高等学校计算机专业规划教材

Fundamentals of Software Engineering

软件工程基础

胡思康　编著

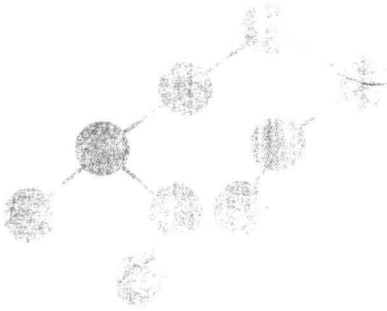
清华大学出版社

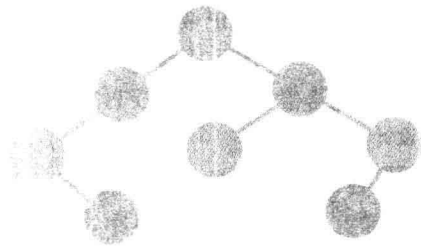
高等学校计算机专业规划教材

Fundamentals of Software Engineering

软件工程基础

胡思康　编著





清华大学出版社
北京

内 容 简 介

本书全面、系统地介绍了软件工程的基本概念、原理和典型的技术方法，在注重软件工程学科的系统性、原理性的同时，通过实际项目来增强读者对软件工程方法与技术在实践中的应用。本书共 11 章，第 1 章是对软件工程的概括。第 2 章至第 5 章按照软件生命周期的开发顺序，以结构化方法为主线，介绍软件工程各阶段的任务、过程、方法和工具。第 6 章介绍软件测试。第 7 章至第 9 章结合软件生命周期过程，以面向对象方法为主线，介绍 UML 统一建模语言、面向对象分析、面向对象设计等内容。第 10 章介绍软件维护。第 11 章介绍软件项目管理。

本书将软件工程教学和实践相结合，可作为高等院校计算机专业或信息类相关专业课程的教材或教学参考书，也能作为有一定实践经验的软件工程人员和需要开发应用软件的广大计算机用户的参考资料。

图书在版编目（CIP）数据

软件工程基础/胡思康编著. --北京：清华大学出版社，2012.7
（高等学校计算机专业规划教材）
ISBN 978-7-302-28317-1

Ⅰ. ①软… Ⅱ. ①胡… Ⅲ. ①软件工程－教材 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2012)第 044211 号

责任编辑：龙启铭　李玮琪
封面设计：常雪影
责任校对：时翠兰
责任印制：何　芊

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 刷 者：北京四季青印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.5　　**字　　数**：477 千字
版　　次：2012 年 7 月第 1 版　　**印　　次**：2012 年 7 月第1 次印刷
印　　数：1～3000
定　　价：29.50 元

产品编号：043554-01

《软件工程基础》 前言

软件是信息化的核心之一，软件产业展现国家科技发展的核心竞争力，体现国家的综合实力。随着计算机应用的不断普及、互联网应用的不断深入和网络技术的不断发展，软件系统的规模和复杂度也不断增加，如何确保开发出符合用户预期的、质量有保证的软件系统仍然是一个巨大挑战，软件危机的存在仍阻碍着软件的发展。

作为计算机科学技术的一个重要分支——软件工程学，成为软件需求、开发、维护、管理的普遍原理和技术相结合的、活跃的研究领域，随着软件工程的迅猛发展，新的技术、方法、工具不断涌现，为读者学习和研究这门学科创造了良好的基础和难得的机遇。

作为软件工程学的入门介绍，本书立足于基本的原理、概念、方法和工具，从实用的角度讲解软件系统需求、设计、实现、测试、维护和管理的内容，同时兼顾对软件工程过程介绍的全面性和系统性。

