



◎ **新世纪** 网络课程教材

010101001011001010
010100111010100101000101001010

0101010
010101010101
1101
01101
010101
01010101110
01010
1101

软件工程

——理论与实践

许家珩 曾翎 彭德中 编著



高等教育出版社

<http://www.hep.com.cn>
<http://www.hep.edu.cn>

新世纪网络课程教材

软件工程

——理论与实践

许家珩 曾 翎 彭德中 编著

高等教育出版社

内容提要

“软件工程”是一门指导计算机软件开发和维护的工程学科,近年来随着我国信息化建设的开展,软件工程取得了前所未有的飞速发展。

本书作为教育部新世纪网络课程建设工程的子课题“软件工程网络课程”配套教材,是在吸取了国内外有关教材的精华,并结合编者多年来进行软件工程的教學及软件开发实践经验,体会的基础上编写的。

内容注重科学性、先进性,强调实践性,提供了丰富的软件开发实例和素材,反映了软件工程的最新发展技术。全书共分 11 章,前 10 章系统地介绍了传统的软件工程方法,面向对象的软件工程方法,基于构件的软件工程方法以及软件测试,软件工程管理等的概念、方法和技术;第 11 章提供了一个综合性的设计型实验“软件工程课程设计”,给出了多个采用面向对象的方法开发的软件实例,还介绍了面向对象的软件开发工具 Rational Rose。

本书可作为高等院校“软件工程”课程的教材或教学参考书,也可供广大工程技术人员和科研人员参考使用。

图书在版编目(CIP)数据

软件工程:理论与实践 / 许家珩, 曾翎, 彭德中编著.

—北京: 高等教育出版社, 2004.7

ISBN 7-04-014147-7

I. 软... II. ①许...②曾...③彭... 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 (2004) 第 052415 号

策划编辑 董建波 责任编辑 刘 英 封面设计 刘晓翔 责任印制 陈伟光

出版发行 高等教育出版社

社 址 北京市西城区德外大街 4 号

邮政编码 100011

总 机 010-82028899

购书热线 010-64054588

免费咨询 800-810-0598

网 址 <http://www.hep.edu.cn>

<http://www.hep.com.cn>

经 销 新华书店北京发行所

印 刷 涿州市星河印刷有限公司

开 本 787×1092 1/16

印 张 19.75

字 数 400 000

版 次 2004 年 7 月第 1 版

印 次 2004 年 7 月第 1 次印刷

定 价 25.90 元 (含光盘)

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

版权所有 侵权必究

前 言

“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,是一门新兴的边缘学科。

自 1968 年在著名的 NATO 会议上为解决“软件危机”而提出“软件工程”这个概念以来,在不到 40 年的时间里,软件工程在理论和实践两个方面都取得了长足的进步,取得了大量研究成果。软件工程的应用水平已成为促进软件产业健康发展的关键。

软件是信息化的核心,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,而且软件开发更具有特殊性和复杂性。

我国在加入 WTO 以来,各行各业加速了信息化建设的步伐,也促进了软件工程的飞速发展。为了不断提高软件开发的质量和水平,必须学习、掌握和应用软件工程的基本理论和技能,了解软件经济学,提高软件产业的管理水平,才能使我国的软件产业在国际竞争中占领一席之地。

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。特别注重实践性也是本书的一大特色。全书共分 11 章,系统介绍了软件工程的基本内容、开发及管理技术,同时重点介绍了软件工程的最新发展技术,如面向对象的方法(OO)、统一建模语言(UML)、软件成熟度模型(CMM)等。在第 11 章“软件工程课程设计”中,提供了一个综合性的设计型实验——“软件工程课程设计”,并提供了 3 个软件开发的典型案例。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课题,以软件工程课程设计的实践来促进软件工程理论的学习。本教材还介绍了面向对象的软件开发工具 Rational Rose。作者开发的“软件工程网络课程”(已由高等教育出版社出版)提供了软件开发过程的指导和丰富的教学资源。

