，讲究软件开发的规范化。

本章主要简述软件工程的产生、研究对象与基本原理、开发方法、工具和环境、软件生存周期模型，以及软件工程的并行工程。

1.1 软件工程的产生

计算机从诞生到现在已经有50多年的历史，软件和硬件一直是计算机系统中密不可分的两个部分，并且软件成份越来越占主导地位。硬件的发展从电子管到晶体管，再到集成电路，直至超大规模、超高速集成电路，性能越来越好，价格越来越低。那么，软件的发展又怎样呢？

1.1.1 软件的特征

软件包括3个方面：

- ✧ 一个或多个计算机程序，这些程序执行时能够提供期望的功能和性能；
- ✧ 一个或多个数据结构，这些数据结构使得程序能够完全操纵信息；
- ✧ 一个或多个文档，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。

软件不同于硬件，它只具备逻辑形式而不具备物理形式。因此，软件具有与硬件不同的特征。

1. 软件是“开发”出来的，不是“制造”出来的

软件和硬件的质量都与“开发”有关，但软件的“制造”只是简单的拷贝，因而软件的质量主要取决于软件“开发”，而在软件“制造”过程中几乎不会引入新的质量问题，即使存在，也可以很容易改正。而硬件在“制造”过程中有可能会引入新的质量问题。尽管在“开发”和“制造”两种活动中，人都很重要，但两种活动中的人员组成及各类人员所要做的工作却不一样。另外，在两种活动中，虽然目标都是要构造一个“产品”，但是两者采用的方法不同。

在20世纪80年代中期的时候,有人提出了“软件工厂”的概念。但这一概念也不意味着软件“开发”和硬件“制造”能够等同起来。实际上,“软件工厂”这一概念着重于使用自动化工具和环境来进行软件开发。

2. 软件可能会被“废弃”,但不会被“用坏”

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”。

在硬件被使用的早期,可能会因为设计或制造时的错误或缺陷造成一些问题。当错误或缺陷被改正之后,硬件即可正常运行一段比较长的时期(不同的硬件其时期长短不同),这一时期硬件出错的可能性非常小,可称为硬件使用的中期。随着时间的推移,硬件出错的可能性会越来越高,这一时期可以称为晚期,晚期代表着硬件变得越来越不能被使用了,也就是“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”。

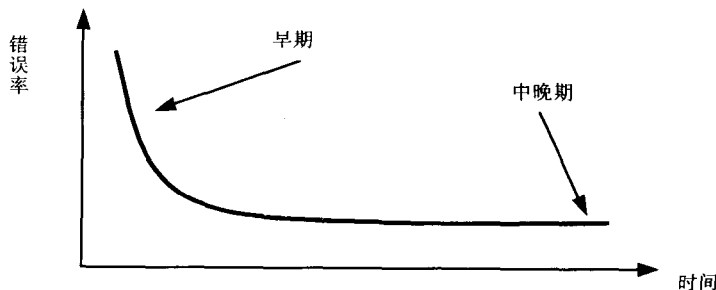


图 1-1 软件随时间变化的错误率曲线

3. 以前软件大多是“定制”的,而不是“装配”的

现代工业社会中,硬件基本上都是通过集成不同生产厂商的各种零部件,进而“装配”起来的。典型的例子是工厂的流水线作业。硬件设计师的一个主要任务就是要从当前市场上可供选择的零部件中进行选择,加上自己的设计,从而可以设计进而制造出新产品。

软件设计师则没有这么幸运。过去,市场上很少会有类似于硬件“零部件”的软件“零部件”。即便有,大多也是具有完整功能的“软件”或独立功能的模块,而不是“零件”或“部件”。因此,软件大多是为用户专门“定制”的,而不是由现有的软件零部件“装配”而成的。现在,这种情况正在快速发生变化。面向对象技术和组件开发技术的广泛使用,使得越来越多的软件部件(也称为软件组件、软件构件)被开发出来,并且可以类似于硬件部件一样做到“即插即用”。当然,软件部件要做到和硬件部件同等程度的“即插即用”,还需要软件工程师进一步的研究和实践。