



普通高等教育
软件工程

“十二五”规划教材



工业和信息化普通高等教育
“十二五”规划教材

12th Five-Year Plan Textbooks
of Software Engineering

软件工程

瞿中 宋琦 刘玲慧 王江涛 ○ 编著

Software
Engineering



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS



普通高等教育
软件工程

“十二五”规划教材

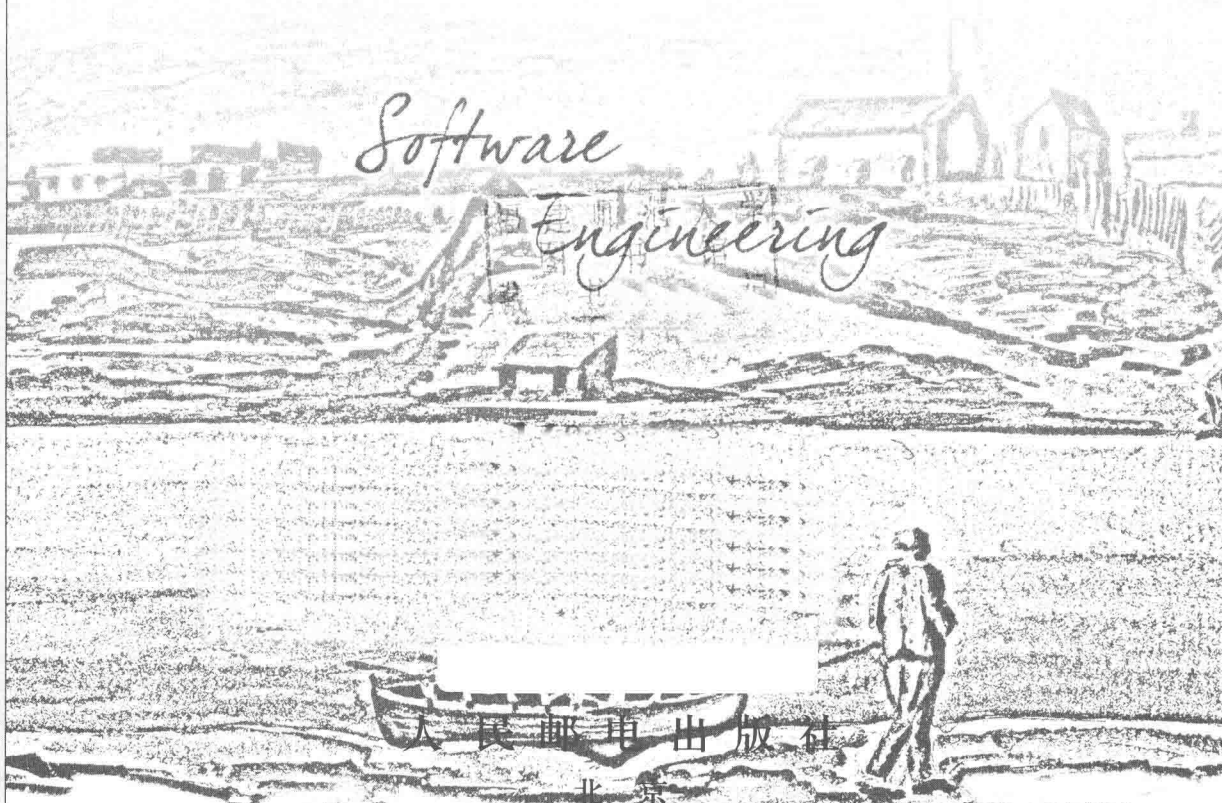


工业和信息化普通高等教育
 “十二五”规划教材

12th Five-Year Plan Textbooks
 of Software Engineering

软件工程

瞿中 宋琦 刘玲慧 王江涛 编著



人民邮电出版社

北京

图书在版编目(CIP)数据

软件工程 / 瞿中等编著. -- 北京: 人民邮电出版社, 2016.8
普通高等教育软件工程“十二五”规划教材
ISBN 978-7-115-43103-5

I. ①软… II. ①瞿… III. ①软件工程—高等学校—教材 IV. ①TP311.5

中国版本图书馆CIP数据核字(2016)第160960号

内 容 提 要

本书从实用的角度出发, 参照美国计算机协会(Association for Computing Machinery, ACM)和美国电气电子工程师学会(Institute of Electrical and Electronic Engineers, IEEE)的计算教程(Computing Curricula) 2014-201 关于软件工程的要求, 吸取了国内外软件工程的精华, 详细介绍了软件工程、软件开发过程、软件计划、需求分析、总体设计、详细设计、编码、软件测试、软件维护、软件工程标准化和软件文档、软件工程质量、软件工程项目管理、开发实例、经典例题分析等知识。每章配有习题, 以指导读者深入地进行学习。

