

Apress®

信息科学与技术丛书

[美] Matt Stephens 著
Doug Rosenberg
郑 静 等译

设计驱动测试

Design Driven Testing: Test Smarter,
Not Harder

- 深入讲解 DDT/ICONIX 流程
- 真实项目贯穿全书
- 聪明测试，远离艰辛
- 适合开发人员、测试人员和项目经理



机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书

设计驱动测试

(美) Matt Stephens
Doug Rosenberg 著
郑 静 等译

机械工业出版社

本书主要介绍了设计驱动测试(DDT)的思想和一种全新的软件开发过程——ICONIX。作者希望通过一个个真实而具体的案例告诉读者,如何在实践中达到测试的最佳平衡和优化。全书共分12章,第1~3章介绍了全新的DDT和传统的TDD之间的差异。第4~8章通过一个真实的Web地图案例,讲解了如何在项目实践中运用DDT的思想。第9~12章主要描述了如何在自动化测试、算法测试、单元测试等环节中使用DDT。

本书可供软件开发人员、测试人员以及项目管理人员阅读和参考。

Design Driven Testing: Test Smarter, Not Harder

By Matt Stephens, Doug Rosenberg, ISBN: 978-1-4302-2943-8

Original English language edition published by Apress Media.

Copyright © 2010 by Apress Media

Simplified Chinese-language edition copyright © 2013 by China Machine Press

All rights reserved.

图书在版编目(CIP)数据

设计驱动测试 / (美)斯蒂文斯(Stephens, M.), (美)罗森伯格(Rosenberg, D.) 著; 郑静等译. —北京: 机械工业出版社, 2013.11

信息科学与技术丛书

ISBN 978-7-111-44066-6

I. ①设… II. ①斯… ②罗… ③郑… III. ①软件—测试
IV. ①TP311.5

中国版本图书馆CIP数据核字(2013)第219716号

机械工业出版社(北京市百万庄大街22号 邮政编码100037)

责任编辑: 车 忱

责任印制: 李 洋

三河市宏达印刷有限公司印刷

2014年1月第1版·第1次印刷

184mm×260mm·18.25印张·452千字

0001—3000册

标准书号: ISBN 978-7-111-44066-6

定价: 49.00元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

电话服务

网络服务

社服务中心: (010) 88361066

教材网: <http://www.cmpedu.com>

销售一部: (010) 68326294

机工官网: <http://www.cmpbook.com>

销售二部: (010) 88379649

机工官博: <http://weibo.com/cmp1952>

读者购书热线: (010) 88379203

封面无防伪标均为盗版

出版说明

随着信息科学技术的迅速发展，人类每时每刻都会面对层出不穷的新技术和新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术不断地受到挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机和互联网在社会生活中日益普及，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络和工程应用等内容，注重理论与实践的结合，内容实用、层次分明、语言流畅，是信息科学与技术领域专业人员不可或缺的参考书。

目前，信息科学技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

译者序

在软件开发过程中，测试一直是不可或缺的重要环节。随着软件工程的发展，各种开发方式百花齐放。对于一个从业人员来说，怎样在众多开发模式中找到适合自己项目的方法，是一个必须面对的问题。正确的开发模式有助于缩短开发周期，保证软件质量，常常能在项目开展中起到事半功倍的作用。

本书作者有多年大型软件开发经验，他们从自身工作中总结经验教训，参考流行的敏捷开发模式，提炼出一套全新的理念——设计驱动测试（Design Driven Testing），并介绍了如何在项目中具体实施。希望这本书能有助于开拓软件开发者的思路，并帮助读者找到真正适合自己的软件开发模式。

本书的译者和审校都是有多年开发测试经验的资深软件工程师，希望能够将本书作者的意图准确地传达给读者，使读者从中受益。

参与本书翻译工作的有邢江宁、杨强、詹涛、郑静、诸雯。郑静担任审校。

由于时间仓促，翻译过程中难免有疏漏之处，恳请读者批评指正。

译者

序

Jim 是 Esri 公司 ArcGIS 地图软件产品的开发经理，负责 2000 万行代码的每日构建。他曾经负责管理 Esri 的运输与物流部门，在那里他使用了本书中介绍的设计技术，这使他在预算之内按时完成了多个百万美元级别的软件项目。

很多人在测试的方方面面都有自己的观点，我曾经听说过各种各样的方法、系统、程序、过程、模式、希望、祈祷，甚至运气——然而还是有一些代码路径永远都不会执行。在这一切方法的帮助下，我们必须清楚怎样交出一个坚如磐石的没有 bug 的软件，不是吗？但是，每当需要发布一个更棒、更稳定的版本，如果测试工程师们被问到“你用恰当的方法测试过整个软件吗？”这类问题时，他们仍然会显得底气不足。

