

《UML 2.2面向对象分析与设计》

图书基本信息

书名：《UML 2.2面向对象分析与设计》

13位ISBN编号：9787302304241

10位ISBN编号：7302304246

出版时间：2013-1

出版社：清华大学出版社

作者：(英)班尼特(Bennett, S.),(英)麦克罗布(McRobb, S.),(英)法默(Farmer, R.)

页数：548

译者：李杨

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

《UML 2.2面向对象分析与设计》

内容概要

本书对前一版进行了修订，反映了信息系统开发中最前沿的方法。Bennett、McRobb和Farmer撰写的这本书是面向对象分析与设计领域的经典图书，是本科生和研究生“系统分析和设计”课程重要的教学用书。

本书为使用UML 2.2中的主要技术进行面向对象分析与设计给出了清晰实用的结构，遵循迭代和增量型方法(它们基于业内标准的统一过程)，将系统分析和设计置于整个系统生命周期的背景中。本书分为4部分：第1部分为信息系统的分析和设计以及面向对象提供了背景；第2部分关注需求获取和系统分析活动以及UML的基本标记法；第3部分涉及系统架构和设计活动，以及对象设计的UML标记法；第4部分介绍系统的实现，如何组织系统生命周期，以及如何开发可重用

《UML 2.2面向对象分析与设计》

作者简介

作者介绍：

本书的作者，Simon Bennett、Steve McRobb和Ray Farmer自从1999年本书第1版付梓的时候就一起工作。他们的合作汇集了各自在信息系统学科丰富的教学研究经验。他们丰富的业界经验也保证了在实际组织中UML知识实践应用的正确性。

Simon Bennett是De Montfort大学技术学院的培训顾问，他在这里提供UML、分析和设计，以及系统架构方面的培训。他是计算机智能中心的准会员。之前作为企业架构师供职于Celesio AG，作为ICT主管供职于Leicester议会的复兴和文化事务部，并且在1999年之前作为首席讲师任教于De Montfort大学。

Simon是Schaum's Outline of UML(2nd edition)一书的合著者，该书也由McGraw-Hill出版。

Steve McRobb是De Montfort大学信息学院的首席讲师。他教授面向对象系统分析与设计课程十几年，目前是“计算、信息技术和信息系统管理”研究生课程的课程负责人。他最近的研究范围主要集中在在线隐私以及ICT对权利关系的影响方面。Steve是Yorkshire Dales国家公园的前任首席行政主任。

Ray Farmer是Coventry大学工程和计算学院的副院长。他的研究兴趣包括信息系统分析与设计、面向服务架构以及工程和计算教育方面的教学研究。他定期作为英国和国际上面向对象分析与设计方面的咨询师。Ray之前在De Montfort大学的信息系统学院担任不同的职位。

书籍目录

案例A1 Agate公司案例研究——

简介

1

A1.1 Agate公司介绍

1

A1.2 现有计算机系统

2

A1.3 使用当前系统的业务活动

2

A1.4 需求总结

3

案例B1 FoodCo公司案例研究——

简介

5

B1.1 FoodCo公司介绍

5

B1.2 现如今的FoodCo公司

8

B1.3 建议

11

第1章 信息系统

13

1.1 介绍

13

1.2 信息系统的历史

14

1.3 信息系统现状

16

1.4 何为系统

17

1.4.1 系统思考

18

1.4.2 系统思考的益处

21

1.4.3 信息系统发展过程中的系统

26

1.5 信息与信息系统

27

1.5.1 信息

27

1.5.2 组织中信息系统的工作

27

1.5.3 信息技术

29

1.6 成功战略

30

1.6.1 确认商业战略	30
1.6.2 信息系统的贡献	31
1.6.3 信息系统和信息技术战略	31
1.7 本章小结	33
问题回顾	33
案例研究、练习和项目	33
拓展阅读	34
第2章 信息系统开发面临的挑战	35
2.1 介绍	35
2.2 挑战	36
2.2.1 终端用户的视角	37
2.2.2 客户的视角	39
2.2.3 开发者的视角	41
2.3 出错的原因	42
2.3.1 质量问题	43
2.3.2 生产率问题	45
2.4 道德层面	46
2.4.1 项目内的道德问题	47
2.4.2 更为广泛的道德问题	48
2.5 失败的代价	49
2.6 本章小结	50
问题回顾	50
案例研究、练习和项目	50
拓展阅读	51
第3章 面对挑战	