本书内容丰富, 结构合理, 既可作为高等学校软件工程、计算机专业课程的教材或教学参考书, 也可作为通信、电子信息、自动化等相关专业读者的计算机课程教材, 还可供软件工程师、软件项目管理者 and 应用软件开发人员阅读参考。

◆ 编 著 瞿 中 宋 琦 刘玲慧 王江涛

责任编辑 刘 博

责任印制 杨林杰

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京鑫正大印刷有限公司印刷

◆ 开本: 787×1092 1/16

印张: 25.5

2016年8月第1版

字数: 738千字

2016年8月北京第1次印刷

定价: 59.80元

读者服务热线: (010)81055256 印装质量热线: (010)81055316

反盗版热线: (010)81055315

目 录

第一篇 面向过程的软件工程

第 1 章 概论	2	2.3.2 系统流程图举例	30
1.1 软件	2	2.3.3 分层	30
1.1.1 软件的定义及特点	2	2.4 软件计划的制订	31
1.1.2 软件的发展历程	3	2.4.1 确定软件计划	31
1.1.3 软件分类	3	2.4.2 复审软件计划	32
1.1.4 软件的应用领域	4	2.4.3 开发方案的选择	32
1.2 软件危机	5	2.5 成本/效益分析	33
1.2.1 软件危机的定义	5	2.5.1 成本估算	33
1.2.2 软件危机产生的原因	6	2.5.2 成本/效益分析的方法	36
1.2.3 软件危机解决的途径	6	2.6 需求分析	37
1.3 软件工程	7	2.6.1 需求分析的概念	37
1.3.1 软件工程的定义和研究对象	7	2.6.2 需求分析的层次	37
1.3.2 软件工程的基本原理	7	2.6.3 需求分析的目标和任务	38
1.3.3 软件工程的基本目标	8	2.6.4 需求分析的原则	38
1.3.4 软件工程的基本原则	8	2.6.5 需求分析的过程及方法	39
1.4 软件生存周期	9	2.6.6 应用域	40
1.5 软件开发过程模型	10	2.6.7 业务模型的建立	40
1.6 软件开发方法及工具	18	2.6.8 需求规格说明书	42
1.6.1 软件的开发方法	18	2.6.9 评审	42
1.6.2 软件的开发工具	20	2.7 传统的软件建模	44
1.7 软件工程的最新发展动向	20	2.7.1 分析建模	44
1.8 典型例题详解	21	2.7.2 数据模型	44
小结	23	2.7.3 功能模型	46
习题 1	24	2.7.4 行为模型	46
		2.7.5 数据字典	46
		2.7.6 分析实例	47
第 2 章 分析阶段	26	2.8 典型例题详解	50
2.1 问题定义	26	2.9 实验——音乐点播管理系统需求分析	52
2.2 可行性研究	26	小结	55
2.2.1 可行性研究的任务	26	习题 2	56
2.2.2 可行性研究的基本内容	27		
2.2.3 可行性研究的步骤	27	第 3 章 总体设计	58
2.3 系统流程图	29	3.1 总体设计的任务及过程	58
2.3.1 系统流程图的符号	29	3.1.1 总体设计的任务	58