上面问题的答案通常是：“我想是的，但是我真的不敢保证把软件的每个地方都测试到了。”最终这个软件还是发布了，这真是太糟糕了。这种结果是保住了技术支持团队的饭碗，因为发布出去的 bug 很快就需要发布升级包或者紧急补丁来修补。其实我们应该能做得更好的，本书将会告诉你怎样去做。

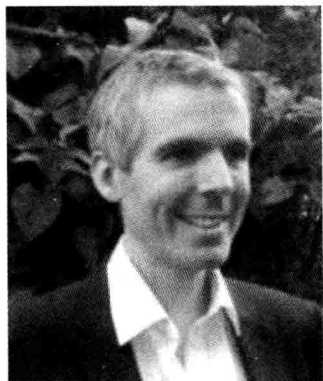
本书将带你体验一个经过验证的软件开发过程——ICONIX 流程（ICONIX Process），并且着重讲解如何使用软件设计来驱动单元测试与验收测试的创建和维护。这就是设计驱动测试（design-driven testing, DDT）。它会告诉你如何使用设计来精确定位哪些关键测试需要建立在设计与对象行为的基础上。这不是测试驱动设计（test-driven design, TDD），不需要首先写单元测试，在设计完之后才开始编码。我不知道你怎么看这个问题，但是我觉得预测未来是件很难的事情，而让工程师编写代码去“适应”一系列测试更困难。

尽管人们对测试有各种各样的观点，但是我想大家一致认同的一点是：测试往往是一项困难并且复杂的工作。作为一个大型团队的开发经理，我明白测试为什么变得难以控制，为什么会导致项目延期。各种组织机构会对测试有着不同的投入，但不幸的是投资回报率都不高。这有可能是对测试投入过多导致浪费了投资。但是更常见的是测试不够（当然通常你认为足够了），比如找错了测试重点，或者是投入不够。这通常是由于你不知道如何在测试上平衡投资并取得准确的测试覆盖率。

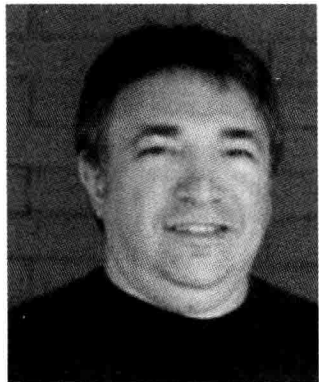
本书通过讲解一个真实的 Web 地图应用的设计与构建来告诉你如何平衡和优化你的测试投资回报率。使用 ICONIX 流程和 DDT 使我们清晰、准确地知道需要做哪些测试以及在哪里做这些测试。此外，使用工具（在本书中用到 Sparx Systems 的 Enterprise Architect 软件）可以自动生成大部分测试代码，这种强大的工具会对你的项目有很大的价值。所以，如果你想在敏捷过程中使用一个可以用很低的成本来自动生成测试的工具，那么这本书很适合你。

Jim McKinney
Esri 公司 ArcGIS 项目开发经理

关于作者



■ **Matt Stephens** 是一家位于伦敦中心的金融机构的软件咨询顾问，并且是独立出版社 Fingerpress (www.fingerpress.co.uk) 的创始人。他为很多杂志和网站（包括 The Register 和 Application Development Trends）供稿。你可以在 Software Reality 在线 (<http://articles.softwarereality.com>) 找到他。

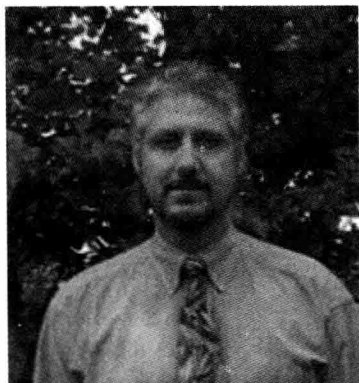


■ **Doug Rosenberg** 于 1984 年在自己的家里创建了 ICONIX (www.iconixsw.com)，在花了几年时间建立了 CASE 工具后，从 1990 年开始他为各个企业讲解面向对象分析与设计。ICONIX 专门从事 UML 与 SysML 的培训，并且提供现场培训与公开课。Doug 在早于 UML 问世之前的 1993 年开发出了统一 Booch/Rumbaugh/Jacobson 建模方法，并且在 1995 年开始写书。

