

敏捷技能修炼



[敏捷技能修炼_下载链接1](#)

著者:Alan Shalloway

出版者:机械工业出版社华章公司

出版时间:2012-9-5

装帧:平装

isbn:9787111395270

《敏捷技能修炼：敏捷软件开发与设计的最佳实践》的4位作者都是世界顶级的软件开发专家和敏捷导师，都有数十年的软件行业从业经验，其中3位曾荣获Jolt大奖。本书是敏捷软件开发领域公认的经典著作，权威性毋庸置疑。

书中内容围绕“敏捷式编程”这一主题展开，对每一位敏捷软件开发人员都应该掌握的核心技能和技术进行了深入阐述，总结出了大量最佳实践，提供了一整套最精炼的技术集合，可以帮助他们在开发中变得游刃有余，极大地提高开发效率和软件质量。

《敏捷技能修炼：敏捷软件开发与设计的最佳实践》共分四个部分：第一部分（1~7章），阐述了在软件开发过程中能起到“四两拨千斤”作用的几种思想方法（“小舵板”），如意图导向编程、分离构造和使用、测试先行和Shalloway原则等，并总结了业界常用的几种实践，包括如何封装、面向接口的设计和验收测试驱动等；第二部分（8~9章），对过度设计和持续集成这两个问题进行了深入的探讨，并给出了最佳实践；第三部分（10~13章），作者分享了很多只有在他们的教学现场才能获得的经验，这些经验是优秀架构师应该具备的，具体包括共性和可变性分析、以开放关闭原则为目标的重构、需求与功能接口、何时以及如何使用继承等重要内容；第四部分是附录，介绍了统一建模语言、提高代码质量的原则，以及如何封装原始数据类型等。

作者简介:

Alan Shalloway, Net

Objectives公司创始人及CEO。Alan有40多年工作经验，他是计算机软件行业，特别是精益、看板、产品系列管理、scrum和敏捷设计方面的思想领导者。他帮助公司在企业级层面向精益和敏捷方式转型，同时教授员工相关课程。Alan开发了关于精益—敏捷的培训辅导方法，这帮助Net

Objectives公司的客户取得了长期的可持续的生产力。他经常活跃于全球范围的高端峰会并发表精彩演讲。他还是《Design Patterns Explained: A New Perspective on Object-Oriented Design》（Jolt获奖作品）和《Lean-Agile Pocket Guide for Scrum Teams》的主要作者。在他的职业生涯中，Alan从事过多个行业。他是精益软件和系统协会（the Lean Software and Systems Consortium）的共同创始人及董事会成员。他拥有麻省理工学院计算机科学系的硕士学位和艾莫利大学（Emory University）数学系的硕士学位。更多详细信息请查看Twitter上的@alshalloway。

Scott

Bain，在计算机技术方面有将近40年经验，从事过软件开发、软件工程、框架设计等方面的工作。Scott也曾经从事课堂和远程教学等教育活动，包括课程的设计、实施培训和相关管理，给用户提供认证培训和终端用户培训。目前Scott在敏捷分析和设计模式、高级软件设计和可持续的测试驱动开发等方面进行授课和提供咨询。同时，Scott还经常在JavaOne和SDWest这样的开发者研讨会上进行精彩的演讲。他是《Emergent Design: The Evolutionary Nature of Professional Software Development》的作者，此书荣获了Jolt生产力奖。

Ken Pugh, Net

Objectives公司高级咨询师。凭借多年的丰富经验，他提供培训和辅导，帮助公司向精益—敏捷转型。他热衷于研究沟通（特别是有效地传递需求）、业务价值交付，以及用精益原则来快速进行高质量的交付。同时，在技术课题方面，他提供从面向对象设计到Linux/Unix等多方面的培训和指导。此外，他还写过好几本编程书籍，包括获得2006年Jolt大奖的《Prefactoring: Extreme Abstraction, Extreme Separation, Extreme Readability》。最近的一本书是《Lean-Agile Acceptance Test Driven Development: Better Software Through Collaboration》。他的客户遍及伦敦、波士顿、悉尼、北京和海得拉巴。工作之余，他喜欢滑雪、帆船、自行车和阿巴拉契亚徒步登山活动。