所附光盘包括两部分内容:第一部分是电子教案,第二部分是教材中所有习题的习题解答。

软件工程是一门处于前沿地位的重要学科,在迅速不断地发展。我们希望读者通过本书的学习,能将理论知识应用于软件开发的实践中,并在实践中不断创新。

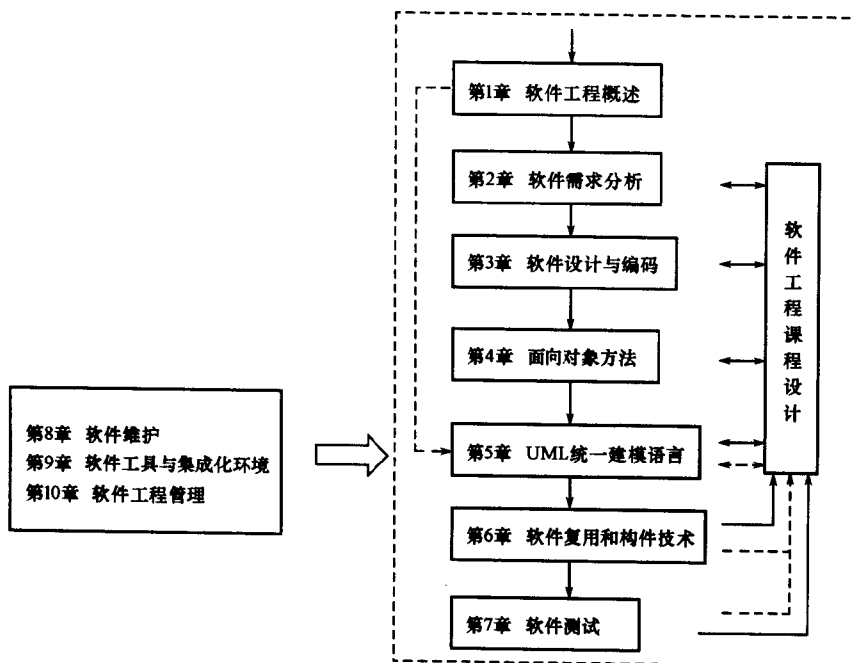
本书由黄迪明教授主审,许家珩教授主编。第 1~5 章、第 11 章由许家珩教授编写,第 8

章、第 10 章由曾翎博士编写；第 6 章、第 9 章由彭德中博士编写，第 7 章由侯晶硕士编写。同时还要感谢白忠建老师、刘智、杨臣、陈昌波及吴知硕士为本书提供了大量宝贵的素材，程永新、陈科硕士对本书进行了仔细的校对。

由于作者水平有限，书中疏漏、欠妥、谬误之处在所难免，恳请读者批评指正。

作者于电子科技大学
2004 年 5 月

软件工程课程学习指导



建议学习路径:

1. 学习完第 1、2 章后,即可开始第 11 章“软件工程课程设计”,课程设计与基本教学内容同步进行。实线箭头表示学习顺序,双向箭头表示执行过程的交迭,即在进行课程设计时,可随时返回基本教学内容的学习。

2. 如果对传统的软件工程方法已有一定了解,并拟用面向对象的方法进行软件开发,建议按照虚线箭头的方向执行;即在学习第 1 章后,学习第 5、6 章,并开始课程设计。

3. 第 8、9、10 章的内容可自行安排选学。

目 录

第 1 章 软件工程概述	(1)	2.2.2 数据流图	(24)
1.1 软件工程的产生和发展	(1)	2.2.3 实例:医院病房监护系统	(28)
1.1.1 软件工程的发展过程	(1)	2.2.4 分层 DFD 图的改进	(30)
1.1.2 软件危机	(2)	2.2.5 数据词典	(32)
1.1.3 软件工程的定义	(3)	2.2.6 加工逻辑说明	(33)
1.1.4 软件工程研究的内容	(3)	2.3 原型化方法	(35)
1.2 软件与软件生存期	(4)	2.3.1 软件原型的分类	(36)
1.2.1 软件的概念和特点	(5)	2.3.2 快速原型开发模型	(37)
1.2.2 软件工程过程	(6)	2.4 系统动态分析	(38)
1.2.3 软件生存期	(6)	2.4.1 状态迁移图	(38)
1.3 软件生存期模型	(7)	2.4.2 Petri 网	(39)
1.3.1 瀑布模型	(8)	习题二	(40)
1.3.2 循环模型	(8)	第 3 章 软件设计与编码	(43)
1.3.3 增量模型	(9)	3.1 软件设计阶段的任务与目标	(43)
1.3.4 螺旋模型	(9)	3.1.1 软件设计在开发阶段中的 重要性	(43)
1.3.5 喷泉模型	(9)	3.1.2 软件设计阶段的任务	(44)
1.3.6 智能模型	(11)	3.2 软件结构与软件结构图	(45)
1.4 软件开发方法	(11)	3.2.1 软件结构的基本概念	(45)
1.4.1 结构化开发方法	(12)	3.2.2 软件的树状结构和网状结构	(47)
1.4.2 面向数据结构的开发方法	(13)	3.2.3 软件结构图	(48)
1.4.3 原型化开发方法	(15)	3.3 模块的独立性	(49)
1.4.4 面向对象的开发方法	(16)	3.3.1 模块独立性	(49)
1.5 软件工具与软件开发环境	(17)	3.3.2 耦合性	(50)
习题一	(18)	3.3.3 内聚性	(51)
第 2 章 软件需求分析	(20)	3.3.4 信息隐蔽	(53)
2.1 基本概念	(20)	3.4 结构化设计方法	(54)
2.1.1 软件需求分析的任务	(20)	3.4.1 结构化设计方法概述	(54)
2.1.2 需求分析的过程	(21)	3.4.2 数据流图的分类与典型的 系统结构	(54)
2.1.3 软件需求分析的原则	(22)	3.4.3 变换分析	(55)
2.1.4 需求分析方法	(23)	3.4.4 事务分析	(57)
2.2 结构化分析方法	(23)		
2.2.1 SA 法概述	(24)		