52	
3.1 介绍	52
3.2 问题响应	54
3.2.1 质量问题	54
3.2.2 安装和运行问题	55
3.2.3 生产率问题	56
3.3 项目的生命周期	57
3.3.1 瀑布型生命周期模型	58
3.3.2 原型	60
3.3.3 迭代和增量型开发	62
3.4 方法论	62
3.4.1 统一软件开发过程	63
3.4.2 敏捷方法	64
3.5 信息系统开发管理	65
3.6 用户参与	67
3.7 软件开发工具	68
3.7.1 模型和技术支持	68
3.7.2 软件构建	70
3.7.3 使用软件开发工具的益处和困难	71
3.8 本章小结	71
问题回顾	71
案例研究、练习和项目	72
拓展阅读	72
第4章 面向对象	73
4.1 介绍	

73	
4.2 基本概念	
73	
4.2.1 对象	
74	
4.2.2 类和对象	
75	
4.2.3 类成员	
76	
4.2.4 一般化和特殊化	
78	
4.2.5 封装、信息隐藏和消息传递	
82	
4.2.6 多态	
84	
4.2.7 对象状态	
85	
4.3 面向对象的起源	
86	
4.4 当前的面向对象语言	
88	
4.5 本章小结	
89	
问题回顾	
90	
案例研究、练习和项目	
90	
拓展阅读	
91	
第5章 建模的概念	
92	
5.1 介绍	
92	
5.2 模型和图	
92	
5.2.1 模型的含义	
93	
5.2.2 图	
94	
5.2.3 模型和图的区别	
96	
5.2.4 UML中的模型	
97	
5.2.5 开发模型	
98	
5.3 绘制活动图	
99	
5.3.1 活动图的目的	

99
5.3.2 活动图的标记法
100
5.4 开发过程
103
5.4.1 基本原则
104
5.4.2 主要活动
105
5.5 本章小结
109
问题回顾
109
案例研究、练习和项目
110
拓展阅读
110
第6章 需求获取
111
6.1 介绍
111
6.2 用户需求
111
6.2.1 当前系统
112
6.2.2 新的需求
113
6.3 事实发现技术
114
6.3.1 背景阅读
114
6.3.2 访谈
115
6.3.3 观察
117
6.3.4 文档抽样
118
6.3.5 调查问卷
119
6.3.6 记住上述技术
121
6.3.7 其他技术
122
6.4 用户参与
122
6.5 文档需求
123
6.6 用例
125

6.6.1 目的	125
6.6.2 标记法	126
6.6.3 使用原型支持用例	130
6.6.4 CASE工具支持	131
6.6.5 使用用例图的业务模型	131
6.6.6 测试和用例	132
6.7 需求获取和建模	132
6.8 本章小结	133
问题回顾	135
案例研究、练习和项目	135
拓展阅读	136
案例A2 Agate公司案例研究——需求模型	138
A2.1 介绍	138
A2.2 需求列表	138
A2.3 执行者和用例	139
A2.4 术语表	144
A2.5 最初架构	144
A2.6 需求建模活动	145
第7章 需求分析	147
7.1 介绍	147
7.2 分析模型	148
7.2.1 分析模型和其他模型的区别	148
7.2.2 好的分析	149

