

实例化需求



[实例化需求_下载链接1](#)

著者:Gojko Adzic

出版者:人民邮电出版社

出版时间:2012-9

装帧:平装

isbn:9787115290267

《实例化需求：团队如何交付正确的软件》是在世界各地调查了多个团队软件交付过程后的经验总结。《实例化需求：团队如何交付正确的软件》介绍了这些团队如何在很短的周期内说明需求、开发软件，并交付正确的、无缺陷的产品；为团队在实施实例化需求说明时使用的模式、想法和工件创建了一致的语言；展示了案例中的团队用来实现实例化需求说明原则的关键性实践；并在案例分析部分展示了一些团队实施实例化需求说

明的历程。

《实例化需求：团队如何交付正确的软件》适合与项目管理、开发、测试、交付有关的人员阅读。

作者介绍:

Gojko Adzic

战略软件交付顾问，专注于敏捷和精益开发，尤其擅长敏捷测试、实例化需求和行为驱动开发。Gojko经常在国际上重要的软件开发和测试会议上发言，并运营着英国的敏捷测试用户小组。最近这十多年来，他一直在财务和能源交易平台、移动定位、电子商务、在线游戏和复杂配置管理系统等行业项目中，从事程序员、架构师、技术指导和顾问等工作。除本书外，他还著有Bridging the Communication Gap、Test Driven.Net Development with FitNesse和The Secret Ninja Cucumber Scrolls等书。

张昌贵 (Steven Zhang)

软件开发经理，CSM, CSPO, CSP，敏捷软件开发参与者，软件开源运动拥护者。

张博超 (Jackson Zhang)

软件开发工程师，CSM, CSPO, CSP。关注敏捷开发，积极实践并推广各种敏捷方法。个人博客：iagile.me/blog

石永超 (Stone Shi)

软件开发工程师，CSM，CSPO，敏捷爱好者，InfoQ中文站编辑。关注高效、高质量的软件开发方法。

目录: 目录

第一部分 开始

第1章 主要优点 2

1.1 更有效地实施变更 4

1.2 更高的产品质量 5

1.3 减少返工 8

1.4 更好的协作 10

1.5 铭记 11

第2章 关键过程模式 12

2.1 从目标中获取范围 13

2.2 协作制定需求说明 14

2.3 举例说明 14

2.4 提炼需求说明 15

2.5 自动化验证时不修改需求说明 15

2.6 频繁验证 17

2.7 演化出一个文档系统 17

2.8 实际的例子 18

2.8.1 商业目标 18

2.8.2 范围 18

2.8.3 关键实例 18

2.8.4 带实例的需求说明 19

2.8.5 可执行的需求说明	20
2.8.6 活文档	20
2.9 铭记	20
第3章 活文档	21
3.1 为什么我们需要权威的文档	22
3.2 测试可以是好文档	22
3.3 根据可执行的需求说明创建文档	23
3.4 以文档为中心的模型所具有的好处	25
3.5 铭记	25
第4章 开始改变	26
4.1 如何开始改变过程	27
4.1.1 把实施实例化需求说明当作更广阔的过程变更的一部分	27
4.1.2 专注于提高质量	27
4.1.3 从功能测试自动化开始	28
4.1.4 引入一个可执行需求说明的工具	29
4.1.5 使用测试驱动开发作为踏脚石	30
4.2 如何开始改变团队文化	31
4.2.1 避免使用“敏捷”术语	31
4.2.2 确保你得到管理层的支持	32
4.2.3 把实例化需求说明当作是比执行验收测试更好的方式来推销	33
4.2.4 不要让测试自动化成为最终的目标	34
4.2.5 不要太关注工具	34
4.2.6 在迁移过程中，遗留脚本也要有人维护	35
4.2.7 跟踪哪些人在运行（以及没有运行）测试自动检查程序	35
4.3 团队如何在流程和迭代中集成协作	36
4.3.1 Ultimate软件公司的Global Talent Management团队	37
4.3.2 BNP Paribas银行的Sierra团队	38
4.3.3 天空网络服务部门	39
4.4 处理签收和可追溯性	40
4.4.1 在版本控制系统中保存可执行需求说明	41
4.4.2 通过导出的活文档来签收	41
4.4.3 签收的是范围，而非需求说明	41
4.4.4 在“精简的用例”上签收	42
4.4.5 引入用例实现	42
4.5 警告信号	43
4.5.1 注意频繁改动的测试	43
4.5.2 当心回退	44
4.5.3 注意组织级的失调	44
4.5.4 当心“以防万一”的代码	44
4.5.5 注意霰弹式修改	45
4.6 铭记	45
第二部分 关键过程模式	
第5章 从目标中获取范围	48
5.1 构建正确的范围	49
5.1.1 理解“为什么”和“谁”	50
5.1.2 理解价值从何而来	51
5.1.3 了解商业用户预期的输出是什么	52
5.1.4 让开发人员提供用户故事的“我想要”部分	53
5.2 在没有高层次控制权的情况下，协作确定范围	53
5.2.1 询问“为什么这些东西有用？”	54
5.2.2 询问替代方案	54
5.2.3 不要只顾最低层次的需求	55
5.2.4 确保团队交付完整的功能	55

