《UML和模式应用（原书第2版）》

图书基本信息

书名：《UML和模式应用（原书第2版）》

13位ISBN编号：9787111145189

10位ISBN编号：7111145186

出版时间：2004-8-1

出版社：机械工业出版社

作者：拉尔曼

页数：488

译者：方梁

《UML和模式应用（原书第2版）》

内容概要

本书英文版面世以来，便广受业界专家和读者的好评。全书叙述清晰、用词精炼、构思巧妙，将面向对象分析设计的概念、过程、方法、原则和个人的实践建议一一道来，博引多家观点，以实例为证，将软件的分析和设计的过程叙述得如逻辑推理一般，于细节处见真知。

本书是一本经典的面向对象分析设计技术的入门书，适用范围非常广，从刚刚入门的初学者，到已经有一定对象技术知识但希望进一步提高开发水平的中级读者，甚至是资深的专业人员，都可以从本书获益匪浅。同时，本书也很适合作为高等院校计算机专业及软件学院相关课程的教材和各类培训班的辅导教材。

《UML和模式应用（原书第2版）》

作者简介

Craig Larman 是国际著名的对象技术专家，曾提出著名的OO设计CRAS和PV原则，擅长于对象技术、OOA/D、模式、UML、敏捷建模，统一过程敏捷化、UP与XP及Scrum方法结合、迭代的敏捷开发方法等领域。

《UML和模式应用（原书第2版）》

书籍目录

《UML和模式应用（原书第2版）》

精彩短评

1、读过第一版，再读第二版。内容有了大幅的改变
2、一本不错的UML模式应用书籍，包含很多重要的分析方法，作者发明的通用责任分析模式是跟GOF的模式不同的分析设计模式，值得借鉴
3、UML和模式的应用
4、翻译的不太好，但绝对是好书
5、很不错的一本书，我看的是影印版，感觉非常好。对于了解软件开发、UML、设计模式的应用有好处，但是在真正的实践中还需要有选择的使用！
6、04
7、这本书不错.深入浅出.老师还拿来当教材呢
8、很多年前看过，当时还是第二版，OOA/D非常好的入门书，其中描述的设计原则在今天看来依旧没有过时
9、作者按照迭代的模式，从OOA/OOD的概念到系统架构中模式的应用，一层一层的展开。介绍了如何从最初的软件需求，结合UML帮助软件开发、设计人员更好地进行领域模型的表达和建立，以及随着迭代的进行，逐步细化设计，最终完成一个有弹性、易维护的软件产品。
10、模式应用
11、不错，很好的入门书籍

《UML和模式应用（原书第2版）》

精彩书评

1、我在多年前买过此书的第二版，非常经典，当时在工作当中对面向对象的概念比较模糊，不知道怎么和工作结合起来，当时面向对象的流派也非常多，看了此书后对OO的概念顿时清晰了很多，每看一章都要从板凳上跳起来，我想世界上每一题材的书多非常多，但是能写到人心里面去的屈指可数，强烈推荐！！

《UML和模式应用（原书第2版）》

章节试读

1、《UML和模式应用（原书第2版）》的笔记-全书

分析和设计可概括为：做正确的事（分析）和正确地做事（设计）

UP与XP之间两个值得注意的区别：
1. UP建议增量编写用例和非功能性需求文档。XP不是
2. UP建议在迭代开始，主要编程之前绘制更多的可视化设计图（例如耗时半天或一天）。XP建议用一点点时间来做可视化设计。