本书根据作者多年从事“软件工程”课程教学和软件开发的实践经验，在介绍相关理论和过程的基础上，着重讲解软件工程在实践中的方法、技术和工具。本书的特点体现在：

(1) 减少软件工程理论的阐述，避免对不同过程和方法的学术讨论。

(2) 介绍软件工程理论的基本概念和过程，它们对软件过程实践起着基石和指导的作用。

(3) 每章的小结对各章的主要内容进行总结，便于读者理解和掌握主要内容。

(4) 鉴于技术人员重技术而轻文档编写的实际情况，书中介绍了软件工程各阶段需要编写的文档框架，并通过实例不断强化文档对实施软件工程的重要性。

(5) 本书中的主要案例都来自于作者的研究和实际工程项目，让读者深切感受到书中介绍的理论是如何指导实践的。

本书以结构化设计思想为基础，全面介绍软件工程过程各阶段的过程、方法和工具，务求让读者对软件工程的实施有一个完整、清晰的认识。之后，再以面向对象设计思想为指导，详细介绍基于面向对象的软件工程开发过程。这样编排的目的，是使读者能通过对两类设计思想及软件过程的比较，不仅便于分析它们各自的优缺点，还能更深入理解相同的软件工程过程结合不同的软件设计思想，对软件分析、实现和维护的影响，以及对软件质量和管理发展的推动。

下面简要介绍本书各章节的概貌，让读者对本书内容有一个提纲挈领的了解。

第 1 章回顾了软件危机的产生，介绍软件工程的产生和发展，包括软件工程的基本概

念、目标和实施原则。通过对软件、软件生命周期和软件过程模型的介绍，让读者对软件工程的基本原理、方法、过程有一个基本认识。

第2章介绍软件需求工程的基本概念、任务和原则，并详细说明结构化分析和建模过程，包括面向数据的数据建模、面向数据流的功能建模和面向状态的行为建模。

第3章介绍软件设计的基本概念、任务和原则，以及目前主流的软件体系结构设计模型，它们分别是以数据为中心的数据仓库模型、客户端/服务器模式的分布式结构模型和层次模型。

第4章从应用角度出发，详细描述了结构化设计的两类设计方法：面向数据流的设计方法和面向数据结构的设计方法，以及它们的设计过程。

第5章从软件工程范畴讨论程序实现和编码，包括程序设计语言的分类、特性、准则及程序编写规范等。

第6章介绍进行软件测试的对象和测试技术。软件测试对象不仅包括源码，还包括设计方案、需求说明等软件工程文档。测试技术主要介绍了白盒测试和黑盒测试。

第7章介绍面向对象软件工程的建模基础。UML通过图形化的表示机制，对面向对象分析和设计提供统一的、标准化的视图、图、模型元素和通用机制来刻画面向对象方法。

第8章介绍面向对象分析的建模过程。面向对象分析模型主要由3种独立模型构成：功能模型、静态模型和动态模型。此外，还详细说明了作为建模基础的静态模型的5个层次。

第9章介绍把面向对象分析阶段得到的需求模型转换为符合用户功能、性能，便于用某种面向对象程序设计语言编程的系统实现方案。

第10章介绍作为软件工程最后一个阶段的软件维护的内容和过程，以及如何提高软件的可维护性和软件再工程。

第11章介绍有关软件项目管理的基本要求和内容。通过对软件项目的估算、项目进度管理、风险管理、质量管理、配置管理等内容的介绍，明确对软件工程全过程的计划、组织和控制等一系列活动，只有经过这一系列环节，才能得到符合用户需求的高质量、高可靠性的软件产品。

