普通高等教育“十一五”国家级规划教材
软件工程专业核心课程系列教材

软件体系结构
原理、方法与实践
（第2版）

张友生 编著

清华大学出版社

普通高等教育“十一五”国家级规划教材
软件工程专业核心课程系列教材

软件体系结构原理、方法与实践
（第2版）

张友生 编著

清华大学出版社
北京

内 容 简 介

本书系统地介绍了软件体系结构的基本原理、方法和实践，全面反映了软件体系结构研究和应用的最新进展。既讨论软件体系结构的基本理论知识，又介绍软件体系结构的设计和产业界应用实例，强调理论与实践相结合，基础知识与前沿发展相结合。

全书共13章，第1章简单地介绍软件体系结构的概念、发展和应用现状；第2章讨论软件体系结构建模，包括“4＋1”视图模型、核心模型、生命周期模型和抽象模型；第3章介绍软件体系结构的风格和特定领域软件体系结构；第4章讨论软件体系结构的描述方法，重点介绍软件体系结构描述语言；第5章介绍UML的基础知识，以及如何使用UML及其扩展机制对软件体系结构建模；第6章介绍XML相关知识，重点讨论基于XML的软件体系结构描述语言；第7章介绍动态软件体系结构及其描述方法；第8章讨论基于服务的体系结构的关键技术和实现方法，以及如何构建一个服务，并给出了一个应用实例；第9章讨论富互联网应用体系结构，重点介绍AJAX和Mashup技术；第10章讨论软件体系结构的分析与测试问题，重点介绍软件体系结构的可靠性风险分析；第11章讨论软件体系结构评估方法，重点介绍ATAM和SAAM方法；第12章讨论基于体系结构的软件开发，包括设计模式、中间件技术和基于体系结构的软件过程；第13章介绍软件产品线的原理和方法、框架技术，重点讨论产品线体系结构的设计和演化。

本书可作为计算机软件相关专业高年级本科生、硕士研究生（含软件工程硕士）和博士研究生的软件体系结构教材，作为软件工程高级培训、系统分析师和系统架构设计师培训教材，也可作为高级软件开发人员的参考书。

图书在版编目(CIP)数据

软件体系结构原理、方法与实践/张友生编著. —2版. —北京：清华大学出版社，2014
软件工程专业核心课程系列教材
ISBN 978-7-302-33504-7

Ⅰ. ①软… Ⅱ. ①张… Ⅲ. ①软件－系统结构 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2013)第189145号

责任编辑：魏江江　薛　阳
封面设计：常雪影
责任校对：梁　毅
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954

印 装 者：北京市清华园胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：24.5　　**字　　数**：610千字
版　　次：2009年8月第1版　2014年1月第2版　　**印　　次**：2014年1月第1次印刷
印　　数：12501～14500
定　　价：39.50元

产品编号：040144-01

前　言

体系结构(architecture)一词在英文里就是“建筑”的意思，软件产业界通常翻译为“架构”。把软件比作一座楼房，从整体上讲，是因为它有基础、主体和装饰，即操作系统之上的基础设施软件、实现计算逻辑的主体应用程序、方便使用的用户界面程序。从细节上看每一个程序也是有结构的。早期的结构化程序就是以语句组成模块，模块的聚集和嵌套形成层层调用的程序结构，也就是体系结构。结构化程序的程序(表达)结构和(计算的)逻辑结构的一致性及自顶向下开发方法自然而然地形成了体系结构。

1. 为什么要学习软件体系结构

由于结构化程序设计时代程序规模不大，通过强调结构化程序设计方法学，自顶向下、逐步求精，并注意模块的耦合性就可以得到相对良好的结构，所以，并未特别研究软件体系结构。

随着软件系统规模越来越大、越来越复杂，整个系统的结构和规格说明显得越来越重要。对于大规模的复杂软件系统来说，对总体的系统结构设计和规格说明比起对计算的算法和数据结构的选择已经变得明显重要得多。在此种背景下，人们认识到软件体系结构的重要性，并认为对软件体系结构的系统、深入的研究将会成为提高软件生产率和解决软件维护问题的新的最有希望的途径。