3.1.2 总体设计的过程	58	4.5 面向数据结构的设计方法	104
3.2 总体设计的原理	59	4.5.1 Jackson 程序设计方法	104
3.2.1 软件结构和过程	59	4.5.2 Warnier 程序设计方法	105
3.2.2 模块设计	60	4.6 基于组件的设计方法	105
3.2.3 结构设计	65	4.7 界面设计	106
3.3 总体设计准则	67	4.7.1 用户界面设计	106
3.4 总体设计的常用方法及工具	68	4.7.2 字符界面设计	107
3.4.1 面向数据流的设计方法	68	4.7.3 菜单设计	108
3.4.2 总体设计中的工具	71	4.7.4 对话框设计	109
3.4.3 总体设计说明书编写规范	73	4.7.5 多窗口界面设计	109
3.5 模块结构设计	73	4.8 典型例题详解	110
3.6 数据存储设计	73	4.9 实验——音乐点播管理系统详细设计	113
3.7 模型—视图—控制器框架	74	小结	114
3.7.1 MVC 模式	74	习题 4	114
3.7.2 MVC 中的模型类、视图类和 控制类	74	第 5 章 编码及测试	116
3.7.3 MVC 的实现	75	5.1 程序设计语言	116
3.8 软件体系结构	75	5.1.1 程序设计语言的发展及分类	116
3.8.1 软件体系结构的兴起	75	5.1.2 程序设计语言的选择标准	118
3.8.2 软件体系结构的概念	76	5.2 程序设计风格	119
3.8.3 软件体系结构的现状	77	5.2.1 源程序文档化	120
3.8.4 软件体系结构的描述方法	79	5.2.2 数据说明	120
3.9 软件体系结构与操作系统	82	5.2.3 表达式和语句结构	121
3.9.1 分层结构	82	5.2.4 输入和输出	121
3.9.2 微内核结构	83	5.3 程序效率	122
3.10 典型例题详解	85	5.3.1 代码效率	122
3.11 实验——音乐点播管理系统总体设计	88	5.3.2 存储器效率	123
小结	91	5.3.3 输入/输出效率	123
习题 3	91	5.4 编程安全	123
第 4 章 详细设计	94	5.4.1 冗余程序设计	123
4.1 详细设计的任务	94	5.4.2 防错程序设计	124
4.2 详细设计的原则	95	5.5 结构化程序设计方法	124
4.3 详细设计的方法及工具	95	5.6 程序的复杂性及度量	126
4.3.1 详细设计的方法	95	5.6.1 代码行度量法	126
4.3.2 详细设计的工具	96	5.6.2 McCabe 度量法	126
4.3.3 详细设计工具的选择	103	5.6.3 Halstead 度量法	128
4.4 详细设计规格说明及复审	103	5.7 软件测试	129
4.4.1 详细设计说明	103	5.7.1 软件测试的意义	129
4.4.2 详细设计复审	103	5.7.2 软件测试的基本概念	129

5.7.3 软件测试的目的、任务、原则和 研究对象	130	5.13 典型例题详解	168
5.7.4 软件测试的发展历史及趋势	132	5.14 实验——音乐点播管理系统测试	172
5.7.5 软件测试的需求规格说明	133	小结	174
5.7.6 软件测试的设计说明	134	习题 5	175
5.8 软件测试的方法	135	第 6 章 软件维护及软件再工程	178
5.8.1 静态测试和动态测试	135	6.1 软件维护	178
5.8.2 黑盒测试法和白盒测试法	136	6.1.1 软件维护的定义	178
5.9 软件测试的步骤	141	6.1.2 软件维护的分类	178
5.9.1 单元测试	141	6.1.3 软件维护的成本	180
5.9.2 集成测试	144	6.1.4 软件维护的特点	181
5.9.3 确认测试	148	6.2 软件维护过程	182
5.9.4 系统测试	148	6.3 软件的可维护性	185
5.9.5 验收测试	152	6.3.1 影响软件可维护性的因素	185
5.10 调试	154	6.3.2 软件可维护性度量	187
5.10.1 调试过程	154	6.3.3 提高软件可维护性的方法	187
5.10.2 调试途径	155	6.4 逆向工程和再工程	189
5.11 测试设计和管理	156	6.4.1 预防性维护	189
5.11.1 错误曲线	156	6.4.2 软件的逆向工程和再工程	189
5.11.2 测试用例设计	157	6.4.3 软件再工程过程	190
5.12 软件测试工具	165	6.4.4 软件再工程的方法	190
5.12.1 自动软件测试的优点	165	6.5 典型例题详解	191
5.12.2 测试工具分类	165	小结	191
5.12.3 自动测试的相关问题	167	习题 6	192

第二篇 面向对象的软件工程

第 7 章 面向对象方法学 195

7.1 面向对象方法学概述	195
7.1.1 面向对象方法学的要点	196
7.1.2 面向对象方法学的优点	197
7.2 面向对象方法学的基本概念	199
7.2.1 对象	199
7.2.2 其他概念	201
7.3 面向对象建模	203
7.4 对象模型	204
7.4.1 类图的基本符号	204
7.4.2 表示关系的符号	205
7.5 动态模型	209
7.6 功能模型	210

7.6.1 用例图	210
7.6.2 用例建模	212
7.7 三种模型之间的关系	212
7.8 典型例题详解	213
7.9 实验——音乐点播管理系统面向对象 方法学	214
小结	217
习题 7	217

