软件开发与设计丛书

MANNING

- 敏捷开发新经典文献
- 亚马逊网站全五星评价
- 成功开发软件的必备秘籍

实例化需求

团队如何交付正确的软件

[塞尔维亚] Gojko Adzic 著　　张昌贵　张博超　石永超　译

Specification by Example

How Successful Teams Deliver the Right Software

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

实例化需求

团队如何交付正确的软件

[塞尔维亚] Gojko Adzic 著　　张昌贵　张博超　石永超　译

Specification by Example

How Successful Teams Deliver the Right Software

人民邮电出版社
北　京

图书在版编目（CIP）数据

实例化需求 : 团队如何交付正确的软件 / (塞尔)阿契克 (Adzic,G.) 著 ; 张昌贵, 张博超, 石永超译. 北京 : 人民邮电出版社, 2012.9
(图灵程序设计丛书)
书名原文: Specification by Example:How Successful Teams Deliver the Right Software
ISBN 978-7-115-29026-7

Ⅰ. ①实… Ⅱ. ①阿… ②张… ③张… ④石… Ⅲ. ①软件开发－电子计算机工业－工业企业管理－组织管理 Ⅳ. ①F407.676.17

中国版本图书馆CIP数据核字(2012)第168201号

内 容 提 要

本书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。书中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。

本书适合与项目管理、开发、测试、交付有关的人员阅读。

图灵程序设计丛书

实例化需求：团队如何交付正确的软件

◆ 著 [塞尔维亚] Gojko Adzic
译 张昌贵 张博超 石永超
责任编辑 傅志红
执行编辑 李 瑛

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：13
字数：307千字 2012年9月第1版
印数：1-4 000册 2012年9月北京第1次印刷

著作权合同登记号 图字：01-2012-4260号

ISBN 978-7-115-29026-7

定价：49.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版权声明

前　言

你手里正拿着或正在屏幕上翻看的这本书，是一系列研究的成果。我们调查了世界各地的多个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。本书呈现的是集体智慧，从公共网站到内部支持系统，涉及大大小小约50个项目。这些项目包含了各种各样的团队，有在一个办公室里办公的小团队，也有跨越大洲的集团公司，他们使用了众多过程，包括极限编程（XP）、Scrum、看板（Kanban）以及一些类似的方法（通常附带有敏捷或精益的字眼）。这些项目有个共同点——项目需求说明和测试能够良好配合，项目组从中获益良多。

实例化需求说明

如何处理需求说明与测试，不同的团队使用不同的名称，但它们都有一套共同的核心原则与思想，而我认为它们在本质上是一致的。很多团队使用以下这些名称来命名这类实践：

- 敏捷验收测试
- 验收测试驱动开发
- 实例驱动开发
- 故事测试
- 行为驱动开发
- 实例化需求说明

相同的做法却有如此多的名字，这事实上也反映了当前在这一领域内有着大量的创新。同时它还反映了一个事实，本书描述的这些做法，影响了团队的需求描述、开发和测试等方面。为保持一致，必须选择一个名字。本书将采用*实例化需求说明*（Specification by Example）这个名称。至于为何选它，稍后的“谈谈术语”一节将详细解释。

实践出真知

本书通过案例研究和访谈来呈现这个话题。之所以选择这种方式，是为了让读者能看到目前真的有团队正在这么做，并且从中获益良多。实例化需求不是一门神秘艺术，虽然有些主流媒体会使人这么觉得。

书中的一切几乎都是来自于现实世界、真实的团队以及切实的经验。有一小部分实践是作为

建议提出的，并没有真实案例研究的支持。我认为这些思想对将来会很重要，也正是因为如此，我才明确地提出它们。

我很确定，我所主导的研究以及我得出的结论，虽然促成了本书的编写，但由于这并不是一项严肃的科学研究，将会被那些号称敏捷开发不可用、业界应该回到“真正的软件工程”[①]的怀疑论者所排斥。那也没关系。相比一项严肃的科学研究所需的资源，编写本书时我接触到的资源是十分有限的。但即使我拥有那些资源，我也不是一个科学家，而且我也不用伪装我自己。我是一名实践者。

本书读者对象

如果你像我一样，是一名实践者，并且靠软件谋生，那么这本书可以提供很多帮助。有些团队已经尝试去实施敏捷过程，并且碰到了低质量、返工以及未达客户预期等问题，本书主要就是写给这些团队的。（没错，这些都是问题。简单地迭代只是权宜之计，并非解决方案。）实例化需求说明、敏捷验收测试、行为驱动开发，以及其他不同叫法所指的这个实践，都能解决这些问题。无论你是测试人员、开发人员、业务分析师，还是产品负责人，这本书都可以帮助你开始实施这些实践，并学习如何用它们在团队中做出更多贡献。