对于软件项目的开发来说，一个清晰的软件体系结构是首要的。传统的软件开发过程可以划分为从概念直到实现的若干个阶段，包括问题定义、需求分析、软件设计、软件实现及软件测试等。软件体系结构的建立应位于需求分析之后，软件设计之前。但在传统的软件工程方法中，需求和设计之间存在一条很难逾越的鸿沟，从而很难有效地将需求转换为相应的设计。而软件体系结构就是试图在软件需求与软件设计之间架起一座桥梁，着重解决软件系统的结构和需求向实现平坦地过渡的问题。

2. 本书的内容

本书共分为 13 章。

第 1 章简单地介绍软件体系结构的概念、发展和应用现状。

第 2 章讨论软件体系结构建模，包括“4＋1”视图模型、核心模型、生命周期模型和抽象模型。

第 3 章介绍软件体系结构的风格和特定领域软件体系结构。

第 4 章讨论软件体系结构的描述方法，重点介绍软件体系结构描述语言。

第 5 章介绍 UML 的基础知识，以及如何使用 UML 及其扩展机制对软件体系结构建模。

第 6 章介绍 XML 相关知识，重点讨论基于 XML 的软件体系结构描述语言。

第 7 章介绍动态软件体系结构及其描述方法。

第8章讨论基于服务的体系结构的关键技术和实现方法,以及如何构建一个服务,并给出了一个应用实例。

第9章讨论富互联网应用体系结构,重点介绍AJAX和Mashup技术。

第10章讨论软件体系结构的分析与测试问题,重点介绍软件体系结构的可靠性风险分析。

第11章讨论软件体系结构评估方法,重点介绍ATAM和SAAM方法。

第12章讨论基于体系结构的软件开发,包括设计模式、中间件技术和基于体系结构的软件过程。

第13章介绍软件产品线的原理和方法、框架技术,重点讨论产品线体系结构的设计和演化。

本书由张友生主编,参与编写工作的有李雄、王勇、桂阳、谢顺、彭雪阳、刘洋波、何玉云、胡光超、左水林、胡钊源。

3. 将本书作为教材

如果将本书作为教材,则根据学生的不同层次,教师可以选讲其中一部分内容,下面列出一些建议,供教师们参考。

(1) 本科生:建议将软件体系结构作为选修课程,安排24课时甚至更少的课时,主要讲解软件体系结构的发展和图形化建模,以及目前比较流行的一些体系结构风格和模式。可以不讲的章节包括2.2节、2.3节、2.4节,3.5节、3.6节、3.7节、3.9节、3.10节,第4章,5.10节、5.11节,6.4节,第7章,第10章,第11章,12.3节、12.5节、12.7节、12.8节,第13章。

(2) 硕士研究生:建议将软件体系结构作为选修课程(如果是计算机软件与理论专业的硕士研究生,则建议作为必修课程)。硕士研究生需要掌握软件体系结构的一些基本理论和常用的研究方法,侧重于软件体系结构学术性方面的内容。可以不讲的章节包括第5章,第6章,第8章,第9章,第12章,第13章。

(3) 软件工程硕士:建议将软件体系结构作为必修课程。软件工程硕士更侧重实践性,需要掌握软件体系结构的基本理论,软件体系结构的图形化建模,以及产业界常用的一些体系结构的原理及应用,掌握软件体系结构的发展趋势。可以不讲的章节包括2.2节、2.3节、2.4节,3.6节、3.9节,第4章,5.11节,6.4节,第7章,第10章,12.7节。

(4) 博士研究生:建议将软件体系结构作为选修课程,或采用报告会议的形式,将本书的所有章节(或部分章节)作为专题报告,对计算机软件相关专业的博士研究生进行教学。

4. 诚挚致谢

在本书出版之际,要特别感谢国内外软件工程和软件体系结构专著、教材和许多高水平论文、报告的作者们(恕不一一列举,名单详见各章中的主要参考文献),他们的作品为本书提供了丰富的营养,使我们受益匪浅。在本书中引用了他们的部分材料,使本书能够尽量反映软件体系结构研究和实践领域的最新进展。

感谢阅读本书以前版本的读者,特别要感谢使用本书作为教材的教师,他们为本书的修订和出版提出了宝贵的意见。

感谢希赛教育网(www.educity.cn)为本书的意见反馈提供了空间和程序,感谢清华大学出版社魏江江主任的帮助,他们在本书的选题、编辑和定稿方面,给予了我们很大的帮助。