第 8 章 面向对象分析 219

8.1 面向对象分析建模过程	219
8.1.1 概述	219
8.1.2 3 个子模型与 5 个层次	220
8.2 需求陈述	221

8.2.1	书写要点	221
8.2.2	例子	221
8.3	建立对象模型	222
8.3.1	确定类与对象	222
8.3.2	确定关联	224
8.3.3	划分主题	226
8.3.4	确定属性	227
8.3.5	识别继承关系	228
8.3.6	反复修改	229
8.4	建立动态模型	231
8.4.1	编写脚本	231
8.4.2	设想用户界面	232
8.4.3	绘制事件跟踪图	232
8.4.4	绘制状态图	233
8.4.5	审查动态模型	234
8.5	建立功能模型	235
8.5.1	绘制基本系统模型图	236
8.5.2	绘制功能级数据流图	236
8.5.3	描述处理框功能	237
8.6	定义服务	237
8.7	典型例题详解	238
8.8	实验——音乐点播管理系统面向 对象分析	240
小结		245
习题 8		246
第 9 章	面向对象设计	248
9.1	面向对象设计的准则	248
9.2	启发规则	250
9.3	软件重用	251
9.3.1	概述	251
9.3.2	类构件	252
9.3.3	软件重用的效益	254
9.4	划分子系统	254
9.4.1	子系统之间的两种交互方式	255
9.4.2	组织系统的两种方案	255
9.4.3	设计系统的拓扑结构	256
9.5	设计子系统	256
9.5.1	设计问题域子系统	256
9.5.2	设计人机交互子系统	258

9.5.3	设计任务管理子系统	260
9.5.4	设计数据管理子系统	261
9.6	设计类中的服务	263
9.6.1	确定类中应有的服务	264
9.6.2	设计实现服务的方法	264
9.7	设计关联	265
9.7.1	关联的遍历	265
9.7.2	关联对象的实现	266
9.8	设计优化	266
9.8.1	确定优先级	266
9.8.2	提高效率的技术	266
9.8.3	调整继承关系	267
9.9	设计模式	269
9.9.1	相关概念	269
9.9.2	描述设计模式	270
9.9.3	23 种设计模式	271
9.10	典型例题详解	272
9.11	实验——音乐点播管理系统面向对象 设计	273
小结		275
习题 9		275

第 10 章 面向对象实现 277

10.1	面向对象语言	277
10.1.1	面向对象语言的优点	277
10.1.2	面向对象语言的技术特点	278
10.1.3	面向对象语言的选择原则	280
10.2	面向对象程序设计风格	281
10.2.1	提高可重用性	281
10.2.2	提高可扩充性	282
10.2.3	提高稳健性	282
10.3	测试策略	283
10.3.1	面向对象测试模型	283
10.3.2	面向对象分析的测试	283
10.3.3	面向对象设计的测试	283
10.3.4	面向对象编程的测试	283
10.3.5	面向对象的单元测试	284
10.3.6	面向对象的集成测试	284
10.3.7	面向对象的系统测试	284

10.4 设计测试用例·····285	10.6 实验——音乐点播管理系统面向对象实现·····287
10.4.1 设计类测试用例·····285	小结·····288
10.4.2 测试类间测试用例·····286	习题 10·····288
10.5 典型例题详解·····287	
第三篇 软件工程管理及开发实例	
第 11 章 软件工程标准化和软件文档·····291	
11.1 软件工程标准化·····291	12.5.2 软件质量管理的指导思想·····320
11.1.1 软件工程标准化的概念·····291	12.5.3 软件质量管理体系·····321
11.1.2 软件工程标准化的类型及意义·····291	12.6 典型例题详解·····325
11.2 软件工程标准的制定与推行·····293	小结·····325
11.3 软件工程标准的层次和体系框架·····294	习题 12·····326
11.3.1 软件工程标准的层次·····294	
11.3.2 中国的软件工程标准化工作·····295	第 13 章 软件工程项目管理·····328
11.4 ISO 9000 国际标准简介·····296	13.1 软件项目管理·····328
11.5 软件文档·····298	13.1.1 软件项目管理的特点·····328
11.5.1 软件文档的作用和分类·····298	13.1.2 软件项目管理的主要职能·····328
11.5.2 软件文档编制的质量要求·····301	13.1.3 软件项目管理的主要内容·····329
11.5.3 软件文档的管理和维护·····302	13.1.4 软件项目管理活动·····329
11.6 典型例题详解·····303	13.2 基于 CASE 技术的开发工具简介·····332
小结·····305	13.3 成本估算·····333
习题 11·····305	13.4 计划和组织·····334
	13.4.1 项目计划的制订·····334
第 12 章 软件工程质量·····308	13.4.2 项目组人员组织与管理·····335
12.1 软件质量概述·····308	13.5 进度计划与控制·····337
12.1.1 软件质量的定义·····308	13.5.1 制订开发进度计划·····337
12.1.2 软件质量的特性·····309	13.5.2 Gantt 图与时间管理·····338
12.2 软件质量的度量模型·····310	13.5.3 工程网络与关键路径·····338
12.2.1 软件度量和软件质量的度量·····310	13.5.4 项目进度跟踪与控制·····340
12.2.2 软件质量的度量模型·····313	13.6 风险管理·····342
12.3 软件质量保证·····315	13.6.1 风险识别与分类·····342
12.3.1 软件质量保证的概念·····315	13.6.2 风险评估与分析·····344
12.3.2 软件质量保证的主要任务·····316	13.6.3 风险策划与管理·····345
12.3.3 软件质量保证的策略·····316	13.6.4 风险规避与监控·····346
12.4 技术评审·····317	13.7 配置管理·····347
12.5 软件质量管理体系·····319	13.7.1 软件配置·····347
12.5.1 软件产品质量管理的特点·····319	13.7.2 软件配置管理的任务·····347
	13.7.3 软件配置管理的过程·····349
	13.8 项目管理认证体系 IPMP 与 PMP·····352
	13.8.1 IPMP 概况·····352
	13.8.2 PMP 简介·····352