- 5.3 更多信息 56
- 5.4 铭记 56
- 第6章 通过协作制定需求说明 58
 - 6.1 为什么需要协作制定需求说明 58
 - 6.2 最热门的协作模型 59
 - 6.2.1 尝试大型的全体工作坊 59
 - 6.2.2 尝试小型工作坊（“神勇三剑客”） 61
 - 6.2.3 结对编写 62
 - 6.2.4 让开发人员在迭代开始前频繁地审查测试 63
 - 6.2.5 尝试非正式交谈 64
 - 6.3 准备协作 65
 - 6.3.1 举办介绍会 65
 - 6.3.2 邀请项目干系人 66
 - 6.3.3 进行具体的准备工作并事先审查 67
 - 6.3.4 让团队成员尽早审查故事 68
 - 6.3.5 只准备初始的实例 69
 - 6.3.6 不要让过度的准备阻碍了讨论 69
 - 6.4 选择协作模型 70
 - 6.5 铭记 71
- 第7章 举例说明 72
 - 7.1 举例说明：一个例子 74
 - 7.2 例子必须精确到位 75
 - 7.2.1 不要在例子中出现“是／否”的回答 75
 - 7.2.2 避免使用等价抽象类 75
 - 7.3 例子必须完整 76
 - 7.3.1 用数据作试验 76
 - 7.3.2 使用替代方法来检验功能 76
 - 7.4 例子必须要真实 77
 - 7.4.1 避免虚构自己的数据 77
 - 7.4.2 直接从客户那里获得基本的例子 78
 - 7.5 例子应该易于理解 79
 - 7.5.1 避免探讨所有可能的组合 80
 - 7.5.2 寻找隐含的概念 80
 - 7.6 描述非功能性需求 81
 - 7.6.1 取得精确的性能需求 82
 - 7.6.2 为UI使用低保真度的原型 82
 - 7.6.3 试用QUPER模型 83
 - 7.6.4 讨论时使用核查清单 84
 - 7.6.5 建立一个参照的例子 84
 - 7.7 铭记 85
- 第8章 提炼需求说明 86
 - 8.1 一个好的需求说明的例子 87
 - 8.1.1 免费送货服务 87
 - 8.1.2 实例 87
 - 8.2 一个劣质需求说明的例子 88
 - 8.3 提炼需求说明时要关心什么 90
 - 8.3.1 实例要精确可测 90
 - 8.3.2 脚本不是需求说明 90
 - 8.3.3 不要使用流程式的描述 91
 - 8.3.4 需求说明应关注业务功能，而不是软件设计 92
 - 8.3.5 避免编写与代码紧密耦合的需求说明 92
 - 8.3.6 不要在需求说明中引入技术难点的临时解决方案 93
 - 8.3.7 不要陷入到用户界面的细节里 93
 - 8.3.8 需求说明应该是不言自明的 94