Amir Kolsky, Net

Objectives的资深咨询师、教练和培训师。Amir从事计算机科学领域已经超过25年了。他在IBM研究院工作过10年，此外有9年时间在各种大小类型的公司担任过首席架构师

和首席技术官等职位。他从2000年开始接触敏捷开发。他先后创建了MobileSpear以及XPand软件公司，专门在以色列和欧洲提供敏捷辅导、软件教育和敏捷项目实施。目前Amir把他的专业经验带到了Net Objectivs，作为敏捷教练和讲师，提供关于精益和敏捷软件流程、工具和实践、Scrum、极限编程、设计模式以及测试驱动开发方面的培训。

目录: 推荐序

译者序

丛书前言

前言

致谢

第一部分 最关键的小舵板

第1章 意图导向编程

1.1 意图导向编程：一个实例

1.2 优点

1.2.1 方法的内聚性

1.2.2 可读性和表达性

1.2.3 调试

1.2.4 重构和增强

1.2.5 单元测试

1.2.6 更易修改和扩展

1.2.7 在代码中发现模式

1.2.8 可迁移的方法

1.3 小结

第2章 分离构造和使用

2.1 一个问题

2.2 两种视图

2.2.1 创建视图

2.2.2 使用视图

2.2.3 隐藏的部分 更容易改动

2.2.4 现实的做法

2.2.5 一些实际的考量因素

2.3 给你的决策计时

2.4 重载和C++

2.5 自我查验

2.6 小结

第3章 代码未动，测试先行

3.1 一个小舵板：测试与可测试性

3.2 什么是测试

3.3 可测试性和代码质量

3.4 案例学习：可测试性

3.4.1 随时应对变化

3.4.2 青蛙一样的程序员

3.5 一个关于测试先行的思考

3.5.1 更好的设计

3.5.2 更清晰的范围和避免不必要的工作

3.5.3 降低复杂性

3.5.4 其他优势

3.5.5 没有例外

3.6 小结

第4章 Shalloway法则和Shalloway原则

4.1 冗余的种类

4.1.1 复制和粘贴

- 4.1.2 “魔法” 数字
- 4.1.3 其他类型
- 4.2 重新定义冗余
- 4.3 其他形式的冗余
- 4.4 设计模式在减少冗余时扮演的角色
- 4.5 很少有开发人员花费大量的时间去 “修改” 代码错误
- 4.6 冗余对代码质量其他方面的影响
- 4.7 小结

第5章 封装

- 5.1 未封装的代码：对全局变量的破坏
- 5.2 成员标志的封装
- 5.3 自封装成员
- 5.4 预防代码更改
- 5.5 封装引用对象的难点
- 5.6 用get() 来打破封装
- 5.7 对象类型的封装
- 5.8 设计的封装
- 5.9 各个层次的封装
- 5.10 实用性建议：把困难封装起来
- 5.11 小结

第6章 面向接口的设计

- 6.1 针对接口的设计
- 6.2 接口的定义
- 6.3 接口约定
- 6.4 分离不同的视图
- 6.5 接口的模拟实现
- 6.6 让接口保持简单
- 6.7 避免过早采用继承体系
- 6.8 接口和抽象类
- 6.9 依赖反转原则
- 6.10 多态性概述
- 6.11 不是每个类都需要接口
- 6.12 小结

第7章 验收测试驱动开发

- 7.1 两种开发流程
- 7.2 验收测试
- 7.3 一个关于验收测试的实例
- 7.4 实现验收测试
 - 7.4.1 针对用户界面的测试脚本
 - 7.4.2 测试用户界面
 - 7.4.3 XUnit测试
 - 7.4.4 验收测试框架
 - 7.4.5 四种方法间的联系
- 7.5 一个练习
- 7.6 如果客户不告诉你怎么做的时候，你应该怎么办
- 7.7 小结

第二部分 基本态度

第8章 避免过度设计或设计不足

- 8.1 给开发人员的箴言
- 8.2 代码质量病理学
- 8.3 避免过度设计或设计不足
- 8.4 把复杂度和返工最小化
- 8.5 永不把代码变得更糟/仅在有目的的情况下降低代码质量
- 8.6 使代码容易修改，足够强大健壮，适应变化并安全可靠