3.4.5 实例:银行贷款文件管理	(58)	4.5 典型的面向对象方法	(97)
3.4.6 模块结构图的改进	(61)	4.5.1 Booch 方法	(98)
3.5 JACKSON 系统开发方法	(65)	4.5.2 Coad/Yourdon 方法	(100)
3.5.1 JACKSON 方法简介	(65)	4.5.3 对象模型技术	(103)
3.5.2 JACKSON 方法的设计过程	(67)	4.5.4 OOSE 方法	(109)
3.5.3 JACKSON 方法的技术构成	(68)	习题四	(110)
3.6 详细设计描述工具	(71)	第 5 章 UML 统一建模语言	(113)
3.6.1 程序流程图	(71)	5.1 UML 概述	(113)
3.6.2 N-S 图	(71)	5.1.1 UML 的形成	(113)
3.6.3 PAD 图	(72)	5.1.2 UML 的主要内容	(114)
3.7 用户界面设计	(74)	5.1.3 UML 的图形表示	(115)
3.7.1 用户界面的特性及设计任务 ..	(75)	5.1.4 UML 的特点	(116)
3.7.2 用户界面的基本类型	(75)	5.2 通用模型元素	(117)
3.7.3 输入/输出用户界面设计	(77)	5.2.1 模型元素	(117)
3.8 程序编码	(78)	5.2.2 约束	(118)
3.8.1 程序设计语言的选择	(79)	5.2.3 依赖关系	(119)
3.8.2 结构化程序设计	(80)	5.2.4 细化	(120)
3.8.3 程序设计风格	(81)	5.2.5 注释	(120)
3.8.4 算法与程序效率	(83)	5.3 用例模型	(120)
习题三	(83)	5.3.1 用例图	(121)
第 4 章 面向对象方法	(87)	5.3.2 确定执行者	(121)
4.1 面向对象方法概述	(87)	5.3.3 确定用例	(123)
4.1.1 什么是面向对象方法	(87)	5.3.4 用例之间的关系	(124)
4.1.2 面向对象方法的主要特点	(88)	5.3.5 用例图实例	(125)
4.2 面向对象的基本概念	(89)	5.4 建立静态模型	(127)
4.2.1 对象与类	(89)	5.4.1 类图与对象图	(128)
4.2.2 继承	(90)	5.4.2 类的识别	(129)
4.2.3 多态性	(91)	5.4.3 属性与操作识别	(132)
4.2.4 消息	(91)	5.4.4 类之间的关系	(132)
4.2.5 方法	(91)	5.4.5 包图	(138)
4.3 面向对象的分析	(91)	5.5 建立动态模型	(139)
4.3.1 需求分析中的问题	(92)	5.5.1 消息	(140)
4.3.2 面向对象分析的特点	(93)	5.5.2 状态图	(140)
4.3.3 面向对象分析的基本任务 与分析过程	(93)	5.5.3 顺序图	(142)
4.4 面向对象的设计	(95)	5.5.4 合作图	(145)
4.4.1 面向对象设计的准则	(95)	5.5.5 活动图	(147)
4.4.2 面向对象设计的基本任务	(96)	5.6 实现模型	(150)
		5.6.1 构件图	(150)