- 8.3.9 使用叙述性标题并使用短篇幅阐释目标 94
- 8.3.10 展示给别人看并保持沉默 94
- 8.3.11 不要过度定义实例 95
- 8.3.12 从简单的例子入手，然后逐步展开 96
- 8.3.13 需求说明要专注 97
- 8.3.14 在需求说明中使用“Given-When-Then”语言 97
- 8.3.15 不要在需求说明中明确建立所有依赖 98
- 8.3.16 在自动化层中应用缺省值 99
- 8.3.17 不要总是依赖缺省值 99
- 8.3.18 需求说明应使用领域语言 100
- 8.4 提炼实战 100
- 8.5 铭记 102
- 第9章 自动化验证而不修改需求说明 103
- 9.1 非得自动化吗 104
- 9.2 从自动化开始 105
 - 9.2.1 为了学习工具，先尝试一个简单的项目 105
 - 9.2.2 事先计划自动化 106
 - 9.2.3 不要拖延自动化工作或将其委派他人 107
 - 9.2.4 避免根据原有的手动测试脚本进行自动化 107
 - 9.2.5 通过用户界面测试赢得信任 108
- 9.3 管理自动化层 109
 - 9.3.1 别把自动化代码当作二等公民 109
 - 9.3.2 在自动化层里描述验证过程 110
 - 9.3.3 不要在测试自动化层里复制业务逻辑 111
 - 9.3.4 沿着系统边界自动化 112
 - 9.3.5 不要通过用户界面检查业务逻辑 113
 - 9.3.6 在应用程序的表皮之下进行自动化 113
- 9.4 对用户界面进行自动化 115
 - 9.4.1 以更高层次的抽象来详细说明用户界面的功能 115
 - 9.4.2 UI需求说明只检查UI功能 117
 - 9.4.3 避免录制的UI测试 117
 - 9.4.4 在数据库中建立环境 118
- 9.5 管理测试数据 119
 - 9.5.1 避免使用预填充数据 119
 - 9.5.2 尝试使用预填充的引用数据 120
 - 9.5.3 从数据库获取原型 120
- 9.6 铭记 121
- 第10章 频繁验证 122
- 10.1 提高稳定性 123
 - 10.1.1 找出最烦人的问题并将其解决掉，然后不停地重复 123
 - 10.1.2 用CI测试历史找到不稳定的测试 124
 - 10.1.3 搭建专用的持续验证环境 125
 - 10.1.4 使用全自动部署 125
 - 10.1.5 为外部系统创建较简单的测试替代品 125
 - 10.1.6 选择性地隔离外部系统 126
 - 10.1.7 尝试多级验证 127
 - 10.1.8 在事务中执行测试 127
 - 10.1.9 对引用数据做快速检查 128
 - 10.1.10 等待事件，而非等待固定时长 128
 - 10.1.11 将异步处理变成可选 129
 - 10.1.12 不要用可执行需求说明做端到端的验证 129
- 10.2 获得更快的反馈 130
 - 10.2.1 引入业务时间 130

- 10.2.2 将较长的测试分割成较小的模块 131
- 10.2.3 避免使用内存数据库做测试 131
- 10.2.4 把快速的和缓慢的测试分开 132
- 10.2.5 保持夜间测试的稳定 132
- 10.2.6 为当前迭代创建一个测试包 133
- 10.2.7 并行运行测试 133
- 10.2.8 禁用风险较低的测试 134
- 10.3 管理失败的测试 135
 - 10.3.1 创建已知失败了的回归测试包 135
 - 10.3.2 自动检查那些被禁用的测试 136
- 10.4 铭记 137
- 第11章 演化出文档系统 138
 - 11.1 活文档必须易于理解 138
 - 11.1.1 不要创建冗长拖沓的需求说明 138
 - 11.1.2 不要使用许多小的需求说明来描述单个功能 139
 - 11.1.3 寻找更高层次的概念 139
 - 11.1.4 避免在测试中使用技术上的自动化概念 139
 - 11.2 活文档必须前后一致 140
 - 11.2.1 演化出一种语言 141
 - 11.2.2 将需求说明语言拟人化 142
 - 11.2.3 协作定义语言 143
 - 11.2.4 将构建模块文档化 143
 - 11.3 活文档必须组织得井井有条，便于访问 144
 - 11.3.1 按用户故事组织当前的工作 144
 - 11.3.2 按功能区域组织用户故事 145
 - 11.3.3 按用户界面的导航路径组织 146
 - 11.3.4 按业务流程来组织 146
 - 11.3.5 引用可执行需求说明时请使用标签而不要使用URL 147
 - 11.4 聆听活文档 147
 - 11.5 铭记 148
- 第三部分 案例研究
- 第12章 uSwitch 152
 - 12.1 开始改变流程 152
 - 12.2 优化流程 154
 - 12.3 当前的流程 156
 - 12.4 结果 157
 - 12.5 重要的经验教训 157
- 第13章 RainStor 159
 - 13.1 改变流程 159
 - 13.2 当前流程 161
 - 13.3 重要的经验教训 162
- 第14章 爱荷华州助学贷款公司 163
 - 14.1 改变流程 163
 - 14.2 优化流程 164
 - 14.3 活文档作为竞争优势 166
 - 14.4 重要的经验教训 167
- 第15章 Sabre Airline Solutions 168
 - 15.1 改变流程 168
 - 15.2 改善协作 169
 - 15.3 结果 171
 - 15.4 重要的经验教训 171
- 第16章 ePlan Services 172
 - 16.1 改变流程 172
 - 16.2 活文档 174