13.9 典型例题详解	353	14.4.5 系统数据库设计	368
13.10 软件工程项目管理实验	354	14.5 详细设计	369
小结	359	14.6 系统实现	371
习题 13	359	14.7 测试与维护	372
第 14 章 简单的人事管理系统设计与开发		14.7.1 测试用例与测试结果	372
361		14.7.2 系统维护	373
14.1 项目论证和计划	361	小结	375
14.1.1 系统调查	361	习题 14	375
14.1.2 新系统的总体功能需求和性能要求	361	附录一 可行性研究报告	377
14.1.3 系统开发的框架	362	附录二 需求规格说明书	380
14.2 可行性分析	362	附录三 总体设计说明书	382
14.3 需求分析	363	附录四 详细设计说明书	385
14.3.1 数据流分析	363	附录五 软件测试的需求规格说明书	387
14.3.2 系统流程图	364	附录六 软件维护手册	390
14.3.3 数据字典	365	附录七 UML 的模型及图示表示	392
14.3.4 系统用例图	366	参考文献	398
14.4 总体设计	366		
14.4.1 功能模块图	366		
14.4.2 层次方框图	367		
14.4.3 IPO 图	367		
14.4.4 工作流程图	367		

第一篇

面向过程的软件工程

第 1 章 概论

本章要点

- 软件、软件工程的概念以及开发的主要原则
- 软件开发过程的模型以及开发方法
- 软件工程的生存周期
- 软件工程发展的最新动向

1.1 软件

1.1.1 软件的定义及特点

软件 (Software), 包括程序 (Program)、相关数据 (Data) 及其说明文档 (Document), 在计算机系统中与硬件 (Hardware) 相互依存。其中程序是按照事先设计的功能和性能要求执行的指令序列; 数据是程序能正常操纵信息的数据结构; 说明文档包含与程序开发、维护和使用过程中有关的各种图文数据。

软件有如下特点。首先, 软件是一种抽象的逻辑实体; 其次, 软件是一种通过智力活动, 把知识与技术转化为信息的一种产品, 它是在研制、开发中被创造出来的; 最后, 在软件的运行和使用期间, 没有硬件那样的机器磨损、老化问题, 但是软件也存在退化问题, 需要维护。图 1-1 所示为硬件与软件使用过程中产生的失效率曲线。

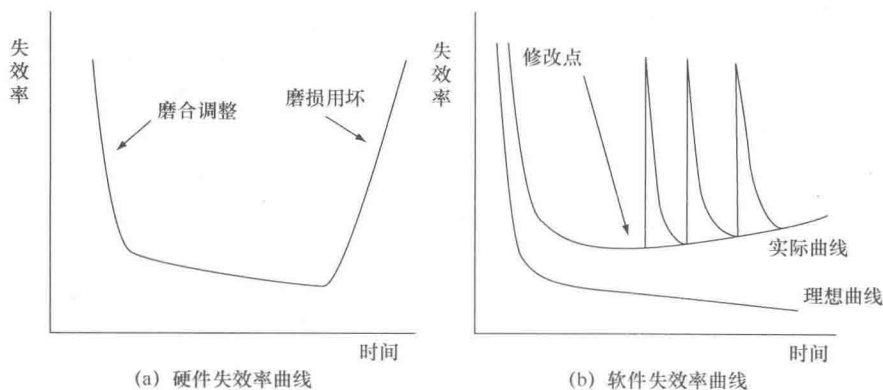


图 1-1 失效率曲线

另外,软件还具有受计算机硬件系统限制、至今尚未摆脱人工开发方式、开发过程复杂、成本相当高、涉及诸多社会因素等特点。

1.1.2 软件的发展历程

世界上首位程序员奥古斯塔·爱达·金 (Augusta Ada King, 1815—1852) 是英国诗人乔治·戈登·拜伦 (George Gordon Byron, 1788—1824) 的独生女,师承计算机数学基础布尔代数的创始人之一奥古斯都·德·摩根 (Augustus de Morgan, 1806—1871),因第一个为分析机编出了程序即“第一套计算机软件”,被誉为“世界上第一位软件工程师”。Ada 为查尔斯·巴贝奇 (Charles Babbage, 1791—1871) 的分析机 (Analytic Machine) 编写流程,其中包括计算三角函数、级数相乘、伯努利方程等。