7.3 分析类图：概念和标记法

150

7.3.1 类和对象

150

7.3.2 特性

151

7.3.3 特性和状态

152

7.3.4 实例之间的链接

153

7.3.5 类之间的关联

154

7.3.6 关联和状态

154

7.3.7 多重性

155

7.3.8 操作

156

7.3.9 操作和状态

158

7.3.10 分析类图的稳定性

158

7.4 用例实现

159

7.5 绘制类图

161

7.5.1 健壮性分析

161

7.5.2 分析类stereotype

161

7.5.3 确认类

164

7.5.4 从通信图到类图

168

7.5.5 查找对象和类的其他

方法

169

7.5.6 添加和定位特性

173

7.5.7 添加关联

175

7.5.8 确定多重性

175

7.5.9 寻找操作

175

7.5.10 初始操作分配

175

7.6 CRC卡

176

7.7 整合分析类图	178
7.8 本章小结	179
问题回顾	179
案例研究、练习和项目	180
拓展阅读	183
案例A3 Agate公司案例研究——需求分析	184
A3.1 介绍	184
A3.2 用例实现	184
A3.3 整合分析类图	189
A3.4 需求分析活动	189
第8章 完善需求模型	191
8.1 介绍	191
8.2 软件和规范重用	192
8.2.1 为何重用	192
8.2.2 难以重用的原因	192
8.2.3 面向对象对重用的贡献	193
8.3 进一步向结构中添加内容	195
8.3.1 寻找和建模一般化	195
8.3.2 寻找和建模组合	199
8.3.3 组合或聚集与一般化的合并	200
8.3.4 组织分析模型——包和依赖关系	201
8.4 重用软件组件	202
8.4.1 组件的UML标记法	203

8.4.2 基于组件的开发	204
8.4.3 组件建模实例	205
8.5 软件开发模式	207
8.5.1 模式的起源	207
8.5.2 什么是软件模式	208
8.5.3 分析模式	209
8.6 本章小结	211
问题回顾	211
案例研究、练习和项目	211
拓展阅读	212
第9章 对象交互	213
9.1 介绍	213
9.2 对象交互和协作	214
9.3 交互顺序图	215
9.3.1 基本概念和标记法	216
9.3.2 管理顺序图	223
9.3.3 分支	226
9.3.4 延续	227
9.3.5 异步消息	227
9.3.6 时间约束	228
9.3.7 实时系统和并发的建模	229
9.3.8 准备顺序图的指导原则	230
9.4 通信图	230
9.4.1 基本概念和标记法	231
9.4.2 通信图中的信息标签	

232	
9.5 交互概览图	
234	
9.6 时序图	
236	
9.7 模型一致性	
237	
9.8 本章小结	
238	
问题回顾	
238	
案例研究、练习和项目	
238	
拓展阅读	
239	
第10章 规范对象操作	
240	
10.1 介绍	
240	
10.2 操作规范的角色	
241	
10.3 合同	
242	
10.4 描述操作逻辑	
243	
10.4.1 非算法型方法	
243	
10.4.2 算法型方法	
246	
10.5 对象约束语言	
251	
10.6 创建操作规范	
253	
10.7 本章小结	
255	
问题回顾	
255	
案例研究、练习和项目	
256	
拓展阅读	
256	
第11章 规范控制	
257	
11.1 介绍	
257	
11.2 状态和事件	
258	
11.3 基本标记法	
259	

11.4 深度标记法	264
11.4.1 组合状态	264
11.4.2 并发状态	265
11.4.3 进入和退出伪状态	267
11.4.4 交叉和选择伪状态	268
11.4.5 历史伪状态	268
11.4.6 状态机的特殊化	269
11.5 准备状态机	269
11.5.1 行为型方法	270
11.5.2 生命周期方法	273
11.6 协议型和行为型状态机	273
11.7 一致性检查	274
11.8 质量准则	274
11.9 本章小结	275
问题回顾	275
案例研究、练习和项目	276
拓展阅读	276
案例A4 Agate公司案例研究—— 深入分析	277
A4.1 介绍	277
A4.2 顺序图	277
A4.3 状态机	278
A4.4 操作规范	280
A4.5 对类图的进一步修订	281
A4.6 需求分析的深入活动	283

第12章 设计	285
12.1 介绍	285
12.2 设计和分析有何不同	285
12.2.1 设计	286
12.2.2 迭代型生命周期中的设计	287
12.3 逻辑设计和物理设计	287
12.4 系统设计和详细设计	289
12.4.1 系统设计	290
12.4.2 详细设计	290
12.5 设计的质量和目标	290
12.5.1 目标和约束	291
12.5.2 设计折中	294
12.5.3 设计中的可衡量目标	295
12.6 本章小结	296
问题回顾	296
案例研究、练习和项目	297
拓展阅读	297
第13章 系统架构	298
13.1 介绍	298
13.2 架构的含义	299
13.3 生成架构模型的原因	301
13.4 对系统架构的影响	302
13.4.1 现有系统	303
13.4.2 企业级架构	304