8.7在非面向对象的代码或遗留系统里编写易于修改代码的策略

8.8小结

第9章 持续集成

9.1建立源代码分支

9.1.1多版本：特殊分支

9.1.2孤立地工作：开发分支

9.1.3问题、解决方案、新的问题

9.2将主干内容合并回分支

9.3测试驱动开发与合并成本

9.4持续集成

9.5持续集成服务器

9.6小结

第三部分 设计问题

第10章 共性和可变性分析

10.1用动词和名词来做指南：警告，前面有危险

10.2真正的问题是什么

10.3我们所需要的

10.4共性和可变性分析

10.4.1共性分析

10.4.2可变性分析

10.4.3面向对象设计“一箭三雕”

10.5发掘对象的新范式

10.6分析矩阵：一个用例学习

10.7小结

第11章 以开放关闭原则为目标的重构

11.1开放关闭原则

11.1.1从开放关闭原则引申到其他

11.1.2开放关闭原则是一个“原则”

11.2重构

11.2.1为何重构

11.2.2负债还是投资

11.2.3重构和遗留系统

11.2.4以开放关闭原则为目标的重构

11.2.5“及时”设计

11.3小结

第12章 需求与功能接口

12.1迪米特法则

12.2耦合，可恶的耦合，还有依赖

12.2.1耦合和可测试性

12.2.2需求与功能

12.3理想的分离方案：需求接口和功能接口

12.4回到迪米特法则

12.5小结

第13章 何时以及如何使用继承

13.1“四人组”

13.2初始向量，最终结果

13.3优先委托

13.4使用继承与使用委托

13.5继承的使用

13.6可扩展性

13.7在敏捷开发里应用四人组的训诫

13.8测试问题

13.9更多

第四部分 附录

附录A统一建模语言概览
附录B代码质量
附录C封装原始数据类型
.....[\(收起\)](#)

[敏捷技能修炼_下载链接1](#)

标签

敏捷开发

软件工程

敏捷

软件开发

软件设计

Programming

agile

Jolt大奖

评论

: TP311.521/3132

还行，该公司技穷于此矣

对于OO初学者还是非常有用的。我通过本书也了解了更多OO的知识。遗憾的是，我比较期待的意图导向编程与共性可变性分析讲得太简单了。

比较关键的工程实践都覆盖到了，写的也是深入浅出，翻译也可以。

开完第一章，感觉我还没到看这书层次，还是看看模板去

设计和代码层面的敏捷方法

在各种敏捷技能已经成为common sense的今天, 这本书对其中重要的部分总结的很好.

有敏捷实践的基础，看起来会容易很多。但是讲的很浅显很泛。

很多人，或许强调敏捷的一个结果，快速原型。但是本质上的敏捷反而从来不去实践，为实现持续集成需要的架构，自动化工具和实践，测试驱动开发，意图导向的编码，还有敏捷的原则。本书没有过多探讨团队，但是其实强大的团队和流程才是敏捷的核心

很多实用性强的原则和方法，非常值得学习

书中分析了多种开发者必备的软件开发技能，就像一本《九阳真经》，至于能练到哪一重，就靠自己勤学苦练了 :D

不是能全部理解，是自己知识面不够宽广，项目经验不够丰富，工作中遇到问题会再回头研读体会。

都是实践性很强的内容，对于认真思考过软件设计的码农来说浅显易懂， 书不厚

[敏捷技能修炼_下载链接1](#)

书评

《敏捷技能修炼：敏捷软件开发与设计的最佳实践》的4位作者都是世界顶级的软件开发专家和敏捷导师，都有数十年的软件行业从业经验，其中3位曾荣获Jolt大奖。本书是敏捷软件开发领域公认的经典著作，权威性毋庸置疑。
书中内容围绕“敏捷式编程”这一主题展开，对每一位敏捷软件...

抛开人云亦云的敏捷不谈，这本书通过引入一些例子和概念，来引导大家去思考如何进行高质量的软件设计。
读这本书，会使我一直沉浸在印证自己的编程经验的状态中，时而，恍然大悟，时而，追悔莫及。
读这本书，你会像我一样有所收获，不过更重要的是坚持不懈的实践，实践出真知。

[敏捷技能修炼_下载链接1](#)