在 20 世纪 40 年代末,随着埃尼阿克 (Electronic Numerical Integrator and Calculator, ENIAC) 问世,以写软件为职业的人开始出现,他们多是经过训练的数学家和电子工程师。到了 20 世纪 60 年代,美国大学里开始出现专门教授人们编写软件的专业,并且对该专业毕业的大学生、研究生授予计算机专业的学位。随着信息产业的迅速发展,软件对人类社会的作用也显得越来越重要,人们对软件的认识也更为深刻。



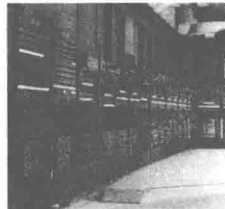
奥古斯塔·爱达·金



乔治·戈登·拜伦



查尔斯·巴贝奇



ENIAC

在发展过程中,软件技术主要经历了四个发展阶段,表 1-1 给出了这四个阶段典型技术的比较。

表 1-1

四个阶段典型技术比较

阶段	第一阶段	第二阶段	第三阶段	第四阶段
典型技术	面向批处理 有限的分布 自定义软件	多用户 实时 数据库 软件产品	分布式系统 嵌入“智能” 低成本硬件 消费者的影响	强大的桌面系统 面向对象技术 专家系统、人工神经网络 并行计算、网格计算

1.1.3 软件的分类型

为了便于人们根据不同的应用要求选择相应的软件,也鉴于不同类型的工程对象对软件的开发和维护有着不同的要求和处理方法,对软件进行分类是必要的。但是人们对软件的关心和侧重点有所不同,难以找到一种统一的严格分类标准。

软件根据规模可分为微型软件、小型软件、中型软件、大型软件、甚大型软件、极大型软件;根据工作方式可分为实时处理软件、分时软件、交互式软件和批处理软件;根据功能可分为系统软件、支撑软件和应用软件。

(1) 系统软件

系统软件是与计算机硬件紧密配合,使计算机的硬件与相关软件及数据进行协调、高效工作的系统,如操作系统、数据库管理系统、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,它频繁地与硬件交互,通过进程管理和数据结构处理,为用户提供服务。

(2) 支撑软件

支撑软件是协助用户开发软件的工具性软件，包括帮助程序员开发软件产品的工具和帮助管理人员控制开发进程的工具。表 1-2 给出了一些支撑软件的实例。

表 1-2 支撑软件举例

类型	支撑软件
一般类型	文本编辑程序 文本格式化程序 磁盘向磁带做数据传输程序 程序库系统
支持需求分析	PSI/PSA 问题描述分析器 关系数据库系统 一致性检验系统 CARA 计算机辅助需求分析器
支持设计	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
支持实现	编辑程序 交叉编辑程序 预编译程序 连接逻辑程序
支持测试	静态分析程序 符号执行程序 模拟程序 测试覆盖检验程序
支持管理	PERT 进度计划评审方法绘图程序 标准检验程序 库管理程序

(3) 应用软件

应用软件是在特定领域内开发、为特定目标提供服务的一类软件，其中商业数据处理软件占很大比例。另外有工程与科学计算软件、计算机辅助设计（Computer Aided Design, CAD）、计算机辅助制造（Computer Aided Manufacturing, CAM）软件、系统仿真软件、智能产品嵌入软件（如汽车油耗系统、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。此外，事务管理、办公自动化、中文信息处理、计算机辅助教学（Computer Aided Instruction, CAI）等方面的软件也得到了迅速发展。

1.1.4 软件的应用领域

Netscape 创始人、硅谷著名投资人马克·安德森（Marc Andreessen, 1971—）在 2011 年 8 月 21 日华尔街日报上发表的《软件正在吞噬整个世界》一文中称，当今的软件应用无所不在，并且正在吞噬整个世界。同年，惠普公司宣布将放



马克·安德森

弃当时处于步履维艰的 PC 业务,转而致力于具有更大增长潜力的软件业务。60 年前的计算机革命,40 年前的微处理器发明,20 年前的互联网兴起,所有这些技术最终都通过软件改变各个行业,并且在全球范围得到推广。