由于作者水平有限，疏漏、欠妥、谬误之处在所难免，恳请读者指正。读者如果对本书有任何意见和建议，欢迎和作者联系：skhu@163.com。

作　者

2012年1月

于北京理工大学

FOREWORD

第 1 章　软件工程概述　/1

1.1　软件工程的发展历程　/1

1.1.1　软件危机　/1

1.1.2　软件危机出现的原因　/3

1.1.3　软件工程的发展　/4

1.2　软件工程的概念　/5

1.2.1　软件工程的定义　/5

1.2.2　软件工程的目标　/6

1.2.3　软件工程的实施原则　/8

1.2.4　软件工程的基本原理　/9

1.3　软件与软件过程　/11

1.3.1　软件的概念　/11

1.3.2　软件的分类　/13

1.3.3　软件生命周期　/14

1.3.4　软件过程　/15

1.4　软件过程模型　/17

1.4.1　瀑布模型　/17

1.4.2　原型模型　/18

1.4.3　增量模型　/19

1.4.4　螺旋模型　/20

1.4.5　喷泉模型　/21

1.4.6　敏捷过程模型　/22

1.4.7　基于四代技术的过程模型　/23

1.4.8　微软解决框架过程模型　/24

1.4.9　组合模型的开发　/25

1.5　软件开发方法　/25

1.5.1　结构化开发方法　/26

1.5.2　面向对象开发方法　/26

1.6　案例描述　/27

1.6.1　简历信息自动获取和查询系统　/27

1.6.2 试卷自动生成系统 /28
1.7 小结 /29
习题 /31

第 2 章 软件需求工程 /32
2.1 软件需求的基本概念 /32
2.1.1 需求分析的任务 /32
2.1.2 需求分析的原则 /33
2.1.3 需求分析的内容 /34
2.2 需求工程的过程 /36
2.2.1 需求工程中的参与人员 /36
2.2.2 需求工程过程中的活动 /36
2.2.3 需求工程的管理 /38
2.3 需求获取技术 /39
2.4 结构化需求分析和建模 /41
2.4.1 结构化分析概述 /42
2.4.2 面向数据的数据建模 /42
2.4.3 面向数据流的功能建模 /44
2.4.4 面向状态转换的行为建模 /48
2.4.5 数据字典 /49
2.4.6 加工逻辑 /51
2.5 案例一 简历自动获取和查询系统的需求建模 /52
2.5.1 数据建模——ER 图描述 /53
2.5.2 功能建模——数据流图 /53
2.5.3 行为建模——状态转换图 /55
2.5.4 数据字典 /55
2.5.5 加工逻辑——PDL 语言的描述 /56
2.6 需求评审 /56
2.6.1 软件需求规格说明文档 /57
2.6.2 需求评审标准 /60
2.7 小结 /61

习题 /61

第 3 章 软件设计基础 /63

3.1 软件设计概述 /63

3.1.1 软件设计与软件需求 /63

3.1.2 软件设计的任务 /64

3.1.3 软件设计的原则 /66

3.2 软件体系结构设计 /67

3.2.1 体系结构设计概述 /67

3.2.2 以数据为中心的数据仓库模型 /68

3.2.3 客户端/服务器模式的分布式结构/69

3.2.4 层次模型 /71

3.3 模块化设计 /72

3.3.1 软件模块化与分解 /72

3.3.2 抽象 /73

3.3.3 信息隐藏 /73

3.3.4 模块独立性 /74

3.3.5 启发式规则 /76

3.4 界面设计 /79

3.4.1 界面设计的任务 /79

3.4.2 界面设计的原则 /80

3.4.3 界面设计的特性 /81

3.4.4 MVC 模型 /81

3.5 软件设计评审 /83

3.5.1 软件设计规格说明文档 /83

3.5.2 软件设计评审标准 /86

3.6 小结 /88

习题 /88

第 4 章 结构化设计方法 /90

4.1 结构化设计方法概述 /90

4.2 面向数据流的设计方法 /91
4.2.1 层次图和结构图 /91
4.2.2 变换分析法 /93
4.2.3 事务分析法 /97
4.2.4 混合分析法 /98
4.3 面向数据的设计方法 /99
4.3.1 Jackson 图 /99
4.3.2 Jackson 系统开发方法 /100
4.4 案例 简历自动获取和查询系统的数据流设计方法 /102
4.4.1 用变换分析法进行设计 /103
4.4.2 用事务分析法进行设计 /104
4.4.3 两种方法的比较 /105
4.5 结构化详细设计的工具 /106
4.5.1 程序流程图 /106
4.5.2 盒图(NS 图) /107
4.5.3 问题分析图 /108
4.5.4 判定树 /110
4.5.5 判定表 /110
4.5.6 详细设计工具的比较 /111
4.6 小结 /112
习题 /113

第 5 章 软件实现 /114
5.1 程序设计语言 /114
5.1.1 程序设计语言的分类 /114
5.1.2 程序设计语言的特性 /115
5.1.3 选择程序设计语言 /116
5.2 程序设计风格 /118
5.2.1 程序编排和组织的准则 /118
5.2.2 程序设计的效率 /122
5.3 代码复用 /124