由于作者水平有限,时间紧迫,加上软件体系结构是一门新兴的学科,本身发展很快,对有些新领域作者尚不熟悉。因此,书中难免有不妥和疏漏之处,我们诚恳地期望各位专家和读者不吝指教和帮助。对此,我们将深为感激。

2013 年 5 月

目　录

第 1 章　软件体系结构概论 …… 1

1.1　从软件危机谈起 …… 1
1.1.1　软件危机的表现 …… 1
1.1.2　软件危机的原因 …… 2
1.1.3　如何克服软件危机 …… 3
1.2　构件与软件重用 …… 3
1.2.1　构件模型及实现 …… 4
1.2.2　构件获取 …… 5
1.2.3　构件管理 …… 6
1.2.4　构件重用 …… 9
1.2.5　软件重用实例 …… 14
1.3　软件体系结构的兴起和发展 …… 17
1.3.1　软件体系结构的定义 …… 18
1.3.2　软件体系结构的意义 …… 19
1.3.3　软件体系结构的发展史 …… 21
1.4　软件体系结构的应用现状 …… 22
思考题 …… 27
主要参考文献 …… 28

第 2 章　软件体系结构建模 …… 29

2.1　“4＋1”视图模型 …… 29
2.1.1　逻辑视图 …… 30
2.1.2　开发视图 …… 31
2.1.3　进程视图 …… 32
2.1.4　物理视图 …… 33
2.1.5　场景 …… 35
2.2　软件体系结构的核心模型 …… 36
2.3　软件体系结构的生命周期模型 …… 37
2.3.1　各阶段之间的关系 …… 37
2.3.2　软件体系结构的生命周期 …… 38
2.4　软件体系结构抽象模型 …… 40
2.4.1　构件及其关系的抽象描述 …… 40
2.4.2　连接件 …… 44

2.4.3 软件体系结构 …… 44
2.4.4 软件体系结构关系 …… 45
2.4.5 软件体系结构范式 …… 46
思考题 …… 48
主要参考文献 …… 48

第3章 软件体系结构风格 …… 50

3.1 经典软件体系结构风格 …… 50
3.1.1 管道与过滤器 …… 51
3.1.2 数据抽象和面向对象系统 …… 52
3.1.3 基于事件的系统 …… 52
3.1.4 分层系统 …… 53
3.1.5 仓库系统及知识库 …… 54
3.1.6 C2风格 …… 55
3.2 客户/服务器风格 …… 56
3.3 三层C/S结构风格 …… 58
3.3.1 各层的功能 …… 59
3.3.2 三层C/S结构应用实例 …… 61
3.3.3 三层C/S结构的优点 …… 65
3.4 浏览/服务器风格 …… 66
3.5 公共对象请求代理体系结构 …… 67
3.5.1 CORBA技术规范 …… 68
3.5.2 CORBA风格分析 …… 68
3.6 正交软件体系结构 …… 70
3.6.1 正交软件体系结构的抽象模型 …… 71
3.6.2 软件体系结构的正交化 …… 72
3.6.3 正交软件体系结构的实例 …… 73
3.6.4 正交软件体系结构的优点 …… 76
3.7 基于层次消息总线的体系结构风格 …… 77
3.7.1 构件模型 …… 78
3.7.2 构件接口 …… 79
3.7.3 消息总线 …… 79
3.7.4 构件静态结构 …… 81
3.7.5 构件动态行为 …… 81
3.7.6 运行时刻的系统演化 …… 82
3.8 异构结构风格 …… 82
3.8.1 异构结构的实例分析 …… 83
3.8.2 异构组合匹配问题 …… 85
3.9 互连系统构成的系统及其体系结构 …… 87

3.9.1　互连系统构成的系统 …… 87
3.9.2　基于 SASIS 的软件过程 …… 88
3.9.3　应用范围 …… 90
3.10　特定领域软件体系结构 …… 92
3.10.1　DSSA 的定义 …… 92
3.10.2　DSSA 的基本活动 …… 93
3.10.3　参与 DSSA 的人员 …… 94
3.10.4　DSSA 的建立过程 …… 95
3.10.5　DSSA 实例 …… 96
3.10.6　DSSA 与体系结构风格的比较 …… 99
思考题 …… 100
主要参考文献 …… 102