几年前，我在大会上碰到的大多数人都没听说过这些思想。而现在，我碰到的大部分人都或多或少知道这些实践，但是很多人都未能妥善落实。在实施敏捷开发的过程中，团队碰到的问题通常都很少有文字记载，所以每一个受挫的团队都认为，自己遇到的问题比较特殊，而这些理念无法在他们的“现实世界”里发挥作用。只需听他们述说短短五分钟，我就能猜中三四个他们碰到的最大问题，这让他们觉得惊讶。而当他们得知其他团队也有同样的问题时，他们更是完全惊呆了。

如果你也在这样的团队当中，那么本书为你做的第一件事情，将是告诉你“你并不孤单”。本书中我所采访的那些团队并不完美——他们也曾遇到数不清的问题。但他们在碰壁之后，并没有放弃，而是决定围绕这些问题继续努力并解决问题。了解这一点通常能鼓舞人们换一种眼光去看待自己的问题。我希望你在读罢本书后也有同样的感受。

如果你正在实施实例化需求说明，本书将就如何解决你当前的问题提供非常有用的建议，同时也能让你了解未来会发生什么事情。我希望你可以从别人的错误中汲取经验，并且完全避免发生同样的问题。

本书也写给有经验的实践者，以及那些在实施实例化需求说明的过程中相对成功的人们。在采访开始之初，我本以为这些事情都已胸有成竹，只是在求证而已。结果我发现，人们在实施过程中居然有如此之多的想法，这是我始料未及的。我从这些案例中学到了很多，希望你也如此。这里所描述的实践和想法，应该会激发你的灵感，促使你对自己的问题尝试变通方案，或者让你

① 关于工程学的严谨有助于软件开发的错觉（如同二流的物理学分支），可参见http://www.semat.org。想更多了解对此错觉的反击，请参考Glenn Vanderburg的演讲“软件工程没用！”（http://confreaks.net/videos/282-lsrc2010-real-software-engineering）。

在见过类似的故事之后，意识到可以如何改善团队的过程。

本书内容

在第一部分，我会介绍实例化需求说明。我不会说服你为什么应该遵循本书描述的原则，而是向你展示——用实例化需求说明的方式——团队从这个过程中获益的例子。如果你在考虑购买这本书，请浏览一下第1章，看看有哪些好处可以带到你的项目中。第2章介绍了关键的过程模式和实例化需求说明的关键工件（artifact）。在第3章，我会更详细地解释活文档的概念。第4章会展示一些改变过程和团队文化的最常见的切入点，也会就开始过程实施时需要注意的地方给出一些建议。

本书写作的一个目的是为团队在实施实例化需求说明时使用的这些模式、想法和工件创建一致的语言。整个实践在业界有许多名称，里面各种要素的名称更是多出一倍。不同的人分别将同一个东西叫作功能文档、故事测试、BDD文档、验收测试等。正因为如此，在第2章中我会对所有的关键要素介绍一些我感觉不错的名字。即使你非常有经验了，我还是建议阅读这1章，以确保我们对本书中的关键名字、用词和模式的理解是一样的。

在第二部分，我会展示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。不同环境中的团队会做非常不同的事，有时甚至为了达到相同效果采取相反或冲突的措施。除了实践外，我还记录了团队贯彻基本原则的环境。第二部分差不多按照过程区域分成7章。

在软件领域没有最佳实践，但是确实有一些好的想法我们可以尝试在不同的环境中去使用。在第二部分中，有些小节标题旁边会有拇指向上或向下的图标，代表的是调查中一些团队觉得有用的做法，或者是他们经常遇到的问题。请根据建议做适当的尝试或回避，但不要完全照搬套用。箭头图标出现的地方代表的是各种做法的精髓。

软件开发不是静态的——团队和环境都在改变，过程也必须随着改变。在第三部分中，案例分析展示了一些团队的实施历程。我记录了他们的过程、约束条件和环境，并分析了这些过程是如何演化的。这些故事有助于你迈开第一步或让你更进一步，并发现新的想法与做事方式。

本书的最后一章，我总结了我从促成本书的案例分析中学到的关键要素。

更上一层楼

在传统的学习模型守破离（Shu-ha-ri）[①]中，本书处于破的层次。破是说要打破陈旧的规则，并证明成功的模型有很多。在我的*Bridging the Communication Gap*一书中，我展示了我的模型及经验。本书中，我尽量不带进以前的背景。只有当我觉得有重要观点需要证明，并且本书中提到的其他任何团队都没有类似情况的时候，我才会展示那些我自己参与过的项目。从这个意义上讲，