5.6.2 配置图	(151)	7.3 黑盒法测试	(184)
习题五	(153)	7.3.1 等价分类法	(184)
第6章 软件复用和构件技术	(155)	7.3.2 边界值分析法	(186)
6.1 软件复用概述	(155)	7.3.3 错误推测法	(186)
6.1.1 软件复用的基本概念	(155)	7.3.4 因果图法	(187)
6.1.2 软件复用的级别	(155)	7.4 软件测试的策略	(188)
6.1.3 软件复用的形式	(157)	7.4.1 单元测试	(188)
6.2 可复用构件与构件工程	(157)	7.4.2 组装测试	(191)
6.2.1 可复用构件	(157)	7.4.3 确认测试	(194)
6.2.2 基于构件的软件工程	(159)	7.4.4 系统测试	(195)
6.3 领域工程分析和基于构件的 开发	(159)	7.4.5 α 测试和 β 测试	(196)
6.3.1 领域分析	(159)	7.4.6 综合测试策略	(196)
6.3.2 构件的开发与构件库	(160)	7.5 排错技术	(197)
6.3.3 基于构件的开发	(162)	7.5.1 排错的原则	(197)
6.4 基于构件的软件开发特点	(163)	7.5.2 几种主要的排错方法	(197)
6.4.1 开发的质量	(163)	7.6 面向对象的测试	(199)
6.4.2 开发的效率	(163)	7.6.1 在面向对象语境中的单元 测试	(199)
6.4.3 开发的成本	(164)	7.6.2 在面向对象语境中的集成 测试	(200)
6.5 软件构件技术的技术规范	(164)	7.6.3 在面向对象语境中的确认 测试	(201)
6.5.1 对象管理组织的CORBA	(165)	习题七	(201)
6.5.2 微软的COM	(167)	第8章 软件维护	(203)
6.5.3 Sun的EJB	(169)	8.1 软件维护的基本概念	(203)
习题六	(171)	8.1.1 软件维护的目的	(203)
第7章 软件测试	(172)	8.1.2 软件维护的类型	(203)
7.1 软件测试的基本概念	(172)	8.1.3 软件维护的特性	(205)
7.1.1 软件测试的目的和重要性	(172)	8.1.4 软件维护的代价	(206)
7.1.2 软件测试的特点和原则	(173)	8.2 软件维护的过程	(207)
7.1.3 软件测试的基本步骤	(176)	8.2.1 维护组织	(207)
7.1.4 静态分析与动态测试	(177)	8.2.2 维护工作的流程	(208)
7.2 白盒法测试	(178)	8.2.3 维护工作的组织管理	(208)
7.2.1 逻辑覆盖	(178)	8.3 软件维护技术	(209)
7.2.2 语句覆盖	(179)	8.3.1 面向维护的技术	(209)
7.2.3 判定覆盖	(180)	8.3.2 维护支援技术	(210)
7.2.4 条件覆盖	(181)	8.3.3 维护档案记录	(210)
7.2.5 判定-条件覆盖	(182)	8.3.4 维护工作评价	(210)
7.2.6 条件组合覆盖	(182)		
7.2.7 白盒法测试实例	(183)		