第 4 章　软件体系结构描述 …… 104

4.1　软件体系结构描述方法 …… 104
4.2　软件体系结构描述框架标准 …… 106
4.3　体系结构描述语言 …… 106
4.3.1　ADL 与其他语言的比较 …… 107
4.3.2　ADL 的构成要素 …… 108
4.4　典型的软件体系结构描述语言 …… 110
4.4.1　UniCon …… 111
4.4.2　Wright …… 112
4.4.3　C2 …… 113
4.4.4　Rapide …… 117
4.4.5　SADL …… 118
4.4.6　Aesop …… 118
4.4.7　ACME …… 120
思考题 …… 126
主要参考文献 …… 126

第 5 章　统一建模语言 …… 128

5.1　UML 概述 …… 128
5.1.1　UML 的发展历史 …… 128
5.1.2　UML 的应用领域 …… 129
5.2　UML 的结构 …… 130
5.2.1　结构概述 …… 130
5.2.2　事物 …… 131
5.2.3　关系 …… 132
5.2.4　图形 …… 133

5.3 用例图 …… 135
5.4 类图和对象图 …… 136
5.5 交互图 …… 137
5.5.1 顺序图 …… 138
5.5.2 通信图 …… 138
5.5.3 定时图 …… 139
5.6 状态图 …… 140
5.7 活动图 …… 141
5.7.1 基本活动图 …… 141
5.7.2 带泳道的活动图 …… 142
5.7.3 交互概览图 …… 142
5.8 构件图 …… 143
5.9 部署图 …… 144
5.10 使用 UML 建模 …… 145
5.11 使用 UML 的扩展机制 …… 149
思考题 …… 152
主要参考文献 …… 153

第6章 可扩展标记语言 …… 154

6.1 XML 概述 …… 154
6.1.1 XML 的特点 …… 155
6.1.2 XML 的作用 …… 156
6.1.3 XML 的应用 …… 158
6.2 解析 XML …… 158
6.2.1 XML 与 HTML 的区别 …… 159
6.2.2 XML 文档 …… 160
6.2.3 CSS 与 XSL …… 162
6.3 XML 编程接口 …… 165
6.3.1 API …… 165
6.3.2 XML 开发工具 …… 167
6.3.3 XML 建模 …… 168
6.4 基于 XML 的软件体系结构描述语言 …… 168
6.4.1 XADL 2.0 …… 169
6.4.2 XBA …… 173
思考题 …… 175
主要参考文献 …… 176

第7章 动态软件体系结构 …… 177

7.1 动态软件体系结构概述 …… 177

7.2　基于构件的动态系统结构模型 …… 179
7.3　πADL 动态体系结构 …… 182
7.3.1　πADL 描述体系结构的框架 …… 183
7.3.2　πADL 动态体系结构建模方法 …… 184
7.3.3　πADL 动态体系结构建模语义 …… 186
7.4　动态体系结构的描述 …… 187
7.4.1　动态体系结构描述语言 …… 188
7.4.2　动态体系结构的形式化描述 …… 189
7.5　动态体系结构的特征 …… 191
7.6　化学抽象机 …… 192
思考题 …… 195
主要参考文献 …… 196

第 8 章　基于服务的体系结构 …… 197

8.1　SOA 概述 …… 197
8.2　面向服务的分析与设计 …… 199
8.3　SOA 的关键技术 …… 201
8.4　SOA 的实现方法 …… 202
8.5　服务描述语言 …… 205
8.5.1　WSDL 概述 …… 205
8.5.2　使用 WSDL 文档 …… 206
8.5.3　WSDL 文档结构 …… 207
8.6　统一描述、发现和集成协议 …… 214
8.6.1　UDDI 数据模型 …… 214
8.6.2　注册 Web 服务 …… 218
8.6.3　调用 Web 服务 …… 218
8.7　消息封装协议 …… 219
8.7.1　消息封装和编码规则 …… 219
8.7.2　SOAP 应用 …… 221
8.7.3　REST …… 224
8.8　构造一个简单的服务 …… 224
8.8.1　编写服务器端 …… 224
8.8.2　编写客户端 …… 225
8.9　Web 服务的应用实例 …… 226
思考题 …… 229
主要参考文献 …… 231