① “守破离”是来自于“合气道”（日本的一种自卫拳术）招式的学习模型。它包含三个层次。第一层“守”，学员必须严格学习一种招式。第二层“破”，学员知道除了他所学的招式外还有很多招式。第三层“离”，学员脱离招式的束缚。

本书在*Bridging the Communication Gap*的基础上更进了一步。

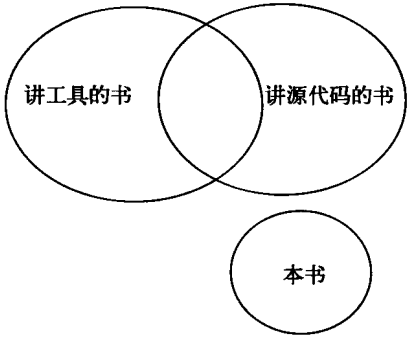
我会在第2章简单介绍一些基本的原则。即使你以前从未听说过这些想法，第2章的简介也应该可以给你足够的信息去理解本书的其余部分，但我不会过多地深入基础的内容。有关实例化需求说明的基础内容在*Bridging the Communication Gap*一书中有详细的描述，我不想在本书中重复。

如果你想更详尽地重温那些基础内容，请访问http://specificationbyexample.com，登记你购买了本书，就可免费获得*Bridging the Communication Gap*的PDF版本。

我想今后我不会就这一主题续写“离”这个层次的书籍——因为该层次是超越书籍的。另一方面，我相信本书可以帮助你到达“离”这一层次。一旦你开始觉得选择什么工具已经无关紧要，那么你就已经达到了这个层次。

本书没有源代码，也不介绍任何工具

本书没有源代码，也没有特定工具的使用说明。我觉得必须事先说明这一点，因为在出版过程中，我就曾多次向别人解释这一点（典型的问题有“什么意思？一本没有源代码的软件开发书？这怎么可能！”）。



实例化需求说明的原则和实践主要影响软件交付团队中的人员沟通，以及他们如何同使用者和项目干系人进行协作。我确信许多工具供应商会试图卖给你一套技术方案。如果有工具可以立即消除遇到的问题，许多经理会乐于为此买单。不幸的是，他们遇到的主要是人的问题，而不是技术问题。

比尔·盖茨说过：“在企业中应用任何一项技术时，首要的法则是，在有效率的系统中导入自动化，将使效率倍增。第二条法则是，在缺乏效率的系统中导入自动化，会使效率更低下。”很多团队在使用实例化需求说明的时候失败了，他们使用自动化工具反而导致他们的过程更加低效。我不想把注意力放在特定的工具上，相反，我想侧重分析团队努力实现这些想法的真实原因。一旦你们能正确地沟通和协作，你们就可以选择适合的工具去使用。在阅读本书后，如果你想知道更多支持实例化需求说明的工具，请访问http://specificationbyexample.com并查看资源部分。

谈谈术语

如果这是你首次听说实例化需求说明、验收测试驱动开发、敏捷验收测试、行为驱动开发，或者人们为这类做法所起的任何其他名字，你应该庆幸自己没有被这些误导性的名字所困扰。你应该放轻松些，而且你可以跳过这个部分。如果你已经接触过那些做法，我在本书中使用的名字可能会让你感到惊讶。请接着读下去，这样你就能理解为什么我使用这些名字，并且你也应该开始使用它们。

在我编写本书的时候，我也遇到了实践者们在编写他们的自动化需求说明时经常遇到的问题。术语应该要一致，这样才能易于理解，当把内容编写成文时很有必要明白这一点。本书是一系列访问的产物，很多我交谈过的人使用不同的名字来指称同一件事情，这样的话，要想保持所讲故事的一致就是相当困难的。

我意识到，实例化需求说明的实践者，包括我自己，通常会因为使用技术术语，导致我们自己以及其他尝试实施这些实践的人都很迷惑，这让我们感到内疚。因此我决定，编写本书的其中一个目标，就是要改变社区中使用的术语。让业务人员更多地参与进来是这些实践的一个主要目标，为此我们必须使用适当的名字去描述那些正确的做法，不要再让人们感到困惑。