5.4 代码评审 /125
5.5 小结 /129
习题 /129

第6章 软件测试 /131
6.1 软件测试基础 /131
6.1.1 软件测试的概念 /131
6.1.2 软件测试过程模型 /132
6.1.3 软件测试原则 /134
6.1.4 软件测试在软件开发各阶段的工作流程 /137
6.1.5 软件测试信息流 /138
6.1.6 软件测试技术分类 /139
6.2 白盒测试 /140
6.2.1 逻辑覆盖 /141
6.2.2 循环测试 /144
6.2.3 路径测试 /145
6.3 黑盒测试 /148
6.3.1 等价类划分 /148
6.3.2 边界值分析 /150
6.3.3 错误推测法 /150
6.3.4 因果图法 /151
6.4 白盒测试和黑盒测试的比较 /153
6.5 软件测试策略 /154
6.5.1 单元测试 /155
6.5.2 集成测试 /157
6.5.3 确认测试 /159
6.5.4 系统测试 /160
6.6 调试 /162
6.6.1 软件调试过程 /162
6.6.2 软件调试方法 /163
6.7 软件测试报告 /164

6.7.1 软件测试说明 /164
6.7.2 软件测试报告 /166
6.8 小结 /167
习题 /167

第7章 UML统一建模语言 /169
7.1 UML的发展 /169
7.1.1 UML的产生 /169
7.1.2 UML的构成 /170
7.1.3 UML的特点 /171
7.2 面向对象的基本概念 /171
7.3 UML视图 /177
7.4 UML的图和模型元素 /178
7.4.1 用例图 /179
7.4.2 类图 /180
7.4.3 包图 /182
7.4.4 状态图 /183
7.4.5 活动图 /183
7.4.6 顺序图 /185
7.4.7 协作图 /186
7.4.8 构件图 /187
7.4.9 配置图 /187
7.5 UML的关系 /188
7.5.1 关联关系 /188
7.5.2 泛化关系 /191
7.5.3 依赖关系 /195
7.5.4 实现关系 /196
7.6 UML的通用机制 /196
7.6.1 修饰 /196
7.6.2 注释 /197
7.6.3 规格说明 /197
7.6.4 扩展机制 /197

7.7 基于 UML 的软件过程 /199
7.8 小结 /201
习题 /202

第 8 章 面向对象分析 /203
8.1 面向对象分析概述 /203
8.1.1 传统软件过程中的不足 /203
8.1.2 面向对象的特点 /204
8.1.3 面向对象分析的基本过程 /205
8.1.4 面向对象分析的 3 类模型 /205
8.1.5 静态模型的 5 个层次 /206
8.2 建立功能模型 /207
8.2.1 识别参与者 /208
8.2.2 识别用例 /209
8.2.3 识别用例间关系 /210
8.2.4 用例描述文档 /211
8.3 建立静态模型(对象模型) /212
8.3.1 识别类与对象 /213
8.3.2 划分主题 /215
8.3.3 确定结构 /215
8.3.4 确定属性 /216
8.3.5 确定服务 /217
8.3.6 类图描述文档 /218
8.3.7 包图描述文档 /219
8.4 建立动态模型 /220
8.4.1 建立顺序图及其描述文档 /220
8.4.2 建立状态图及其描述文档 /221
8.4.3 建立协作图及其描述文档 /223
8.4.4 建立活动图及其描述文档 /224
8.5 小结 /226
习题 /226

第 9 章　面向对象设计　/228
9.1　面向对象设计概述　/228
9.1.1　面向对象分析与设计的关系　/228
9.1.2　面向对象设计原则　/229
9.2　精化类及类间关系　/230
9.2.1　设计类的属性　/230
9.2.2　设计类的方法　/231
9.2.3　设计类间泛化关系　/232
9.2.4　设计关联类　/234
9.3　数据设计　/234
9.3.1　基于关系数据库的数据设计　/234
9.3.2　基于其他方式的数据设计　/237
9.4　人机交互设计　/237
9.5　建立实现模型　/239
9.5.1　构件图及其描述文档　/239
9.5.2　配置图及其描述文档　/240
9.6　设计模式简介　/242
9.6.1　概述　/242
9.6.2　Singleton 模式　/243
9.6.3　Abstract Factory 模式　/244
9.6.4　Mediator 模式　/245
9.6.5　Adapter 模式　/247
9.7　面向对象的测试　/248
9.7.1　面向对象测试概述　/249
9.7.2　面向对象的单元测试　/249
9.8　小结　/251
习题　/252

