

## <<浮现式设计>>

### 图书基本信息

书名：<<浮现式设计>>

13位ISBN编号：9787115259783

10位ISBN编号：711525978X

出版时间：2011-8

出版时间：人民邮电

作者：Scott L.Bain

页数：279

译者：赵俐^华洁

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：<http://www.tushu007.com>

## <<浮现式设计>>

### 内容概要

浮现式设计是一种敏捷技术，强调在开发过程中不断演进。

由Scott L.

Bain编著的《浮现式设计：专业软件开发的演进本质》的讨论围绕着专业软件开发方法的演进主题展开，强调了让软件成为一个真正专业的重要性，以及以演进方式开发软件的重大意义。

书中谈到了如何在演进过程中综合运用设计模式、重构、单元测试和测试驱动开发等实践，以及何时制定耦合、内聚和封装等关键决策，而且通过准确生动的示例说明了如何开发出真正有用的软件。

《浮现式设计：专业软件开发的演进本质》主要面向软件开发者群体，尤其是对敏捷开发感兴趣的程序设计人员。

## <<浮现式设计>>

### 作者简介

拥有30年从业经验的资深计算机技术专家。

主要从事开发、工程和设计。

他还曾负责设计、提供和管理认证培训课程。

从事最终用户的技能培训，既有课堂授课，也有远程教育。

过去8年来。

Scott一直在华盛顿州普捷湾畔的Net

Objectives公司工作，负责培训课程以及有关设计模式、重构、单元测试、测试驱动开发的咨询工作。

Scott与Net

Objectives的CEO Alan

Shalloway一起为敏捷环境中的设计模式集成做出了卓越的贡献。

他还经常在各种开发者会议(例如JavaOne和SDWest)上发表演讲。

## &lt;&lt;浮现式设计&gt;&gt;

## 书籍目录

## 第1章 软件开发这个职业

- 1.1 人类制作软件已经有多久的历史了
- 1.2 软件开发是一种什么样的活动
- 1.3 软件开发缺少了什么
- 1.4 谁说了算
- 1.5 独特性

## 第2章 从衣橱到探月

- 2.1 软件开发中的模式 and 专业化
- 2.2 Andrea的衣橱
- 2.3 探月
  - 2.3.1 因素的连锁变化
  - 2.3.2 不同的因素导致不同的设计
  - 2.3.3 还有更多环境因素
  - 2.3.4 成本和获益
  - 2.3.5 火星探险
- 2.4 模式的价值
- 2.5 小结

## 第3章 软件开发的本质

- 3.1 失败率过高
- 3.2 成功的定义
- 3.3 Standish Group
- 3.4 做了错误的事情
- 3.5 做事的方式错了
- 3.6 随着时间的推移, 软件开发也有所改善
- 3.7 一个原因: 土木工程的类比
- 3.8 放弃希望
- 3.9 有时等待和拖延也是必要的
- 3.10 桥是硬的, 软件是软的
- 3.11 我们在变化的海洋中游泳
- 3.12 接受变化
- 3.13 拥抱变化
- 3.14 利用变化
- 3.15 更好的类比: 不断演进的系统
- 3.16 小结

## 第4章 代码的演进: 初级阶段

- 4.1 用对象结构来代替过程逻辑
- 4.2 面向对象和模式的起源
- 4.3 一个示例: 简单条件和Proxy模式
- 4.4 下一步: 多路径条件选择
- 4.5 为什么要采用对象结构
- 4.6 从多个条件中选择一个
- 4.7 小结