13.4.3 技术参考架构	305
13.5 架构风格	305
13.5.1 子系统	306
13.5.2 分层和分区	307
13.5.3 MVC架构	311
13.5.4 分布式系统架构	314
13.5.5 架构和开发的组织结构	316
13.6 并发	317
13.7 处理器分配	318
13.8 系统设计标准	319
13.8.1 数据库设计	319
13.8.2 用户界面设计标准	319
13.8.3 构建准则	319
13.9 Agate软件架构	320
13.10 本章小结	321
问题回顾	322
案例研究、练习和项目	322
拓展阅读	322
第14章 详细设计	324
14.1 介绍	324
14.2 在面向对象的详细设计中添加的内容	325
14.3 特性和操作规范	325
14.3.1 特性的数据类型	325
14.3.2 派生特性	

326	
14.3.3	主操作
327	
14.3.4	操作签名
328	
14.3.5	可见性
329	
14.4	将特性和操作归入类中
331	
14.4.1	分配职责
331	
14.4.2	接口
332	
14.4.3	耦合和内聚
333	
14.4.4	Liskov替换原则
336	
14.5	设计关联
336	
14.5.1	一对一关联
337	
14.5.2	一对多关联
338	
14.5.3	多对多关联
340	
14.5.4	保持类最小
341	
14.6	完整性约束
342	
14.6.1	引用完整性
342	
14.6.2	依赖约束
343	
14.6.3	域完整性
344	
14.7	设计操作算法
344	
14.8	本章小结
345	
	问题回顾
345	
	案例研究、练习和项目
346	
	拓展阅读
346	
	第15章 设计模式
347	
	15.1 介绍
347	

15.2 软件开发模式	348
15.2.1 框架	348
15.2.2 模式目录和语言	348
15.2.3 软件开发原则和模式	348
15.2.4 模式和非功能性需求	349
15.3 文档模式——模式模板	349
15.3.1 模板内容	349
15.3.2 模板的其他方面	350
15.4 设计模式	350
15.4.1 设计模式的类型	350
15.4.2 创建型模式	351
15.4.3 结构型模式	355
15.4.4 行为型模式	359
15.5 如何使用设计模式	363
15.6 使用模式的优缺点	364
15.7 本章小结	365
问题回顾	365
案例研究、练习和项目	365
拓展阅读	365
第16章 人机交互	367
16.1 介绍	367
16.2 用户界面	368
16.2.1 什么是用户界面	368
16.2.2 对话隐喻	368
16.2.3 直接操纵隐喻	

370	
16.2.4	良好对话的特征
372	
16.2.5	风格指导
374	
16.3	用户界面设计方法
375	
16.3.1	非正式方法和正式方法
375	
16.3.2	实现可使用性
381	
16.4	标准和合法需求
382	
16.5	本章小结
383	
	问题回顾
384	
	案例研究、练习和项目
384	
	拓展阅读
384	
	第17章 设计边界类
386	
17.1	介绍
386	
17.2	表示层结构
387	
17.3	为用户界面建立原型
388	
17.4	设计类
390	
17.5	使用顺序图设计交互
393	
17.6	类图回顾
399	
17.7	用户界面设计模式
400	
17.8	使用状态机建模界面
402	
17.9	本章小结
408	
	问题回顾
409	
	案例研究、练习和项目
409	
	拓展阅读
409	
	第18章 数据管理设计

411	
18.1 介绍	411
18.2 持久性	412
18.2.1 持久性需求	412
18.2.2 存储机制概述	412
18.2.3 持久性架构	413
18.3 文件系统	415
18.3.1 文件和记录结构	415
18.3.2 文件的组织方式	416
18.3.3 文件访问	416
18.3.4 文件类型	419
18.3.5 文件使用示例	420
18.4 数据库管理系统	421
18.4.1 文件和数据库	421
18.4.2 数据库的类型	423
18.5 设计关系型数据库管理系统	425
18.5.1 关系型数据库	425
18.5.2 数据建模和规范化	426
18.5.3 将类映射为表	430
18.6 设计对象数据库管理系统	432
18.7 分布式数据库	436
18.8 数据管理类设计	438
18.8.1 分层结构	438
18.8.2 PersistentObject超类	439
18.8.3 数据库代管者框架	