第 10 章　软件维护　/254
10.1　软件维护概述　/254
10.1.1　软件维护的任务　/254
10.1.2　软件维护的特点　/255

10.1.3 软件维护的分类 /255
10.2 软件维护过程 /257
10.2.1 软件维护方式 /257
10.2.2 软件维护管理的基本内容 /258
10.2.3 维护中存在的问题 /263
10.2.4 维护活动记录 /264
10.3 软件的可维护性 /264
10.3.1 可维护性因素 /265
10.3.2 提高软件的可维护性 /265
10.4 逆向工程 /268
10.5 小结 /270
习题 /271

第 11 章 软件项目管理 /272
11.1 软件项目管理概述 /272
11.1.1 软件项目管理的特点和内容 /272
11.1.2 软件项目管理目标 /273
11.1.3 软件项目管理的 4P 观点 /274
11.2 软件项目规模度量 /275
11.2.1 代码行技术 /276
11.2.2 功能点计算 /277
11.2.3 代码行与功能点间的转换 /280
11.3 软件项目估算 /280
11.3.1 代码行和功能点的其他估算模型/281
11.3.2 专家估算模型 /281
11.3.3 Putnam 模型 /282
11.3.4 COCOMO 模型 /282
11.3.5 项目估算模型的小结 /285
11.4 项目进度管理 /285
11.4.1 项目进度控制 /285
11.4.2 甘特图 /286

11.4.3 工程网络图 /287
11.5 项目风险管理 /288
11.5.1 软件风险概念 /289
11.5.2 风险管理过程 /289
11.6 项目质量管理 /293
11.6.1 软件质量因素 /293
11.6.2 软件质量保证活动 /297
11.6.3 软件质量保证计划 /298
11.7 软件配置管理 /300
11.7.1 软件配置项 /300
11.7.2 配置管理过程 /301
11.7.3 软件配置管理计划 /304
11.8 项目人员组织管理 /305
11.8.1 团队组织 /305
11.8.2 团队组织方式 /306
11.9 软件能力成熟度模型 /308
11.9.1 基本概念 /308
11.9.2 软件能力成熟度模型等级 /309
11.9.3 关键过程域 /310
11.10 小结 /311
习题 /312

参考文献 /314

第 1 章　软件工程概述

随着全球信息化的不断发展、智能产品不断涌现、硬件技术不断提升而价格却不断下降，越来越多的工作、生产和生活用品都在使用各类应用系统，而这些系统都离不开软件系统强有力的支撑。随着应用的不断延伸、需求的不断增长，虽然软件研发能力不断增强，技术不断创新，但同时软件系统的规模和复杂度也在不断提高，特别是互联网将计算机和通信技术相融合，这对软件系统提出了更高、更新的要求。然而迄今为止，软件系统发展仍然没有彻底摆脱软件危机的困扰。

为了准确地描述用户需求、可控地管理软件开发、有效地进行软件维护，软件研究人员开始寻找消除软件危机的有效途径，并从 20 世纪 60 年代末起，逐渐形成了一门新兴的工程学科——软件工程。

1.1　软件工程的发展历程

自从 1946 年 2 月在美国出现第一台电子计算机以来，计算机技术发展十分迅猛，并且已由最初的国防和数学计算领域，逐渐转移到商用、家用等各领域。在计算机广泛应用的同时，软件系统越来越趋向大型化、复杂化和智能化，使得软件开发越来越复杂，而软件系统却越来越难以满足人们的需求。加之软件产品目前仍难以采用工业化方式批量开发，导致软件成本的居高不下。软件生产已经在各国占据重要位置，早在 20 世纪 60 年代末，当时的工业发达国家就已经意识到“软件危机”的危险性，即在软件开发成本急剧增长的同时，软件开发周期难以确定、开发过程难以控制、软件质量难以保证、软件维护难以为继。