## 第5章 使用和发现模式

- 5.1 根据上下文进行设计: 我做的另一个木匠活
- 5.2 模式引出了另一个看问题的角度

## &lt;&lt;浮现式设计&gt;&gt;

5.3 模式提供了一种讨论设计的语言

5.4 本书中的模式

5.5 小结

## 第6章 软件开发金字塔

6.1 构成专业的元素

6.2 一种形象的表现

6.3 小结

## 第7章 注重软件质量

7.1 封装

7.2 内聚

7.2.1 方法内聚

7.2.2 视角层的内聚

7.2.3 类内聚

7.2.4 内聚到何种程度才足够

7.3 耦合

7.3.1 有意耦合与意外耦合

7.3.2 耦合类型

7.4 冗余

7.5 可测试性

7.6 可读性

7.7 软件的病症

7.7.1 内聚性较差的信号

7.7.2 意外耦合或不合逻辑耦合的信号

7.7.3 冗余的信号

7.8 小结

## 第8章 注重原则和智慧结晶

8.1 使用与创建分离

8.1.1 Fowler的三层视角

8.1.2 另一种视角

8.1.3 使用的视角

8.1.4 一个单独的视角：创建

8.1.5 最后考虑构造细节

8.1.6 回到现实

8.2 开闭原则

8.2.1 类级的开闭原则

8.2.2 方法级的开闭原则

8.3 依赖倒置原则

8.4 GoF的建议

8.4.1 设计方法的接口

8.4.2 设计类的接口

8.4.3 GoF：优先使用对象聚合而非类继承

8.5 GoF：在设计中思考什么应该变化并封装会发生变化的概念

8.6 小结

## 第9章 注重实践

9.1 统一编码风格

9.1.1 注释

9.1.2 命名类、方法和变量

## &lt;&lt;浮现式设计&gt;&gt;

- 9.1.3 编码标准的好处
- 9.2 意图导向编程
- 9.3 封装构造函数
  - 9.3.1 原则与实践
  - 9.3.2 做出决定
- 9.4 公共性-可变性分析
- 9.5 实践与自由
- 9.6 小结
- 第10章 注重纪律：单元测试
  - 10.1 测试的经济学
    - 10.1.1 单元测试
    - 10.1.2 先写测试
  - 10.2 JUnit框架
    - 10.2.1 JUnit基础知识
    - 10.2.2 JUnit示例
    - 10.2.3 Rule.java：先编码，再测试
    - 10.2.4 RuleContainer.java：先测试，再编码
    - 10.2.5 消除冗余：@Before和@After
    - 10.2.6 自动化批量测试
    - 10.2.7 异常和单元测试
  - 10.3 模拟对象
    - 10.3.1 MockObject框架
    - 10.3.2 伪对象
    - 10.3.3 依赖注入和Endo-Testing技巧
    - 10.3.4 Endo-Testing
  - 10.4 小结
- 第11章 注重纪律：重构
  - 11.1 重构质量糟糕的代码
  - 11.2 重构质量优秀的代码
  - 11.3 结构变化与功能变化
  - 11.4 重构可帮助你做出选择
  - 11.5 模式可以成为重构的目标
  - 11.6 避免重构：预构
  - 11.7 重构技巧
  - 11.8 重构遗留代码
  - 11.9 小结
- 第12章 测试驱动开发
  - 12.1 何谓测试驱动开发
    - 12.1.1 测试驱动与先写测试
    - 12.1.2 从单元测试的角度来设计
  - 12.2 测试与质量
    - 12.2.1 测试与内聚
    - 12.2.2 测试与耦合
    - 12.2.3 测试与冗余
  - 12.3 测试驱动开发与模式
    - 12.3.1 Strategy模式
    - 12.3.2 乌龟站在乌龟上，一直向下