441	
18.8.4	使用数据管理产品 或框架
445	
18.9	本章小结
447	
	问题回顾
447	
	案例研究、练习和项目
447	
	拓展阅读
448	
	案例A5 Agate公司案例——简介
449	
A5.1	介绍
449	
A5.2	架构
449	
A5.3	用例示例
450	
A5.4	类图
451	
A5.5	顺序图
455	
A5.6	数据库设计
458	
A5.7	状态机
459	
A5.8	设计活动
460	
	第19章 实现
461	
19.1	介绍
461	
19.2	软件实现
462	
19.2.1	软件工具
462	
19.2.2	编程和文档标准
464	
19.3	组件图
465	
19.4	部署图
467	
19.5	软件测试
469	
19.5.1	由谁执行测试
469	
19.5.2	测试内容

470	
19.5.3	测试文档
472	
19.6	数据转换
473	
19.7	用户文档和培训
474	
19.7.1	用户手册
474	
19.7.2	用户培训
474	
19.8	实现策略
475	
19.9	评审和维护
476	
19.9.1	下一步骤
476	
19.9.2	评审过程和评估报告
477	
19.9.3	维护活动
478	
19.10	本章小结
480	
	问题回顾
480	
	案例研究、练习和项目
481	
	拓展阅读
481	
	第20章 软件重用
482	
20.1	介绍
482	
20.2	为什么要进行重用
482	
20.2.1	项目的选择
484	
20.2.2	组织的结构
485	
20.2.3	重用合适的单元
486	
20.2.4	组件标准
487	
20.3	为重用规划策略
489	
20.3.1	选择视角法
489	
20.3.2	RSEB
490	

20.4 商用组件	491
20.5 案例研究示例	493
20.6 Web服务	499
20.7 本章小结	501
问题回顾	501
案例研究、练习和项目	501
拓展阅读	502
第21章 软件开发过程	503
21.1 介绍	503
21.2 过程、方法和方法论	504
21.2.1 方法论	504
21.2.2 为何使用方法论	506
21.3 统一软件开发过程	506
21.4 动态系统开发方法	510
21.4.1 Atern生命周期	511
21.4.2 DSDM Atern技术	513
21.4.3 DSDM中的项目选择	514
21.5 Scrum	514
21.6 极限编程	516
21.7 选择方法论时的考虑	517
21.8 硬性与软性方法论	519
21.9 本章小结	521
问题回顾	521
案例研究、练习和项目	522
拓展阅读	

522

附录A 标记法汇总

523

附录B 部分解决方案和答案指导

531

术语表

540

章节摘录

版权页：插图：本书的作者在购买一双鞋子的时候，店员为正确录入商品价格而费尽力气。会有新的现金金额录入系统，但是因为这双鞋子是促销产品，所以录入起来很困难。顾客购买这种鞋子，并且免费附赠一双袜子。因为袜子也是常规商品，所以需要袜子也进行记录。这意味着袜子通过促销被“卖出”，尽管是免费的。处理这种问题的一个简单方法就是在出售时由助理人员将价格置零。助理人员会尝试这样做，但是系统可能会特别阻止商品以零售价卖出。助理求助于经理。在经过一些类似情况后，会发现解决这种交易的唯一方法就是把鞋子的价格降低1便士，而袜子以1便士的价格售出，因此总的售价是正确的。既然店员明白如何做，那么在以后类似的情况下，他们处理起来也就简单多了。但是处理一项经常出现的任务是很繁琐的。这种糟糕的设计还有很多例子，它们给用户带来了不少不快，并且浪费了时间。“这个系统看起来很美——但是能用来完成一点有用的事情吗？”系统可能设计得美观，易于使用，但是仍然不能处理“正确的”事情，这对于应该由系统执行的任务来说是一个问题。例如图书馆分类查询系统，如果只有在书的名称和作者姓名都拼写无误的情况下才能检索出书号，那么这一系统的功能就受限。读者在查询的时候很可能不知道书的名称，即便知道作者的姓名，也可能拼写错误。系统失败的另一个例子是，虽然满足了用户的需求，但是系统性能很差（这一点与刚才系统可用性的问题重复）。系统对用户来说，作用可能值得怀疑，因为系统要求他们按照看起来毫无意义的方式工作。用于佐证的例子虽然很古老，但是依旧很有价值，因为作者可以清晰地描述出来。设计仓库管理系统的部分原因是，提高管理员对仓库中紧张空间的使用控制。工人发现新的系统为了以最佳的方式最大化存储空间的利用率，但这却剥夺了他们的自由决定权：因为他们会看到，节省这些空间能为公司带来改善，他们找到了使用系统的方法。他们因为态度不好而饱受管理人员的责难，而且作为部门的主要职员，工人被自认为，不必要的废文和程序而烦恼，但是使用它们却又是他们对公司的责任。如果软件错误和失败危及生命，那么将更加令人忧心。一个极端的例子是伦敦救护车服务计算机辅助调度（London Ambulance Service Computer Aided Dispatch, LASCAD）系统，该系统在1992年发布之后不久就被弃置。估计总的开发成本为4300万英镑。该系统本来是为了加快为紧急情况派出救护车的流程，但实际上却降低了响应时间。在出现几次因为病人长时间等待而导致死亡的情况之后，该系统被弃置了。虽然这些声明从未被证实过，但是继续运行该系统的风险是不可接受的。关于是否是因为软件错误而导致英国皇家空军Chinook直升机在Kintyre半岛的Mull坠机的争论，至今尚未停息。机上所有的29人全部丧生，包括高级警官和军方情报长官。英国政府能接受的定论是飞行员的疏忽大意。然而，事故调查组的结论是机组人员可能因主要的技术（例如软件）故障而分心。Computer Weekly的一系列报道以及British TV的4频道新闻，宣称英国国防部内部报告已经对此类型直升机在特定情况下引擎控制软件的稳定性产生了顾虑。英国上议院选举委员会建议驳回国防部的观点。在最近，2007年6月份，又有新的证据可能推翻官方结论（Collins, 2007），但是至今政府拒绝重启调查，而将责任继续归咎于飞行员。