设计驱动测试是他在软件工程方面的第六本书（也是与 Matt Stephens 合作出版的第四本）。他还编写了多本多媒体教程，如 Enterprise Architect for Power Users，及很多电子书，如 Embedded Systems Development with SysML。

他在写书和培训之余，喜欢拍摄全景的虚拟现实的照片，你可以在他的网站 VResorts.com 上看到。

关于技术评审人



■ **Jeffrey P. Kantor** 是大型太空搜索望远镜数据管理系统 (LSST) 的项目经理。**Kantor** 负责实现计算和通信系统, 其功能是校验、质量评估、处理、存储以及终端用户和外部系统访问 LSST 产生的天文图片和工程数据。

Kantor 曾在美国军队中担任了四年的俄语语言与信号专家, 自 1980 年开始, 他曾在许多行业的 IT 部门担任各个级别的职位, 其中包括航空航天和国防、半导体制造、地球物理、软件工程咨询、智能家居、耐用消费品制造、零售以及电子商务。

他为很多企业、政府与研究所创建、调整、应用、审计过软件开发过程。他帮助很多组织通过了 ISO9000 认证与 SEI CMM 2 评估。**Kantor** 曾为 30 多个组织提供了面向对象分析设计、UML、用例驱动测试和项目管理相关的培训与咨询。

他非常喜欢与家人在一起, 并且热爱足球 (运动员、裁判、教练) 和山地自行车运动。

■ **David Putnam** 有 25 年软件开发经验, 他是英国本世纪最热情的敏捷支持者。

从早期为建筑业制作数据库系统开始, 他同时兼任大学讲师。在这段时间中, 他发表了很多关于软件开发的文章并经常出席敏捷开发会议。**David** 现在是一个独立的敏捷开发咨询顾问, 他为多家著名的英国公司提供了很多宝贵的建议。

致 谢

作者致谢：

Esri 的 Mapplet 项目组，包括：Wolfgang Hall, Prakash Darbhamulla, Jim McKinney, Witt Mathot。他们负责书中项目例子的编写。

Sparx Systems 的工作人员，包括：Geoff Sparks, Ben Constable, Tom O'Reilly, Aaron Bell, Vimal Kumar, Estelle Gleeson。他们负责开发 ICONIX 插件和结构化场景编辑器。

Alanah Stephens——她永远是我们最好的“爱丽丝”；Michelle Stephens，在另一本书的项目中，她表现出极大的耐心。

来自 Large Synoptic Survey Telescope 项目的 Jeff Kantor 和 Robyn Allsman。他们帮助说明了用例线程扩展。

Mike Farnsworth 和来自弗吉尼亚 DMV 的其他的人们。他们参加了 Mapplet 的建模研讨会。Barbara Rosi-Schwartz 对 DDT 提供了反馈意见，Jerry Hamby 让我们分享了他的 Flex 经验，并对 MXML 提供了逆向工程的帮助。

最后，还要感谢 Apress 的编辑：Jonathan Gennick, Anita Castro 和 Mary Ann Fugate。

开 场 白

当心那些流行的敏捷开发

已经下午四点了，那些不靠谱的代码今天第十次开始自动编译，最脆弱的是那些索引卡，总是不停地准备重构啊重构。

亲爱的孩子，当心那些流行的敏捷开发，越多代码越多 bug，代码重构看上去很美，你会因此变成高手。

不停地编写单元测试，期望以此避免讨厌的 bug，TDD 让你头晕目眩，其实你只想要些简单的验证。

然而，必须按时完成，幸亏一切顺利，最后期限到来之前，测试全部通过，一切看上去很美。

下午两点半了，我想我们已经完成，是时候来点美味的点心。

什么，还有另一种方法？瞧瞧这个神奇的点子，先做设计而非测试，终于找到了个更简单的方式，真是美妙的一天。

已经下午四点了，那些不靠谱的代码今天第十次开始自动编译，薄薄的索引卡，依旧不停地重构啊重构。

目 录

出版说明	
译者序	
序	
关于作者	
关于技术评审人	
致谢	
开场白	

第一部分 DDT vs. TDD

第 1 章 有人弄反了	2
DDT 要解决的问题	2
很难知道什么时候完成	3
将测试放在后期代价更大	3
测试设计糟糕的代码很困难	3
用户级测试很容易被遗忘	4
开发人员变得自负	4
测试有时缺少目标	4
对 DDT 的与工具无关的快速概览	4
DDT 的结构	5
DDT 实战	6
TDD 与 DDT 的不同之处	7
示例项目：Mapplet 2.0 介绍	8
小结	10
第 2 章 使用 TDD 的 Hello World	12
TDD 的十大特性	12
10. 测试驱动设计	12
9. 完全没有文档	13
8. 所有东西都是单元测试	13
7. TDD 测试不是完全的单元测试	13
6. 验收测试提供针对需求的反馈	13
5. TDD 导致盲目自信的变更	13
4. 设计在不断增长	14
3. 有一些预先设计就可以了	14
2. TDD 产生了大量测试	14