## <<浮现式设计>>

12.3.3 模拟对象/模拟乌龟

12.4 模拟对象

12.5 模拟乌龟

12.6 测试Decorator模式

12.7 小结

第13章 模式与因素

13.1 在演进的设计中做决策

13.2 Christopher Apexander与他所提出的“因素”

13.2.1 信号处理器示例

13.2.2 PKZip示例

13.2.3 测试与因素

13.3 更多选择，更多因素

13.4 小结

第14章 浮现式设计：案例分析

14.1 问题领域：MWave公司

14.2 团队

14.3 最简单的能够正常运作的设计

14.4 新需求：更复杂的机器

14.5 顺便介绍一下

14.6 更多好消息

14.7 小结：设计是一次漫长而奇特的旅行

第15章 结束语：展望2020年

附录A 演进路径

附录B 示例中用到的模式简介

附录C 有用幻觉之原理

参考书目

## &lt;&lt;浮现式设计&gt;&gt;

## 章节摘录

版权页：插图：先编写类，然后再编写可以验证其行为是否正常的测试，这样做会不会更好？还是先编写测试再编写满足该测试的代码会更好呢？

凭直觉，我会选择先编写代码；毕竟写了代码才有东西去测试。

我过去常常这样想，因为我认为测试是一种验证，而不是意图的表达。

现在，我则认为我的设计是由测试驱动的。

我不是为了验证代码行为而编写测试；我写测试的目的是为了在编写代码之前先定义代码的行为应该是什么。

要这么做，我必须理解行为应该是什么样的，而这对我来说，也是一场很好的检测，让我可以据此判断自己是否“弄明白”了。

在写单元测试时，我实际上是把我对代码的期望写出来。

例如，“如果我向add（）方法传递两个整数，假设是5和3，那么它应该返回的值是8。

”为了编写这个测试，我必须充分理解这一意图，并且能够想出一个或者一组示例，用来测试那个实现我的期望的类。

Douglas Adam在他的书《银河系漫游指南》。

中讲述了一个名叫“深思”的古老的大型超级计算机的故事，它被要求回答“宇宙、生命及万物”的终极问题。

实现这种艰巨的任务也是可能的，尽管如此，但“深思”说：“程序需要运行一段时间。

”确切地说，它需要运行750万年。

一旦开始运行，在完成之前，程序将无法停止（这有点像在要登机时合上自己的笔记本一样），于是那个古老的文明社会在几千年来一直等待着答案。

当这个大日子到来之际，“深思”朗诵出了“宇宙、生命及万物”的终极答案：“四十二。

”显然，这里的问题是，询问“宇宙、生命及万物”这一问题的人根本没有真正明白他们要问的是什么，所以这个答案对他们毫无用途。

人们问“深思”是否可以解释一下为什么答案会是四十二，深思很遗憾地说不能，但是它可以设计出一台更优秀的电脑来解释。

而这台电脑就是地球以及在地球上繁衍生息的人类社会。

换句话说，问题可能比答案更为复杂。



## <<浮现式设计>>

### 媒体关注与评论

“这本书就是一座智慧的金矿！

本书蕴藏的丰富的智慧来自软件工程领域多年经验的累积。

它集合了大量软件开发的最佳实践，并将其运用于浮现式设计的理念之中。

本书是软件开发人员的必读书籍！

我给开发小组领导者的建议是：那些没读过这本书的程序员。

到年末统统开掉！

” ——Amazon.com “这本书的不少代码示例是用，Java写的.也有一些是C#，它们浅显易懂、直指要害。

本书写作风格独特。

Bain轻松随意的叙述手法让枯燥的技术问题看起来妙趣横生，他甚至拿自己现实生活中的小事作类比，读来让人忍俊不禁。

” ——Truewill.net

## <<浮现式设计>>

### 编辑推荐

《浮现式设计：专业软件开发的演进本质》是一部敏捷开发与软件设计的至尊宝典、深入剖析软件开发过程的演进本质、敏捷开发大师AlanShalloway鼎力推荐。

随着软件的不断进化和成熟。

软件开发过程变得越来越复杂、越来越依赖各种方法学和开发方法。

要想让软件兑现它的承诺，提供长期稳定的回报.软件开发必须足够成熟。

成为一门真正的专业。

《浮现式设计:专业软件开发的演进本质》阐明了构建下一代软件的方法。

作者把当今最重要的开发原则汇集成成了一个统一的、流线化的、实用的软件开发方法，他汲取了模式、重构和测试驱动开发的精华，阐述了如何在整个软件生命周期中高效地开发、管理变更以及持续交付健壮、可靠且经济高效的系统。

《浮现式设计:专业软件开发的演进本质》反映了对系统开发的自然流程的深刻理解.并帮助开发人员顺应这个流程，而不是反其道而行之。

Bain向我们介绍了一次只做一个步骤的浮现式设计的原则和实践，展示了如何随着时间的推移逐步促进软件系统的自然演进.使系统更好地工作并提供更大价值。

为了演示这种方法。

书中提供了大量代码示例，最后还提供了一个完整的案例分析。

<<浮现式设计>>

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:<http://www.tushu007.com>