《UML 2.2面向对象分析与设计》

编辑推荐

《国外计算机科学经典教材:UML 2.2面向对象分析与设计(第4版)》对前一版进行了修订,反映了信息系统开发中最前沿的方法。Bennett、McRobb和Farmer撰写的这本《国外计算机科学经典教材:UML 2.2面向对象分析与设计(第4版)》是面向对象分析与设计领域的经典图书,是本科生和研究生“系统分析和设计”课程重要的教学用书。

《UML 2.2面向对象分析与设计》

精彩短评

- 1、翻译教材最怕词不达意，或者语言别扭，这点本书还算不错；但是书中随便出现的英文缩写让人头疼，文中也不给出解释，突然出现，书后附录也没有词典，让人看了莫名；原书是有网络资料的，但是按照原文给的网站发现只提供服务给欧洲用户，这个出版社按道理应该解决一下。
- 2、还没细看，这对我来说是个完全陌生的领域，等看完以后再评论吧。不过我收到的书封面和纸还可以。
- 3、uml的具体用法讲的不多，只是uml在软件过程中的运用，软件过程讲得很多

1、学习目标 系统分析和设计的含义本书是关于系统分析和设计的一本书，但是这些术语是什么意思呢？很多人都熟悉计算机编程的思想。他们了解，要使计算机执行某一复杂的任务，而这一任务运行着一项业务，就不得不编写出一系列指令来明确规范计算机应该完成的事情。这些指令以诸如C++、Java或C#之类的编程语言编写，形成了我们所熟知的计算机软件。然而，很少有人知道程序员在开始编写代码之前事先需要完成的工作。程序员不能简单地制定业务操作的规则，或是猜测需要输入系统中的数据类型，这些数据类型会被存储并且稍后会被访问，以屏幕显示或报告的形式提供给用户。系统分析师的工作是：调查业务的工作方式；理解和归档现有系统，而不管系统是手动系统还是计算机化的系统；记录业务需要操作的数据类型，并且记录确定这些数据处理方式的规则。系统分析师生成新系统必须要实现的规范。规范会明确无误地定义系统如下方面中的大多数： 需要保存在系统中的数据的特征。 用于确保数据正确输入的规则。 描述数据如何处理的规则。 用于数据输入、查询和报告的窗口的布局和内容。 用于确保只有授权用户才能访问数据的规则。 系统预期性能，例如系统处理的数据量，以及系统必须以多快速度处理信息请求。对于上述工作中的一部分，在早期是通过称为业务分析师的专业人员完成的，他们在了解组织期望的系统工作方式，以及按照系统分析师可以使用的方式归档这些需求方面有所专长。我们通常在大部分情况下，将业务分析师的工作置于更为广义的系统分析分类中。假定一份规范可以组建计算机系统，不过计算机系统还是有很多方式可以组建的。如果为一名工程师提供一份规范，说明需要穿过一条河流，他会选择建立摆渡系统、隧道或桥梁。如果桥梁是解决方案，那么还有很多可能的方式用来辅助修建桥面。设计桥梁的工程师可以选择吊桥的方式、支撑桥面的桥墩的数目、桥面多车道的宽度、是否包含自行车道或人行道。会有成百上千个决定需要作出，从非常宏观的方面到为行人设计扶手之类的详细设计。系统设计师在设计新的计算机系统方面扮演类似的角色。给定系统必须完成的任务的规范，对于系统如何执行所需要的进程，会有诸多不同的可能性。