8.4 软件可维护性	(211)	10.3.4 专家估算模型	(246)
8.4.1 软件可维护性的定义	(211)	10.3.5 IBM 估算模型	(247)
8.4.2 提高可维护性的方法	(213)	10.3.6 Putnam 估算模型	(247)
8.5 逆向工程和再工程	(217)	10.3.7 COCOMO 模型	(248)
8.5.1 逆向工程	(217)	10.3.8 成本估算方法	(250)
8.5.2 软件重构	(218)	10.3.9 成本/效益分析	(251)
8.5.3 再工程的成本/效益分析	(218)	10.4 软件项目的组织与计划	(251)
8.5.4 再工程的风险分析	(219)	10.4.1 软件项目管理的特点	(252)
习题八	(219)	10.4.2 软件开发进度计划	(252)
第9章 软件工具与集成化环境	(221)	10.4.3 人员配备与组织	(254)
9.1 软件开发工具	(221)	10.4.4 软件开发小组与软件 生产率	(255)
9.2 集成化 CASE 环境	(223)	10.5 软件质量保证	(256)
9.2.1 概述	(223)	10.5.1 软件质量因素的定义	(257)
9.2.2 集成化的 CASE 开发环境的 体系结构	(224)	10.5.2 软件质量保证工作	(257)
9.3 软件开发工具 Rational Rose	(226)	10.5.3 软件项目的跟踪与控制	(258)
9.3.1 Rose 工具简介	(226)	10.6 软件能力成熟度模型(CMM)	(258)
9.3.2 业务用例图	(227)	10.6.1 CMM 的基本概念	(258)
9.3.3 用例图	(228)	10.6.2 软件过程的成熟度等级	(259)
9.3.4 类图	(230)	10.6.3 关键过程区域	(261)
9.3.5 协作图与时序图	(232)	10.6.4 软件企业如何实施 CMM	(261)
9.3.6 活动图	(233)	习题十	(265)
9.3.7 状态图	(234)	第11章 软件工程课程设计	(267)
9.3.8 构件图和部署图	(235)	11.1 课程设计目的与要求	(267)
习题九	(237)	11.1.1 目的与要求	(267)
第10章 软件工程管理	(238)	11.1.2 命题原则	(267)
10.1 软件工程管理概述	(238)	11.2 课程设计步骤安排	(268)
10.1.1 软件管理的任务与目标	(238)	11.2.1 确定课题	(268)
10.1.2 软件的作用范围	(239)	11.2.2 需求分析	(268)
10.1.3 资源要求	(240)	11.2.3 软件设计	(268)
10.2 可行性研究	(241)	11.2.4 编码与测试	(269)
10.2.1 可行性研究的任务	(241)	11.2.5 验收测试	(269)
10.2.2 系统的描述	(241)	11.3 案例分析	(270)
10.2.3 可行性研究报告	(242)	11.3.1 案例一 ATM 系统	(270)
10.3 成本估算技术	(243)	11.3.2 案例二 医院病房监护 系统	(281)
10.3.1 影响成本估算的因素	(244)	11.3.3 案例三 会议系统	(288)
10.3.2 成本估算模型	(245)	参考文献及参考网站	(303)
10.3.3 Halstead 理论模型	(246)		

第 1 章 软件工程概述

1.1 软件工程的产生和发展

软件工程是在克服 20 世纪 60 年代末出现“软件危机”(Software Crisis)的过程中逐渐形成和发展的。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 Software Engineering(软件工程)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。

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,是一门新兴的边缘学科,它涉及到计算机科学、工程科学、管理科学、数学等多个学科。软件工程的研究范围广,不仅包括软件系统的开发方法和技术、管理技术,还包括软件工具、环境及软件开发的规范。

软件是信息化的核心,国民经济、国防建设、社会发展及人民生活都离不开软件。软件产业关系到国家经济发展和文化安全,体现了国家综合实力,是决定 21 世纪国际竞争地位的战略产业。因此大力推广应用软件工程的开发技术及管理技术,提高软件工程的应用水平,对促进我国软件产业与国际接轨,推动软件产业的迅速发展起着十分重要的作用。

1.1.1 软件工程的发展过程

软件工程的产生和发展是与软件的发展过程紧密相关的。自从第一台电子计算机诞生以来,就开始了软件的生产,软件工程提出至今,它的发展已经历了四个重要阶段。

1. 第一代软件工程(20 世纪 60 年代末到 20 世纪 70 年代)

20 世纪 60 年代末,软件生产主要采用“生产作坊”方式。随着软件需求量、规模及复杂度的迅速增大,生产作坊的方式已不能够适应软件生产的需要,出现了所谓“软件危机”,即软件生产效率低,大量质量低劣的软件涌入市场或在开发过程中夭折。由于软件危机的不断扩大,对软件生产已经产生了严重危害。

为了克服软件危机,在著名的 NATO 软件可靠性会议上第一次提出了“软件工程”这一名词,将软件开发纳入了工程化的轨道,基本形成了软件工程的观念、框架、技术和方法。这一阶段又称为传统的软件工程。

2. 第二代软件工程(20 世纪 80 年代中到 20 世纪 90 年代)

20 世纪 80 年代中开始,以 Smalltalk 为代表的面向对象的程序设计语言相继推出,面向

对象的方法与技术得到发展。从 20 世纪 90 年代起,研究的重点从程序设计语言逐渐转移到面向对象的分析与设计,演化成为一种完整的软件开发方法和系统的技术体系。20 世纪 90 年代以来,出现了许多面向对象的开发方法的流派,面向对象的方法逐渐成为软件开发的主流。所以这一阶段又称为对象工程。