当我们编写需求说明时，这个教训是显而易见的。我们都知道应该要保持术语的一致性，避免使用具有误导性的术语。但当我们谈论过程的时候，我们没有那么做。例如，当我们在实例化需求说明的环境中说持续集成的时候，我们并不是说要运行集成测试。因此，为什么要使用这个术语，然后不得不给其他人解释验收测试与集成测试的不同？在我开始使用需求说明工作坊（specification workshop）这个名字来代表有关验收测试的集体会议前，很难说服业务人员去参加。一个简单的名字变更就解决了这个问题。通过使用更好的名字，我们可以避免许多毫无意义的讨论，马上就让大家走上正确的道路。

为什么使用“实例化需求说明”这个名字

首先我想解释一下，为什么我选择实例化需求说明作为这些实践的总称，而没有使用敏捷验收测试、行为驱动开发或者验收测试驱动开发。

在2010年的伦敦领域驱动开发交流大会[①]上，Eric Evans跟别人争论，说敏捷作为一个术语已经失去了一切意义，因为现在什么都可以称为敏捷。很不幸的是，他是正确的。尽管有大量的著作讲如何正确地实施极限编程、Scrum以及其他不那么流行的敏捷过程，但我见过太多太多的团队，试图去实现冠以敏捷一词的过程，但那些过程又显而易见地违背了敏捷的精神。

为了避免这种关于敏捷是否可行（以及什么是敏捷）的无意义争论，在本书中，我尽量避免使用敏捷这一术语。只有当我提到的团队基于敏捷宣言概括的原则定义了良好的过程，并开始实施时，我才会使用它。由于不会经常提到敏捷这一术语，敏捷验收测试这个名称也就无从谈起了。

① http://skillsmatter.com/event/design-architecture/ddd-exchange-2010

这里描述的实践没有形成一个成熟的软件开发方法论。它们可以补充其他方法论——无论是基于迭代还是基于工作流的——使需求说明和测试更加严谨，增强不同项目干系人和软件开发团队成员之间的沟通、减少不必要的返工，并让改变更加容易。因此我不想使用任何“驱动开发”之类的名字，尤其不会使用行为驱动开发（BDD）的字眼。不要认为这说明我反对BDD。恰恰相反，我喜欢BDD，而且我认为本书实际上主要在讲BDD的核心内容。但BDD同样有名字歧义的问题。

BDD到底代表了什么总是在变化。关于什么是BDD，什么不是BDD，Dan North是最具话语权的。在Agile Specifications, BDD, and Testing Exchange 2009[①]上，他说BDD是一种方法论。（事实上，他将其称为“第二代的、由外而内的、基于拉动的、多项目干系人、多尺度的、高度自动化的敏捷方法。”）为了避免在North所说的BDD和我理解中的BDD之间产生任何混淆和歧义，我不想使用这个名字。本书讲的是一组宝贵的实践，在很多方法论中你都可以使用，包括BDD（如果你能接受BDD是一种方法论的说法）。

我也想尽量避免使用测试这个字眼。很不幸地，很多经理和业务人员认为测试是一种技术辅助活动，不是他们想参与的事情。毕竟，他们有专门的测试人员去处理这件事情。实例化需求说明要求项目干系人以及交付团队的成员（包括测试人员、开发人员、业务分析人员）积极地参与进去。只要我们在标题中不放入测试这样的词汇，那么故事测试、敏捷验收测试以及其他类似的名字自然就不会出现。

这让实例化需求说明成为了最有意义的名字，它的负面影响最小。

过程模式

实例化需求说明由一些过程模式（Process Pattern）组成，后者是更广义的软件开发生命周期的组成要素。本书中我用的名字是在英国敏捷测试用户组会议、敏捷联盟功能测试工具邮件组以及工作坊中经过一系列讨论得到的。其中有些名字已经用了一段时间，而另一些大多数读者还比较陌生。

业内流行用一个实践或工具的名字来描述过程的一部分。功能注入（Feature Injection）就是一个很好的例子——用它来描述从商业目标中获取项目范围这一过程，就很受欢迎。但是功能注入只是其中一种技术，还有其他方法可以达到同样的目的。为了讨论不同的团队在不同的环境中做什么，我们需要更宏观的概念来囊括所有这些实践。一个好的名字能很好地说明预期的结果，并且清晰地指出这些实践的关键区别。

以功能注入与类似实践为例，其结果就是项目或里程碑的一个范畴。与其他定义范围的方法相比，关键区别在于，我们专注的是商业目标。因此我提出从目标中获取范围（deriving scope from goals）这一概念。

在实例化需求说明过程中，团队碰到的最大的一个问题是：谁应该在什么时候写些什么东西。

① http://skillsmatter.com/podcast/java-jee/how-to-sell-bdd-to-the-business

所以我们需要一个好名字来明确地说明大家都应参与（需要在团队开始编码或测试前做），因为我们要使用验收测试作为开发的目标。测试先行（Test First）是个不错的技术名词，然而商业用户并不能理解，更何况它没有包含合作的意思。我建议我们关注通过协作制定需求说明（specifying collaboratively），而非测试优先或写验收测试。