- 16.3 当前的流程 175
- 16.4 重要的经验教训 176
- 第17章 Songkick 177
 - 17.1 改变流程 177
 - 17.2 当前的流程 179
 - 17.3 重要的经验教训 180
- 第18章 思想总结 182
 - 18.1 协作制定需求能在项目干系人与交付团队之间建立信任 182
 - 18.2 协作需要事先准备 183
 - 18.3 协作的方式多种多样 183
 - 18.4 将最终目的视为业务流程文档，不失为一种有用的模型 184
 - 18.5 活文档带来的长期价值 184
- 附录A 资源 186
 - • • • • [\(收起\)](#)

[实例化需求_下载链接1](#)

标签

软件工程

需求分析

项目管理

软件开发

敏捷

需求

产品经理

计算机

评论

理念说得是不错的，就是需要具体来实现才行。译文偏晦涩，有些影响阅读。

挺杂乱的。

应用系统，我该如何恨你！？

除了规格文档可测试外，不值一提，只能说老外写书赚钱太容易了

读了两遍，第二遍收获颇丰。读一本书，如果能给你带来一点感悟，那就赚回书钱了。如果带给你两三点感悟，花费的时间也就值了。如果带给你五六点，甚至七八点，那就可以为作者捐款以示感谢了。

作者写的不错，翻译的一般

仍然觉得实例太少。书中给出的实例其实是一种假象，总是三言两语，隔靴搔痒。当然，书中这种实例化需求的观点是可以点赞的。

真正关心软件如何解决“业务问题”的书，也许geek不爱看，但这正是很多编写企业级软件的、自认为是geek的人所欠缺的。读这本书有十年前读“极限编程”时的震撼。

只是粗略看到很好地讲述需求提取章节的内容，已经超出我的预期了，继续学习

BDD

对我来说，重要的是清楚区分了BDD和ATDD，一个是描述需求，一个是测试自动化，出发点的不同决定了一切

我总觉得翻译过来的书,好多都是扯犊子.

很多术语！入行不深，很难看懂。

口口相传,改来改去,是我经历与看到过的大多数软件开发情形,自己也经历过,项目经理不做需求控制,产品经理不断改需求,导致整个项目推不动,每天项目进度像驴拉磨一样--原地打转,这本书给了我一些启示与答案.

充满了真知灼见的书，赞！！！！

收获最大的是有了BDD和ATDD的区分。需求不需要脚本话的描述，需求应该是以用户的角度出发，采用Given—When—Then的语法方式来统一大家的认识。

其实这本书我觉得更适合测试工程师看....特别是在敏捷中做的自动化测试工程师

最早提出了活文档的概念，实例化需求这个用语确实也是比BDD更加突出重点。这本书基本只描述了整理需求的内容，没有描述具体工具的使用。前面对于术语的说明我感觉倒是挺不错的。整体看，感觉读BDD in Action也够了。

理念挺好的，就是有点啰嗦

读书过半才觉有所收获，前半部分实在觉得过于琐碎

[实例化需求_下载链接1](#)

书评

这个礼拜终于断断续续用了空档时间读完了一本买了却一直没时间坐下来好好研究的书。这本书给我一种很奇妙的读后感，因为书中既没有程式码，也不介绍任何工具，甚至实际软件例子也很少，篇幅最多的甚至是模糊的团队访谈。但读完了以后，却让我在软件开发上流程上有了更大的启...

想看懂这本书最好先了解一下 BDD和一些BDD测试框架： FitNesse JBehave Cucumber等等 本书的关键point BDD Given when then 使用BDD的test case当作需求文档，即所谓的“活文档” 由测试和开发一起维护“活文档” 自动化功能测试 其他的point比如敏捷、频繁验证在别...

一年多前做过一个项目，在需求阶段陷入沼泽，需求的含糊，语言表达的差异，以及操作流程的变化带来的差异性，导致了与用户的需求沟通一团糟。那个时候我和团队做出了一个决定，把各种可能的场景 - 操作的过程通过示例的方式表达出来，先和用户确认在各种不同的场景期望行为是什...

现在只可以在图书馆借到，求购二手《实例化需求》，请加我QQ147886833
现在只可以在图书馆借到，求购二手《实例化需求》，请加我QQ147886833
现在只可以在图书馆借到，求购二手《实例化需求》，请加我QQ147886833
现在只可以在图书馆借到，求购二手《实例化需求》，请加我QQ147...

实例化需求 下载链接1