3. 第三代软件工程

随着规模的不断增大,开发人员也随之增多,开发时间相应持续增长,加之软件是知识密集型的逻辑思维产品,这些都增加了软件工程管理的难度。人们在软件开发的实践过程中认识到:提高软件生产率、保证软件质量的关键是对“软件过程”的控制和管理,提出对软件项目管理的计划、组织、成本估算、质量保证、软件配置管理等技术与策略,逐步形成了软件过程工程。

4. 构件工程

20 世纪 90 年代起,基于构件(Component)的开发方法取得了重要进展,软件系统的开发可通过使用现存的可复用构件组装完成,而无需从头开始构造,从而达到提高效率和质量、降低成本的目的,称之为构件工程。

1.1.2 软件危机

1. 软件危机的产生

“软件危机”的出现是由于软件的规模越来越大,复杂度不断增加,而软件需求量也不断增大,“生产作坊”式的软件开发模式及技术已不能满足软件发展的需要。

软件开发过程是一种高密集度的脑力劳动,需要投入大量的人力、物力和财力。由于软件开发的模式及技术不能适应软件发展的需要,致使大量质量低劣软件产品涌向市场,有的甚至在开发过程中就夭折了。国外在开发一些大型软件系统时,遇到了许多困难,有的系统最终彻底失败了;有的系统则比原计划推迟了好多年,而且费用大大超过了预算;有的系统不能符合用户当初的期望;有的系统则无法进行修改维护。

例如 IBM 公司的 OS/360,约有 100 万条指令,花费了 5000 个人一年的劳动,经费达数亿美元,而结果却令人沮丧,错误多达 2000 个以上,系统根本无法正常运行。OS/360 系统的负责人 Brooks 这样描述开发过程的困难和混乱:“像巨兽在泥潭中做垂死挣扎,挣扎得越猛,泥浆就沾得越多,最后没有一个野兽能够逃脱淹没在泥潭中的命运……”。

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而无法取得成功。

还有,可以称为 20 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。

2. 软件危机的表现

20 世纪 60 年代末期所发生的软件危机,反映在软件可靠性没有保障、软件维护工作量大、费用不断上升、进度无法预测、成本增长无法控制、程序人员无限度地增加等各个方面,以致形成人们难以控制软件开发的局面。

软件危机主要表现在两个方面:

- (1) 软件产品质量低劣,甚至在开发过程中就夭折。
- (2) 软件生产率,不能满足需要。

1.1.3 软件工程的定义

自从 1968 年提出“软件工程”这个术语以来,对于软件工程就有了各种各样的定义,但是它们的基本思想都是强调在软件开发过程中应用工程化原则的重要性。

例如,Boehm 曾经为软件工程下了以下定义:“运用现代科学技术知识实践并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。”

1983 年,IEEE(国际电气与电子工程师协会)所下的定义是:软件工程是开发、运行、维护和修复软件的系统方法。

1990 年,IEEE 又将其定义更改为:对软件开发、运作、维护的系统化的、有规范的、可量化的方法应用,即是对软件的工程化应用。

软件工程有方法、工具和过程三个要素。软件工程方法就是研究软件开发“如何做”的技术;软件工具是研究支撑软件开发方法的工具、软件工具的集成环境如计算机辅助软件工程 CASE;软件工程过程则是指将软件工程方法与软件工具相结合,实现合理、及时地进行软件开发的目的。

1.1.4 软件工程研究的内容

软件工程是一门指导软件开发的工程学科,是以计算机理论及其他相关学科的理论为指导,采用工程化的概念、原理、技术和方法进行软件的开发和维护,把经实践证明的科学的措施与最先进的技术方法结合起来,以较少的代价获得高质量的软件。

1. 软件工程研究的内容

软件工程涉及的学科多,研究的范围广。归结起来软件工程研究的主要内容有以下 4 个方面:方法与工具、工具及环境、管理技术、标准与规范。

软件开发方法,主要讨论软件开发的各种方法及其工作模型。包括多方面的任务,如软件系统需求分析、总体设计,以及如何构建良好的软件结构、数据结构及算法设计等,同时讨论具体实现的技术,软件工具为软件工程方法提供了支持,研究计算机辅助软件工程 CASE,建立软件工程环境。

软件工程管理,是指对软件工程全过程的控制和管理,包括计划安排、成本估算、项目管

理、软件质量管理。软件工程标准化与规范化,使得各项工作有章可循,以保证软件生产率和软件质量的提高。软件工程标准可分为几个层次:国际标准、行业标准、企业规范和项目规范。