第 9 章　富互联网应用体系结构 …… 233

9.1　RIA 的概念 …… 233

9.1.1 RIA的提出 …… 233
9.1.2 丰富的含义 …… 234
9.1.3 RIA的优点 …… 234
9.2 RIA模型 …… 235
9.3 RIA客户端开发技术 …… 237
9.4 AJAX技术 …… 239
9.5 Mashup技术 …… 240
9.5.1 Mashup的体系结构 …… 240
9.5.2 Mashup实现技术 …… 242
思考题 …… 243
主要参考文献 …… 244

第10章 软件体系结构的分析与测试 …… 245

10.1 体系结构的可靠性建模 …… 245
10.2 软件体系结构的风险分析 …… 249
10.2.1 风险分析的方法 …… 249
10.2.2 风险分析的步骤 …… 250
10.3 基于体系结构描述的软件测试 …… 255
10.3.1 测试方法 …… 255
10.3.2 实例与实现 …… 256
思考题 …… 258
主要参考文献 …… 258

第11章 软件体系结构评估 …… 259

11.1 软件体系结构评估概述 …… 259
11.1.1 软件质量属性 …… 259
11.1.2 几个基本概念 …… 261
11.1.3 评估的主要方式 …… 263
11.2 ATAM评估方法 …… 264
11.2.1 ATAM评估的步骤 …… 265
11.2.2 ATAM评估的阶段 …… 271
11.3 SAAM评估方法 …… 274
11.3.1 SAAM评估的步骤 …… 274
11.3.2 SAAM评估实例 …… 278
思考题 …… 281
主要参考文献 …… 282

第12章 基于体系结构的软件开发 …… 283

12.1 设计模式 …… 283

12.1.1　设计模式概述 …… 283
12.1.2　设计模式的组成 …… 285
12.1.3　设计模式的描述 …… 287
12.1.4　模式和软件体系结构 …… 288
12.1.5　设计模式的层次 …… 289
12.1.6　设计模式的分类 …… 291
12.1.7　设计模式示例 …… 294
12.1.8　MVC 模式的设计与实现 …… 297
12.2　中间件技术 …… 300
12.2.1　中间件概述 …… 300
12.2.2　主要的中间件 …… 302
12.2.3　中间件与构件的关系 …… 306
12.3　基于体系结构的设计方法 …… 307
12.3.1　有关术语 …… 308
12.3.2　ABSD 方法与生命周期 …… 309
12.3.3　ABSD 方法的步骤 …… 311
12.4　体系结构的设计与演化 …… 318
12.4.1　设计和演化过程 …… 318
12.4.2　实验原型阶段 …… 319
12.4.3　演化开发阶段 …… 321
12.5　基于体系结构的软件开发模型 …… 321
12.5.1　体系结构需求 …… 322
12.5.2　体系结构设计 …… 323
12.5.3　体系结构文档化 …… 324
12.5.4　体系结构复审 …… 324
12.5.5　体系结构实现 …… 324
12.5.6　体系结构演化 …… 325
12.6　应用开发实例 …… 326
12.6.1　系统简介 …… 326
12.6.2　系统设计与实现 …… 329
12.6.3　系统演化 …… 331
12.7　基于体系结构的软件过程 …… 332
12.7.1　有关概念 …… 332
12.7.2　软件过程网 …… 333
12.7.3　基本结构的表示 …… 335
12.7.4　基于体系结构的软件过程 Petri 网 …… 336
12.8　软件体系结构演化模型 …… 341
12.8.1　SA 静态演化模型 …… 341
12.8.2　SA 动态演化模型 …… 343

思考题 …… 346
主要参考文献 …… 348

第13章 软件产品线体系结构 …… 350

13.1 软件产品线的出现和发展 …… 350
13.1.1 软件体系结构的发展 …… 350
13.1.2 软件重用的发展 …… 351
13.2 软件产品线概述 …… 352
13.2.1 软件产品线的过程模型 …… 353
13.2.2 软件产品线的组织结构 …… 355
13.2.3 软件产品线的建立方式 …… 356
13.2.4 软件产品线的演化 …… 357
13.3 框架和应用框架技术 …… 358
13.4 软件产品线基本活动 …… 360
13.4.1 产品线分析 …… 361
13.4.2 产品开发 …… 363
13.5 软件产品线体系结构的设计 …… 364
13.5.1 产品线体系结构简介 …… 364
13.5.2 产品线体系结构的标准化和定制 …… 366
13.6 软件产品线体系结构的演化 …… 367
13.6.1 背景介绍 …… 367
13.6.2 两代产品的各种发行版本 …… 369
13.6.3 需求和演化的分类 …… 372
思考题 …… 375
主要参考文献 …… 376