1. TDD 实在太难了	14
使用 TDD 实现登录用例	14
理解需求	15
考虑设计	16
编写第一个测试先行的测试	17
编写登录检查代码从而使测试通过	21
创建模拟对象	22
从重构代码看设计的浮现	24
TDD 中的验收测试	30
结论: TDD 实在太难了	30
小结	31
第 3 章 使用 DDT 的 Hello World	32
ICONIX/DDT 的十大特性	32
10. DDT 包含业务需求测试	32
9. DDT 包含场景测试	33
8. 测试是被设计驱动的	33
7. DDT 包含控制器测试	33
6. DDT 测试更灵活, 更简单	33
5. DDT 中的单元测试是“经典”的单元测试	33
4. DDT 中的测试用例可以转换成测试代码	34
3. DDT 测试用例指导测试计划	34
2. DDT 测试对开发和测试团队都很有用	34
1. DDT 可以消除重复工作	34
使用 DDT 实现登录	34
步骤 1: 创建健壮性图	36
步骤 2: 创建控制器测试	38
步骤 3: 添加场景	40
步骤 4: 将控制器测试用例转换成类	42
步骤 5: 生成控制器测试代码	43
步骤 6: 绘制序列图	46
步骤 7: 创建单元测试用例	48
步骤 8: 填充测试代码	52
小结	56

第二部分 真实世界中的 DDT: Mapplet 2.0 旅游网站

第 4 章 Mapplet 项目简介	59
ICONIX 流程/DDT 十大“To-Do”列表	60
10. 创建架构	60
9. 对需求达成共识并进行测试	61

8. 从问题域驱动设计	63
7. 使用 UI 故事板编写用例	65
6. 编写场景测试验证用例	66
5. 测试概要设计和详细设计	69
4. 经常更新模型	69
3. 保持测试脚本与需求同步	73
2. 更新自动化测试	73
1. 比较待发布版本和原始用例	74
小结	77
第 5 章 详细设计和单元测试	78
单元测试十大 “To-Do” 列表	79
10. 从序列图开始	79
9. 在设计中标识测试用例	81
8. 为每个测试用例编写场景	82
7. 聪明测试：避免重叠测试	84
6. 把测试用例转换为 UML 类	85
5. 编写单元测试和相关的代码	89
4. 编写白盒单元测试	92
3. 使用模拟对象框架	96
2. 用单元测试测试算法逻辑	99
1. 编写集成测试的独立套件	99
小结	100
第 6 章 概要设计和控制器测试	101
控制器测试十大 “To-Do” 列表	102
10. 从健壮性图开始	102
9. 为控制器标识测试用例	105
8. 为每个测试用例定义一个或者多个场景	107
7. 填写描述、输入和验收标准	110
6. 生成测试类	110
5. 实现测试代码	114
4. 编写容易测试的代码	115
3. 编写 “灰盒” 控制器测试	117
2. 串联控制器测试	118
1. 编写集成测试的独立套件	119
小结	120
第 7 章 验收测试：扩展用例场景	121
场景测试的十大 “To-Do” 列表	122
Mapplet 用例	122
10. 从一个叙述性用例开始	122

9. 把这个用例转换成一个结构化的场景	125
8. 确保涵盖所有的可选方案和意外场景	126
7. 增加前置条件和后置条件, 将每个场景分支连接起来	126
6. 生成活动图来检查结构化场景	127
5. 创建外部测试集来细化场景	128
4. 把测试用例放进用例图	129
3. 进入 EA 测试视图	129
2. 根据需要细化场景	130
1. 为测试团队生成测试计划文档	130
这个过程的精髓是.....	131
小结	134
第 8 章 验收测试: 业务需求	135
十大需求测试 “To-Do” 列表	136
10. 从一个域模型开始	136
9. 编写业务需求测试	138
8. 对需求进行建模和整理	138
7. 从需求创建测试用例	139
6. 与用户一起审查你的计划	141
5. 编写手工测试脚本	143
4. 编写自动化需求测试	143
3. 导出需求测试用例	144
2. 使测试用例可见	144
1. 让你的团队参与其中!	144
小结	145