软件还正在吞噬许多被广泛认为主要存在于实体世界的行业价值链。今天的汽车里,软件操作着引擎,控制着安全功能,给乘客带来娱乐,引导驾驶员向目的地行驶,实现每辆汽车与移动设备、卫星和全球定位系统(Global Positioning System, GPS)网络相连接。目前,电动汽车的发展趋势将会加速向软件转移,完全由电脑控制。

实体店零售商沃尔玛,利用软件提升了它的后勤和配送能力,增强了竞争力。国际性速递集团联邦快递也同样如此,其卡车、飞机和配送中心组成了被认为是最好的软件网络。今天以及未来,航空公司的成败将取决于它们凭借软件正确地定价机票、优化路线及计算收益的能力。

此外,金融服务业、医疗卫生以及教育行业等也都进入软件驱动的时代。每个行业的公司都必须做好迎接软件革命到来的准备,其中甚至包括那些今天已经具有了软件基础的行业。

1.2 软件危机

1.2.1 软件危机的定义

1968 年北大西洋公约组织(North Atlantic Treaty Organization, NATO)的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“软件危机”(Software Crisis)这个名词,与会人员最后得出结论:软件工程应当使用已建立的工程学科的基本原理和范型(Paradigm,即方法示例)来解决所谓的软件危机;顾名思义,软件危机指软件产品的质量低得通常不能接受,并且不能满足交付日期和预算限制。



NATO

目前,概括来说,软件危机包含两方面问题:一方面是如何开发软件,以满足不断增长、日益复杂的需求;另一方面是如何维护数量不断膨胀的软件产品。鉴于软件危机的长期性和症状不明显的特征,近年来也有“软件萧条(Depression)”或者“软件困扰(Affliction)”的说法。不过,“软件危机”强调了问题的严重性,也为大多数软件工作者熟悉。

具体来说,软件危机有以下一些典型表现。

(1) 对软件开发成本和进度的估计常常不准确。

拖延几个月甚至几年工期的现象并不罕见,这种现象降低了软件开发组织的信誉。以丹佛新国际机场为例,按原定计划要在 1993 年万圣节前启用,但一直到 1994 年 6 月,机场的计划者还无法预测系统何时能达到可使机场开放的稳定程度。

(2) 用户对“已完成”系统不满意的现象经常发生。

软件开发人员和用户间的交流不充分,造成开发人员对用户的需求一知半解,仓促编写程序,最终导致产品与用户期望值的差距过大。

(3) 软件产品的质量往往靠不住。

Bug 一大堆, Patch 一个接一个,软件质量保证技术并没有完全应用到软件开发的全过程中,软件产品的质量也就无从保证。

1979 年 11 月 9 日检测到一个绝对称不上幽默的软件错误。北美防空防天司令部收到由全球军事指挥控制系统(World Wide Military Command and Control System, WWMCCS)计算机系统网络发出的警报,警报显示苏联已经向美国发射导弹,而实际上,该警报是由于软件错误引起的。

(4) 软件的可维护程度非常低。

实时的现实世界在不停地变化, 而许多程序的错误难以改正, 更不可能使这些程序适应新的硬件环境, 也不可能根据用户需要在原有程序上增加新功能。“可重用软件”仍有很长的一段路要走。

(5) 软件通常没有适当的文档数据。

软件不仅是程序, 还应该有完整的文档数据。这些文档数据应该是在软件开发过程中产生出来的, 而且应该是和代码程序完全一致的, 对于软件开发组织的管理人员、开发人员、维护人员而言都是至关重要、必不可少的。缺乏必要的文档数据, 必然给软件开发和维护带来许多困难。

(6) 软件的成本不断提高。

随着微电子技术的进步和生产自动化的不断发展, 硬件成本逐年下降, 然而软件开发需要大量人力, 软件成本所占比例持续上涨。

(7) 软件开发生产率的提高赶不上硬件的发展和人们需求的增长。

软件产品的“供不应求”现象使人类不能充分利用现代计算机硬件提供的巨大潜力。

以上列举的仅仅是软件危机的一些典型表现, 事实上, 软件危机带给软件开发和维护的问题远不止这些。

1.2.2 软件危机产生的原因

20 世纪 60 年代中期到 20 世纪 70 年代中期, 人们以“软件作坊”的形式开发软件。开发的方法基本上仍然沿用早期的个性化软件开发方式, 当软件的数量急剧膨胀, 软件需求日趋复杂, 维护的难度越来越大, 开发成本越来越高, 失败的软件开发项目屡见不鲜, “软件危机”就这样开始了。“软件危机”使得人们开始对软件及其特性进行更进一步的研究, 人们改变了早期对软件的不正确看法, 那些被认为很难被别人看懂、通篇充满了程序技巧和窍门的程序不再是优秀的程序。而除了功能正确、性能优良之外, 容易看懂、容易使用、容易修改和扩充的程序才是真正优秀的程序。