系统设计师的角色，是从设计应该最好地满足用户以及其他系统成功的利益相关者(例如使用系统的组织管理层)的设计的广泛可能性中选择。在进行最优设计的选择过程中，系统设计者不仅会考虑系统必须完成的规范，也会考虑对系统处理数据以及必须响应的速度等方面的所有期望。根据系统分析师生成的规范，生成新系统如何工作的规范。该规范会明确无误地定义系统的如下方面： 作为整体，系统被组织为模块或子系统的方式，这些部分互相交互的方式，以及它们被置于不同处理器和计算机中的方式。 编程语言和用于构造软件的现有软件组件。 在面向对象系统中，待发布的系统功能以及类的规范，这些规范用于在程序运行期间保存数据。 存储数据的数据库的结构，以及需要保存在系统中的数据的特征。 操作数据，并且满足系统性能需求的算法的详细逻辑。 用于数据输入、查询和报告的窗口的物理外观和体验，包括配色方案、字体以及待使用的各种界面控件的精确类型——文本框、单选按钮等。 安全子系统控制访问系统的方式。 系统如何满足数据量的需求、数据处理速率的需求以及用户请求被响应的需求。程序员之后会采用这一设计，并且将之转换为程序代码，进而成为有用的信息系统。选择使用面向对象编程语言(参见第4章对于面向对象的解释)是在开发新系统的项目实施早期需要作出的决定。如果使用一种面向对象语言，就能将分析和设计按照一定方式转换为面向对象的程序代码。统一建模语言(UML)是归档分析和设计活动输出的一种方式，这些活动使得程序员更加容易地将设计转换为代码。UML是生成模型(用于归档分析和设计)的标准标记。有了这一标准标记，更易于不同团队之间进行交流。当然在现实世界中，上述内容都没有明确的界限。您会发现工作头衔为分析师/程序员的人，他们与用户讨论，分析和归档用户需求，设计解决方案，编写代码。在一些项目中，管理层会要求所有的分析在设计开始之前完成。而在其他项目中，只要大部分的关键分析材料可以使用，设计者就可以开始工作，程序员甚至可能开始生成软件的最初版本，虽然此时需求仍在制定当中。第2章描述了在开发信息系统时面临的一些挑战。正如您将会看到的，这并不是一个简单的过程。针对如何增加组织介入信息系统开发过程的可能性，使系统得到期望拥有的功能，有很多观点。一些人将此作为工程问题，认为系统的开发应该与桥梁、汽车或飞机之类的工程产品的开发类似。还有人考虑到了系统中的人为因素(使用系统的人群)，这些因素使开发系统的工作具有挑战性，而系统分析和设计应该更多地考虑系统的社会-技术背景。这两种观点都有可取之处。在本书中，我们解释了系统是什么，并且描述了开发信息系统的挑战，以及人们建议解决这些挑战的一些方法。我们之后会介绍开发系统的生命周期，从业务分析到设计，最后介绍在实现中解决的一些问题，以及为组织这一过程而提出的一些方法。信息系统存在于现实世界的商业和

《UML 2.2面向对象分析与设计》

其他组织中。为了演示真实系统的开发，我们使用了两个公司的案例研究——一个是为了本书示例，另一个是为了读者练习。在介绍完这两个案例研究之后，我们开始在第1章使用读者熟悉的一些实例来介绍信息系统的理论知识。

《UML 2.2面向对象分析与设计》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu000.com