第三部分 高级 DDT

第 9 章 单元测试的反模式 (反面案例)	147
末日圣殿 (特指某一种代码)	148
大背景	148
HotelPriceCalculator 类	149
支持类	151
服务类	152
反模式	154
10. 复杂的构造函数	154
9. 滥用类继承	155
8. 静态微触发器	157
7. 静态方法和变量	159
6. 单例设计模式	160
5. 紧耦合	162

4. UI 代码里实现业务逻辑	164
3. 滥用私有属性	165
2. 声明为 final 的服务对象	166
1. 热心的程序员开发的不成熟的功能	166
小结	167
第 10 章 为易于测试而设计	168
十大为测试而设计的“To-Do”列表	168
末日圣殿——彻底修正	169
用例——确定我们需要做什么	170
识别控制器测试	171
计算总价测试	172
获取最新价格测试	172
为易于测试而设计	173
10. 将初始化代码放在构造函数之外	173
9. 慎用继承	174
8. 避免使用静态初始化块	175
7. 使用对象级别的方法和变量	176
6. 避免使用单例设计模式	176
5. 保持类解耦合	178
4. 将业务逻辑放在 UI 代码之外	179
3. 使用“黑盒”和“灰盒”测试	184
2. 为常量预留“final”修饰符——通常需要避免修饰复杂类型（如 Service Objects）为 final	184
1. 坚持使用用户用例和设计	185
Quote Hotel Price 用例的详细设计	185
控制器测试：计算总价	186
控制器测试：获得最新价格的测试	187
重构设计和代码	187
小结	189
第 11 章 自动化的集成测试	190
十大集成测试“To-Do”列表	190
10. 在概要设计里寻找测试模式	191
9. 不要忘记安全性测试	192
安全性测试：SQL 注入攻击	192
安全性测试：建立安全会话	193
8. 决定编写哪个“等级”的集成测试	194
三个等级的不同点	194
了解编写哪个等级的集成测试	194
7. 概要设计驱动单元/控制器级别的集成测试	195
6. 从用例场景驱动场景测试	198

5. 编写端到端场景测试	198
模拟一个场景中的步骤	199
共享测试数据库	199
Mapplet 例子：“高级搜索”用例	201
Vanilla xUnit 场景测试	201
4. 使用“业务友好”型测试框架	202
3. 将测试 GUI 代码作为场景测试的一部分	204
2. 不要低估集成测试的难度	204
网络延迟	206
数据库元数据变化	206
随机变化的（又名“敏捷”）接口	206
远程系统中的 bugs	207
阴雨天	207
1. 不要低估集成测试的价值	207
编写集成测试的关键点	208
小结	209
第 12 章 单元测试算法	210
十大算法测试“To-Do”列表	211
10. 从概要设计的控制器开始工作	211
9. 将控制器扩展成算法设计	213
8. 把图和域模型对应起来	214
7. 分割那些看上去不止做一个检查的判断结点	214
6. 为每个结点（活动和判断结点）建立一个测试用例	215
5. 为每个测试用例定义测试场景，一组输入和期望结果	216
4. 按照算法，从不同的源中创建输入数据	218
3. 把逻辑流程对应到独立的方法和类上	219
2. 编写“白盒”单元测试	223
1. 在其他类型的设计图上使用 DDT 技术	232
小结	232
附录 爱丽丝漫游用例国	234
介绍	234
第 1 部分	235
爱丽丝在看书的时候睡着了	235
用例驱动开发的承诺	236
一种把用例文本和对象连接起来的分析模型	236
简洁且直接	236
<<包含>> 还是<<扩展>>	236
我们迟到了！我们必须开始编码了！	237
爱丽丝想知道如何才能把用例变成代码	237

抽象的……基本的237

有点太过抽象了？237

目的中心化.....238

我们真的打算为每个用例都指定这些东西吗？238

第 2 部分239

 爱丽丝口渴了240

 爱丽丝感到头晕240

 设想……（敬请约翰·列侬原谅，这首歌改编自他的作品）240

 结对编程意味着再也不用把需求写下来了241

 没时间去写需求了242

 你也许也会说“代码就是设计”242

 谁在乎用例？243

 C3 项目被中止了243

 一次且只有一次？244

 没有写下需求之前，爱丽丝拒绝开始写代码245

 你因为预先设计而被定罪246

 CMM 已经死了，砍掉她的脑袋！247

 一些严肃的设计重构247

第 3 部分247

 爱丽丝醒了248

 缩小“什么”和“如何”之间的距离248

 静态模型和动态模型被连接在了一起248

 行为被定位到序列图里248

 这里面的教训在于.....249

尾声——乱七八糟的测试.....250

索引253