在软件开发和维护工作中存在如此之多的严重问题, 一方面与软件本身的特点有关, 另一方面与软件开发和维护的不正确方法有关。总体来说, 有如下 5 点。

(1) 忽视软件开发前期的需求分析。

(2) 开发过程没有统一的、规范的方法论的指导, 文件资料不齐全, 忽视人与人的交流。

(3) 忽视测试阶段的工作, 提交用户的软件质量差。

(4) 忽视软件的维护。

(5) 缺少规范而盲目编写程序。

1.2.3 软件危机解决的途径

为了更好地解决软件危机带来的各种问题, 首先应该对软件有一个比较全面的认识。1983 年美国电气电子工程师学会 (Institute of Electrical and Electronics Engineers, IEEE) 对软件的定义为“计算机程序、方法、规则、相关的文档数据以及在计算机上运行程序必需的数据”。由此可以看出软件其实包含五个配置部分, 其中方法和规则是在文档中说明, 并由程序加以实现。软件开发是一种组织良好、管理严格、各类人员协同配合、共同完成的工程项目。软件危机的解决途径可以从以下两方面着手。

(1) 应该推广在实践中总结出来的开发软件的成功技术和方法, 并且探索更好的、更有效的技术和方法, 尽快纠正计算机系统早期发展阶段形成的关于软件开发的错误概念。

(2) 应该使用更好的软件工具。在适当的软件工具支持下, 开发人员可以更好地完成工作。

总之, 按工程化的原则和方法组织软件开发工作是有效的, 也是摆脱软件危机的一个主要方法。

1.3 软件工程

1.3.1 软件工程的定义和研究对象

著名的软件工程专家巴利·玻姆 (Barry W. Boehm, 1935—) 把软件工程定义为“运用现代科学技术知识来设计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数据”。这个定义指出了软件工程包括了在成本限额内按时完成开发和修改软件, 也指出软件工程应当包含技术和管理两方面的目标。



巴利·玻姆

弗里德里希·鲍尔 (Friedrich L. Bauer, 1924—2015) 把软件工程定义为“建立并使用完善的工程化原则, 以较经济的手段获得能在实际机器上有效运行的



弗里德里希·鲍尔

可靠软件的一系列方法”。这个定义是将系统工程的原理应用到软件的开发和维护中, 认为软件工程应当有完善的工程化原则。

1993年IEEE进一步给出了更全面的定义, 即软件工程把系统化的、规范化的、可度量的途径应用于软件开发、运行和维护的过程中, 也就是把工程化应用于软件中。

软件工程是一门研究如何用系统化、规范化、数量化等工程原则和方法进行软件开发和维护的学科, 包括软件开发技术和软件项目管理。

从上面给出的各种软件工程的定义可以看出, 实际上软件工程的具体研究对象就是软件系统。它包括了方法、工具和过程3个要素。

1.3.2 软件工程的基本原理

自从1968年提出“软件工程”这一术语以来, 研究软件工程的专家们陆续提出了100多条关于软件工程的准则或信条。美国著名的软件工程专家巴利·玻姆综合这些专家的意见, 并总结多年的开发软件经验, 于1983年提出了软件工程的7条基本原理。

(1) 用分阶段的生存周期计划严格管理。

在软件开发与维护的漫长过程中, 需要完成许多性质各异的工作。这条原理指出, 应该把软件生存周期分成若干阶段, 并相应制订出切实可行的计划, 然后严格按照计划对软件的开发和维护进行管理。

(2) 坚持进行阶段评审。

统计结果表明, 大部分错误是发生在编码之前的, 大约占63%, 错误发现得越晚, 改正它要付出的代价就越大。因此, 软件的质量保证工作不能在编码结束之后进行, 应坚持进行严格的阶段评审, 以便尽早发现错误。

(3) 实行严格的产品控制。

开发人员最痛恨的事情之一就是改动需求。但是实践证明, 需求的改动往往是不可避免的。这就要求采用科学的产品控制技术来顺应这种要求, 也就是要采用变动控制和基准配置管理。当需求变动时, 其他各个阶段的文档或代码随之相应变动, 以保证软件的一致性。

(4) 采纳现代程序设计技术。

从20世纪60年代的结构化软件开发技术, 到现在的面向对象技术, 从第一、第二代语言, 到第四代语言, 人们已经充分认识到方法大于气力。采用先进的技术既可以提高软件开发的效率, 又