第 1 章　软件体系结构概论

软件开发不仅是把程序写好就够了，还要做前期的需求和体系结构设计(架构设计)，而且，前期的设计比程序编码更重要。关于这一点，初入行业的软件工程师时常会感到困惑，因为他们往往初出茅庐，没有大型项目经验。在他们的心中，只有程序，没有体系结构。

为了解决软件新人的这些困惑，本章从软件危机谈起，介绍软件构件、软件重用的基本概念，阐述软件体系结构的意义、发展和应用现状。

1.1　从软件危机谈起

软件危机(software crisis)是指在计算机软件的开发(development)和维护(maintenance)过程中所遇到的一系列严重问题。20 世纪 60 年代末至 20 世纪 70 年代初，“软件危机”一词在计算机界广为流传。事实上，几乎从计算机诞生的那一天起，就出现了软件危机，只不过到了 1968 年在原西德加密施(Garmish)召开的国际软件工程会议上才被人们普遍认识到。

1.1.1　软件危机的表现

时至今日，“软件危机”虽然在软件行业已经家喻户晓，但很多工程师还是不能很好地理解软件危机究竟为何物，在实际的软件项目中，软件危机的“表现”在哪里，本节就这些内容做一个简单的介绍。

1. 软件成本日益增长

在计算机发展的早期，大型计算机系统主要是被设计应用于非常狭窄的军事领域。在这个时期，研制计算机的费用主要由国家财政提供，研制者很少考虑到研制代价问题。随着计算机市场化和民用化的发展，代价和成本就成为投资者考虑的最重要的问题之一。

20 世纪 50 年代，软件成本(cost)在整个计算机系统成本中所占的比例为 10%～20%。但随着软件产业的发展，软件成本日益增长。相反，计算机硬件随着技术的进步、生产规模的扩大，价格却在不断下降。这样一来，软件成本在计算机系统中所占的比例越来越大。

到 20 世纪 60 年代中期，软件成本在计算机系统中所占的比例已经增长到 50%左右，而且该数字还在不断地递增。下面是一组来自美国空军计算机系统的数据：1955 年，软件费用约占总费用的 18%，1970 年达到 60%，1975 年达到 72%，1980 年达到 80%，1985 年达到 85%左右。而如今，购买一台普通的计算机只要两三千元人民币，但如果把常用的操作系统、办公软件、安全软件等装好，却要远远超过购买计算机的费用。

2. 开发进度难以控制

由于软件是逻辑、智力产品，软件的开发需建立庞大的逻辑体系，这与其他产品的生产是不一样的。例如，工厂里要生产某种机器，在时间紧的情况下可以要工人加班或者实行

“三班倒”,而这些方法都不能用在软件开发上。

在软件开发过程中,用户需求(requirement)变化等各种意想不到的情况层出不穷,令软件开发过程很难保证按预定的计划实现,给项目计划和论证工作带来了很大的困难。

Brooks在其名著《人月神话》中指出:“在已拖延的软件项目上,增加人力只会使其更难按期完成”。事实上,软件系统的结构很复杂,各部分附加联系极大,盲目增加软件开发人员并不能成比例地提高软件开发能力。相反,随着人员数量的增加,人员的组织、协调、通信、培训和管理等方面的问题将更为严重。

许多重要的大型软件开发项目,如IBM OS/360和世界范围的军事命令控制系统(WWMCCS),在耗费了大量的人力和财力之后,由于离预定目标相差甚远不得不宣布失败。

3. 软件质量差

软件项目即使能按预定日期完成,结果却不尽如人意。1965—1970年,美国范登堡基地发射火箭多次失败,绝大部分故障是由应用程序错误造成的。程序的一些微小错误可以造成灾难性的后果,例如,有一次,在美国肯尼迪发射一枚阿脱拉斯火箭,火箭飞离地面几十英里高空开始翻转,地面控制中心被迫下令炸毁。后经检查发现是飞行计划程序里漏掉了一个连字符。就是这样一个小小的疏漏造成了这支价值1850万美元的火箭试验失败。