初始阶段：

目标：预见项目范围、构想和业务案例

需要解决的主要问题：
* 项目相关人员对项目构想达成基本一致
* 项目是否值得继续认真研究

需求：
* 用例是一种用于发现和记录需求（尤其是功能性需求）的机制
* 用例模型是所有用例的集合，是系统功能和环境的模型。

简洁用例+临时用例 ==〉详述用例

需求分析时合理的用例：
计算机应用进行需求分析时，应专注于基本业务过程（EBD）级别的用例

构想和补充规范：用来帮助识别其他需求。

细化阶段：

目标：
* 根据初始阶段的需求分析，建立领域模型和设计模型并在迭代中不断细化、完善
* 建立核心架构，解决高风险元素，定义大多数元素，评估总体进度和资源。

细化阶断需要几个迭代，不断细化/完善设计，按照需求优先级完善系统功能实现。

用例模型：
* 通过SSD（系统顺序图）描述用例主要成功场景 --〉用来识别系统事件细节和边界。
* 通过操作契约描述 --〉添加细节，同时更新领域模型

领域模型：
* 语言分析（问题域语言描述，场景描述）--〉识别概念类（名词及名词短语）
* 语义分析 --〉 识别概念类间关联
* 场景描述等 --〉 识别、添加属性

设计模型 -- 对象设计：

* GRASP （General Responsibility Assignment Software Patterns）: 根据职责设计对象。从一个对象行为角度看，职责与对象义务相关联。职责分两种：
    * 了解型 （Knowing）
    * 行为型 （Doing）

* GRASP
    * 信息专家
    * 控制器
    * 低耦合
    * 高内聚
    * 多态
    * 纯虚构：设计的类，不代表任何问题域中概念，只是为了达到支持高内聚、低耦合和重用的目的
    * 中介：解决直接耦合问题
    * 受保护变化：找出预计有变化或不稳定的元素，为其创建稳定“ 接口 ”而分配职责。

* 应用GoF模式对系统进一步分析设计建模

设计模型 -- 后续：随着迭代及持续，逐步细化系统整体设计
* 建立用例间关系 --〉泛化建模
* 状态图：为状态行为建模 （状态建模）

优先使用状态图说明外部和临时事件以及对事件的反应，而不是使用它来描述基于内部事件的对象行为。

* 架构模式
    * 模式的种类：
        * 架构模式：大尺度和粗粒度的设计。
        * 设计模式：中小尺度的对象和框架设计。
        * 习惯用法：面向语言或实现的低层次设计解决方案。

    * 层架构模式：
        * 将系统逻辑结构组织到不同层中，每层具有独立和相关的职责，较低层为低级和通用的服务，较高层更多为特定应用。
        * 从较高层到较低层进行协作和耦合，避免从低层到高层的耦合。
        * 层模式是几个核心架构模式之一，但并不能适用于所有问题。其它还有管道和过滤器模式。
        * 有些情况下，层增加会带来性能问题。

* 包设计和组织原则：
    * 按功能性内聚垂直和水平划分组织成包
    * 将一族接口打包
    * 基于开发工作和不稳定的类簇打包
    * 职责最大包应最稳定
    * 提取出独立元素
    * 使用工厂方法减少对具化包的依赖
    * 避免包之间循环依赖

* 框架设计 （特别是持久化框架设计）

架构分析：

* 架构因素：辨别和分析影响架构的非功能性需求
* 架构决策：对架构有重要影响的需求，分析可选方案并作出决定

架构视图：

* 逻辑视图：层、子系统、包、框架、类和接口的说明
* 进程视图：
* 部署视图：进程和组件到处理节点的物理部署和节点之间的物理网络配置
* 数据视图：对持久化数据的定义描述。包括对象到数据库映射规格
* 用例视图
* 实现视图：可交付代码

持久化框架：

* 功能
  * 在持久化框架存储器中存储和检索对象
  * 提交和回滚事务

* 关键思想
  * 映射（mapping）
  * 对象标识（Object identify）
  * 数据库映射器（database mapper）
  * 物化和反物化（materialization and dematerialization）
  * 缓存（caches）
  * 对象的事务状态（transaction state of object）
  * 事务操作（transaction operation）
  * 延迟物化（lazy materialization）
  * 虚拟代理（virtual proxy）

构造阶段：
迭代实现遗留的风险较低和比较容易的元素，准备部署。

《UML和模式应用（原书第2版）》

版权说明