1.1.1　软件危机

“什么是软件危机?”目前，大多数软件研究领域的人员都接受的定义是，软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。虽然随着软件管理和软件技术的不断进步，部分问题得到有效的改善，但软件危机仍然还困扰着软件和软件产业的发展。这些严重问题主要表现在以下几个方面。

(1) 软件开发进度和成本难以控制。由于用于项目估算的数据来自以往的统计值和经验数据，因而项目进度和成本的估算常常很不准确，同时也极大地降低甚至损害了软件开发人员和组织的声誉。

【案例 1.1】　伦敦股票交易系统当初预算 4.5 亿英镑，后来追加到 7.5 亿，历时 5 年，但最终还是失败了，导致伦敦股票市场声誉下跌。

【案例 1.2】　微软于 2001 年年底宣布启动 Vista 研发，最初预计在 2003 年完成该项

目。但在2005年年初和2006年3月分别推迟发布时间，直到在最初预计时间4年之后的2007年1月才正式发布该系统，耗时5年并投入了60亿美元。

(2) 软件产品难以满足用户的需求。用户对自己将要使用的软件系统的需求并不完全了解和掌握。同时，软件开发人员又常常在对用户需求不甚明了的前提下，匆匆着手开始程序设计和实现。软件开发人员之间、软件开发人员和用户之间的交流并不充分，信息掌握并不对等。盲目上马的项目必然导致最终的软件产品不符合用户实际的应用需求和操作习惯。

【案例1.3】 美国政府统计署(GAO)2000年的数据显示，美国军方每年花费数十亿美元购买软件，在其所购买的软件中，可以直接使用的只占2%，还有3%需要修改，而高达95%的软件都成为了垃圾。

(3) 软件质量难以得到保证。首先，难以给出客观的一致的软件质量评价体系。对于同一类型的软件，不同的专家和用户，站在不同的角度进行评审，结论难以达到统一。其次，软件可靠性和质量保证，在技术审查、管理复审、程序正确性证明和软件测试等方面亟待加强。

【案例1.4】 1963年美国的火箭控制系统程序，在该程序中，把FORTRAN语句"DO 5 I=1,3"写成了"DO 5 I=1.3"，结果使得发往火星的火箭爆炸，造成1000多万美元的损失。

【案例1.5】 1967年苏联"联盟一号"载人宇宙飞船在返航时，由于软件忽略了一个小数点，导致在进入大气层时因打不开降落伞而烧毁。

(4) 软件产品难以进行维护。软件产品在使用过程中发现的各类错误，并不是都能修改的，甚至某些错误是难以修改的。加之用户的需求不断变更，软件的运行环境不断改变，用户需要在原有软件系统中修改或增加一些新功能也已变得不可能。此外，不断攀升的软件维护费用也让用户不堪重负。

【案例1.6】 20世纪70年代软件维护费用占软件项目总预算的35%～40%，1980年—1990年上升到40%～60%，目前已经上升到70%～80%，且仍有上升趋势。

(5) 软件的文档资料难以管理。由于开发过程的不规范、需求的不确定性以及用户与开发人员缺少彼此沟通的桥梁——文档；因此，评价软件质量的相关文档资料的缺失，导致软件难以管理和维护，进而带来更严重的困难和问题。

(6) 软件产品的生产率难以得到提高。硬件系统的高速发展得益于标准的制订和工业化生产线。1965年Gordon Moore发现了摩尔定律：集成电路的性能每隔18个月提高一倍，而价格则下降一半。但这一定律并不适用于软件系统。软件系统的研发完全不同于硬件系统，因为软件系统本质上是一个智力活动，加之缺乏对主观意向的资料描述，因此更难以采用生产线的方式批量生产。

实际经验表明，在对计算机硬件不了解的情况下，普通人仍能组装一台计算机。对于计算机主板上的接口，如果各类板卡能插入某接口，通常说明该板卡的物理安装正确，否则就换另一个接口尝试。而对于软件系统而言，即使是相同应用领域的相同功能，仍难以采用类似硬件组装的方式"组装"软件，仍然要重新进行软件系统的开发过程，因而使得普通人难以介入到软件系统的开发中来。