需要强调的是,随着人们对软件系统研究的逐渐深入,软件工程所研究的内容也在不断地发生变化。

2. 软件工程的目標

软件工程研究的目標是以較少的投資獲取高質量的軟件。即軟件的開發要在保證質量和效率的同時,盡量縮短開發期,降低軟件成本。軟件過程所實現的多目標中,有的是互補的,例如縮短開發期,顯然可降低成本,維護是需要代價的,易于維護就可降低總成本。高性能與高可靠性是互補的。而有的目標則是互斥的,例如,要獲得高的可靠性,通常要採取一些冗餘的措施,往往會增加成本。

為了實現軟件工程的多目標,要對軟件的各項質量指標進行綜合考慮,以實現軟件開發的“多、快、好、省”的總目標。軟件工程目標如圖 1.1 所示。

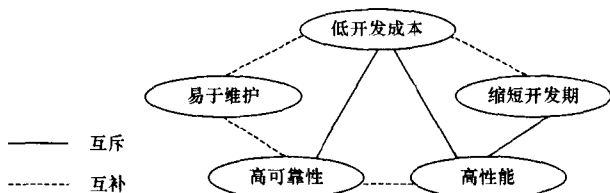


圖 1.1 軟件工程的目標

3. 軟件工程的基本原則

過去,軟件工程的基本原則是抽象、模塊化、清晰的結構、精確的設計規格說明。但今天的認識已經發生了很大的變化。現在提出的軟件工程 4 條基本原則是:

(1) 必須認識軟件需求的變動性,採取適當措施來保證結果產品能忠實地滿足用戶要求。在軟件設計中,通常要考慮模塊化、抽象與信息隱蔽、局部化、一致性等原則。

(2) 穩妥的設計方法大大地方便了軟件開發,以達到軟件工程的目標。軟件工具與環境對軟件設計的支持來說,頗為重要。

(3) 軟件工程項目的質量與經濟開銷直接取決於對它所提供的支撐的質量與效用。

(4) 有效的軟件工程只有在對軟件過程進行有效管理的情況下才能實現。

近年來,印度的軟件產業迅速發展,其成功的經驗是,嚴格按照國際規範進行科學管理。在本教材中,雖然主要討論軟件開發技術,只在第 10 章討論軟件管理技術,但軟件管理仍然是軟件開發成功的關鍵,因此,將介紹當前質量管理的國際規範軟件成熟度模型 CMM。

1.2 軟件與軟件生存期

“軟件工程”是指在軟件生產中採用工程化的方法,採用一系列科學的、現代化的方法技術來開發軟件。這種工程化的思想貫穿到軟件開發和維護的全過程。

为了进一步学习有关软件工程的方法、技术,首先介绍软件、软件生存期及软件工程过程这几个重要的概念。

1.2.1 软件的概念和特点

1. 软件

“软件就是程序,开发软件就是编写程序”是一个错误的观念。这种错误观点的长期存在,影响了软件工程的正常发展。

事实上,正如 Boehm 指出的,软件是程序以及开发、使用和维护程序所需的所有文档。它是由应用程序、系统程序、面向用户的文档及面向开发者的文档 4 部分构成。

即,软件 = 程序 + 文档。

软件具有如下特点:

- (1) 软件是一种逻辑实体,不是具体的物理实体。
- (2) 软件产品的生产主要是研制。
- (3) 软件具有“复杂性”,其开发和运行常受到计算机系统的限制。
- (4) 软件成本昂贵,其开发方式目前尚未完全摆脱手工生产方式。
- (5) 软件不存在磨损和老化问题,但存在退化问题。

图 1.2 给出了硬件的失效率曲线,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失效率急剧上升。

图 1.3 所描述的软件失效率曲线,它没有 U 型曲线的右半翼,表明软件随着使用时间的增加失效率降低,因为软件不存在磨损和老化问题,然而存在退化问题。

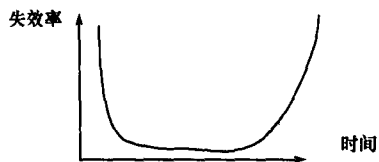


图 1.2 硬件失效率曲线

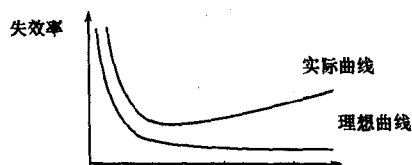


图 1.3 软件失效率曲线

2. 软件的分类

(1) 按照软件功能划分

- ① 系统软件:如操作系统、设备驱动程序等。
- ② 支撑软件(实用软件):协助用户开发的工具软件,如编辑程序、程序库、图形软件包等。
- ③ 应用软件:如工程与科学计算软件、CAD/CAM 软件、CAI 软件、信息管理系统等。