听起来很普通，只是把每个数字上的可能性考虑进自动化功能测试里。如果自动化了，我们为什么不这么做？但是如此复杂的测试作为沟通工具是无法使用的，在实例化需求说明里，我们是要用测试来沟通。因此不是编写功能测试，而是关注举例说明（illustrating using examples），并且期望它能输出关键实例（key examples）这些实例表明我们只需要恰当地描述所需的环境就可以了[①]。

关键实例是原料，但是如果仅仅关注验收测试，为什么不干脆就用50列100行的复杂表格来作为实例，并且不带任何说明？反正是机器来测试。用了实例化需求说明，测试既是给机器看的，也是给人看的。我们需要说清楚的是，使用实例说明后，还有一个步骤，就是抽取属性和实例的最小集合来说明业务规则，并加上标题和描述等。我建议把这个步骤称为提炼需求说明（refining the specification）[②]。

提炼的结果既是需求说明，同时也是开发的目标、检查验收的客观方法以及之后的功能回归测试。我不想叫它验收测试的原因是，它很难说明为什么文档要用领域语言来写、可读性要高，还要容易理解。我认为我们应该将提炼的结果称为带实例的需求说明（specification with examples），从而揭示出一个事实，就是需求说明需要基于实例，但绝不是只包含原始数据。将这个工件称作需求说明可以明显地指出大家都需要关注它，而且它需要容易理解。除此之外，关于是否用这些检查来自动验收软件或自动拒绝不满足需要的代码，还有另外一种截然不同的观点[③]。

我不想再花时间和那些买了QTP的人争论，告诉他们QTP对验收测试完全没有用。只要我们谈论测试自动化，总有人推销测试人员已经用来做自动化的可怕玩意儿。敏捷验收测试和BDD工具与QTP之类的工具有很大的差别，它们处理完全不同的问题。不应将需求说明解释成一种自动化的技术。我们把在不歪曲任何信息的情况下将验证自动化称作自动化验证而不改变需求说明（automating validation without changing specifications），而非验证自动化。不改变原有需求说明的自动化验证可以帮助我们避免可怕的脚本和在测试需求说明中直接使用技术类库。可执行的需求说明应该与在白板上看到的一样，而不应该转译成Seleninum命令。

在将需求说明验证自动化后，我们可以用它来验证系统。事实上，我们得到了可执行的需求说明（executable specification）。

我们要频繁地检查所有的需求说明，确保系统还在按所期望的运行，同样重要的是确保需求说明还能描述系统的行为。如果我们将这称作回归测试，那么很难向测试人员解释为什么他们不应该在之前既小又好而且明确的需求说明中再加500万个测试用例。如果再提到持续集成，我们

① 谢谢David Evans给的建议。

② 谢谢Elisabeth Hendrickson建议的这个名字。

③ http://www.developsense.com/blog/2010/08/acceptance-tests-lets-change-the-title-too

将很难解释为什么不总是端到端地运行这些测试，并检查整个系统。对于一些遗留系统，我们需要针对一个部署好的正在运行中的环境来运行验收测试。技术上的集成测试应在部署前运行。所以我们不谈回归测试或持续集成，我们谈*频繁验证*（validating frequently）。

实例化需求说明具有长期回报，前提是拥有一个对系统自身功能的引用，该引用具有与代码一样的价值，但是更易读懂。长期来说，这使得开发有效率得多，能促进与商业用户的合作，促成软件设计和商业模型的一致，并且使大家工作更简单。但是要做到这点，该引用必须有意义，必须有人维护，而且还必须内部保持一致，并与代码保持一致。我们不应该有大量的测试还在用数年前使用的术语。你很难让忙碌的团队回过头去更新测试，但是在重大改变之后回头去更新文档，这是大家所愿意看到的。所以我们不要关注那些含有数百个测试的文件夹，让我们把注意力放到*演化活文档系统*（evolving a living documentation system）上。这样就更容易解释，为什么很多事情必须不言而喻，为什么商业用户也需要使用这些，为什么需要良好地组织以方便查找。

所以情况就是这样：我选择了这些名字，不是因为它们以前很流行，而是因为它们有意义。这些过程模式的名字必须创建一种思考模型，该模型可以明确地指出那些重要的事，并且减少困惑。我希望你能明白这点，并接受这个新的术语。

目　录

第一部分　开始