在“软件作坊”里,由于缺乏工程化思想的指导,程序员几乎总是习惯性地以自己的想法去代替用户对软件的需求,软件设计带有随意性,很多功能只是程序员的“一厢情愿”而已,这是造成软件不能令人满意的重要因素。

4. 软件维护困难

正式投入使用的软件,总是存在着一定数量的错误,在不同的运行条件下,软件就会出现故障,因此需要维护。但是,由于在软件设计和开发过程中,没有严格遵循软件开发标准,各种随意性很大,没有完整的真实反映系统状况的记录文档,给软件维护造成了巨大的困难。特别是在软件使用过程中,原来的开发人员可能因各种原因已经离开原来的开发组织,使得软件几乎不可维护。

另外,软件修改是一项很“危险”的工作,对一个复杂的逻辑过程,哪怕做一项微小的改动,都可能引入潜在的错误,常常会发生“纠正一个错误带来更多新错误”的问题,从而产生副作用。

有资料表明,工业界为维护软件支付的费用占全部硬件和软件费用的40%～75%。

1.1.2　软件危机的原因

从软件危机的种种表现和软件作为逻辑产品的特殊性可以发现软件危机的原因,可以归结为如下4个方面。

1. 用户需求不明确

在软件开发过程中,需求多变是造成项目失败的最主要原因之一。具体来说,用户需求不明确问题主要体现在4个方面:

(1) 在软件开发出来之前,用户自己也不清楚软件的具体需求。

(2) 用户对软件需求的描述不精确,可能有遗漏、有二义性,甚至有错误。

(3) 在软件开发过程中,用户还提出修改软件功能(function)、界面(interface)、支撑环

境(environment)等方面的要求。

(4) 软件开发人员对用户需求的理解与用户本来的愿望有差异。

2. 缺乏正确的理论指导

缺乏有力的方法学和工具方面的支持。由于软件不同于大多数其他工业产品,其开发过程是复杂的逻辑思维过程,其产品极大程度地依赖于开发人员高度的智力投入。由于过分地依靠程序设计人员在软件开发过程中的技巧和创造性,加剧软件产品的个性化,也是发生软件危机的一个重要原因。

3. 软件规模越来越大

随着软件应用范围的增大,软件规模愈来愈大。大型软件项目需要组织一定的人力共同完成,而多数管理人员缺乏开发大型软件系统的经验,多数软件开发人员又缺乏管理方面的经验。各类人员的信息交流不及时、不准确,有时还会产生误解。软件项目开发人员不能有效地、独立自主地处理大型软件的全部关系和各个分支,因此容易产生疏漏和错误。

4. 软件复杂度越来越高

软件不仅是在规模上快速地发展扩大,而且其复杂性(complexity)也急剧地增加。软件产品的特殊性和人类智力的局限性,导致人们无力处理"复杂问题"。所谓"复杂问题"的概念是相对的,一旦人们采用先进的组织形式、开发方法和工具提高了软件开发效率和能力,新的、更大的、更复杂的问题又摆在人们的面前。

1.1.3　如何克服软件危机

人们在认真地研究和分析了软件危机背后的真正原因之后,得出了"人们面临的不单是技术问题,更重要的还是管理问题。管理不善必然导致失败"的结论,便开始探索用工程的方法进行软件生产的可能性,即用现代工程的概念、原理、技术和方法进行计算机软件的开发、管理和维护。于是,计算机科学技术的一个新领域——软件工程(software engineering)诞生了。

软件工程是用工程、科学和数学的原则与方法研制、维护计算机软件的有关技术及管理方法。软件工程包括三个要素:方法、工具和过程,其中:

(1) 软件工程方法为软件开发提供了"如何做"的技术,是完成软件工程项目的技术手段。

(2) 软件工具是人们在开发软件的活动中智力和体力的扩展和延伸,为软件工程方法提供了自动的或半自动的软件支撑环境。

(3) 软件工程过程则是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。

迄今为止,软件工程的研究与应用已经取得很大成就,它在软件开发方法、工具、管理等方面的应用大大缓解了软件危机造成的被动局面。

1.2　构件与软件重用

尽管当前社会的信息化过程对软件需求的增长非常迅速,但目前软件的开发与生产能力却相对不足,这不仅造成许多急需的软件迟迟不能被开发出来,而且形成了软件脱节现