(2) 按照软件规模划分

如表 1.1 所示,按照软件的不同大小(源代码行)、参加人数、研制时间分为微型、小型、

中型、大型、甚大型和极大型。需要说明的是,随着软件产品规模的不断增大,类别的指标也会发生变化。

表 1.1 软件规模分类

类别	参加人数	研制期限	产品规模(源代码行)
微型	1	1~4 周	500 行
小型	1	1~6 月	1000~2000 行
中型	2~5	1~2 年	5000~50 000 行
大型	5~20	2~3 年	5000~500 000 行
甚大型	100~1000	4~5 年	1 000 000 行
极大型	2000~5000	5~10 年	1 000 000~10 000 000 行

(3) 按照软件工作方式划分

可以分为实时处理软件、交互式软件、批处理软件。

(4) 按照软件服务对象的范围划分

① 项目软件:由客户委托开发的软件。

② 产品软件:由软件开发机构开发,提供给市场的。

此外,还可以按照软件使用的频率及按照软件失效的影响进行划分。

1.2.2 软件工程过程

软件工程过程(Software Engineering Process)是指在软件工具的支持下,所进行的一系列软件工程活动。通常包括以下 4 类基本过程:

(1) 软件规格说明:规定软件的功能及其运行环境。

(2) 软件开发:产生满足规格说明的软件。

(3) 软件确认:确认软件能够完成客户提出的要求。

(4) 软件演进:为满足客户的变更要求,软件必须在使用的过程中演进。

软件工程过程具有可理解性、可见性(过程的进展和结果可见)、可靠性、可支持性(易于使用 CASE 工具支持)、可维护性、可接受性(为软件工程师接受)、开发效率和健壮性(抵御外部意外错误的能力)等特性。

1.2.3 软件生存期

软件生存期(Life Cycle)是指一个从用户需求开始,经过开发、交付使用,在使用中不断地增补修订,直至软件报废的全过程,也称软件生命周期(SDLD)。

GB 8567 中规定,软件生命周期分为 7 个阶段,下面分别进行介绍。

1. 可行性研究和项目开发计划

可行性研究和项目开发计划阶段必须要回答的问题是“要解决的问题是什么”。

2. 需求分析

需求分析阶段的任务不是具体地解决问题,而是准确地确定“软件系统必须做什么”,确定软件系统必须具备哪些功能。

3. 概要设计

概要设计就是设计软件的结构,包括组成模块,模块的层次结构,模块的调用关系是,每个模块的功能等等。同时还要设计该项目的应用系统的总体数据结构和数据库结构,即应用系统要存储什么数据,这些数据是什么样的结构,它们之间有什么关系等。

4. 详细设计

详细设计阶段就是为每个模块完成的功能进行具体描述,要把功能描述变为精确的、结构化的过程描述。

5. 编码

编码阶段就是把每个模块的控制结构转换成计算机可接受的程序代码,即写成以某特定程序设计语言表示的“源程序清单”。

6. 测试

测试是保证软件质量的重要手段,其主要方式是在设计测试用例的基础上检验软件的各个组成部分。测试分为模块测试、组装测试和确认测试。

7. 维护

软件维护是软件生存期中时间最长的阶段。已交付的软件投入正式使用后,便进入软件维护阶段,它可以持续几年甚至几十年。

在大部分文献中将生存周期划分为五个阶段,即要求定义、设计、编码、测试及维护。其中要求定义阶段包括可行性研究、项目开发计划和需求分析,设计阶段包括概要设计和详细设计。

为了描述软件生存期的活动,提出了多种生存期模型。例如,瀑布模型、循环模型、演化模型和螺旋模型等。

1.3 软件生存期模型

软件生存期模型是描述软件开发过程中各种活动如何执行的模型。它确立了软件开发和演绎中各阶段的次序限制以及各阶段活动的准则,确立开发过程所遵守的规定和限制,便于各种活动的协调以及各种人员的有效通信,有利于活动重用和活动管理。

目前有许多种软件生存期模型,如瀑布模型、增量模型、螺旋模型、喷泉模型、变换模型和基于知识的模型等。