用例故事胜过用户故事

(Use Case over User Story)

——

评 Mike Cohn 的 Advantages of User Stories for Requirements

1.0

张恂

www.zhangxun.com

2013-7-1

在 2004 年十月发表于 Informit Network 上的 Advantages of User Stories for Requirements 这篇著名文章中，敏捷大师 Mike Cohn 分析和列举了用户故事的主要优点。然而，我们并不赞同他认为用户故事比用例故事（Use Case）更好、更适合敏捷开发的观点。下面是我们对这篇文章的评论、分析与反驳。

定义

Cohn 对用户故事的定义是：

Extreme programming (XP) introduced the practice of expressing requirements in the form of user stories, short descriptions of functionality–told from the perspective of a user–that are valuable to either a user of the software or the customer of the software.

用户故事是 XP（极限编程）的核心需求技术，采用简短的形式，从用户的角度来描述对用户或客户有价值的系统功能(functionality)。

（用户故事能描述非功能需求吗？）

XP 大师 Ron Jeffries 建议每个用户故事应该由“卡片、对话与确认”三部分（即 3C: Card, Conversation, Confirmation）组成：

1.Written description of the story, used for planning and as a reminder.

2.Conversations about the story that serve to flesh out the details of the story.

3.Tests that convey and document details that can be used to determine when a story is complete.

Because user story descriptions are traditionally handwritten on paper note cards, Ron Jeffries has named these three aspects with the wonderful alliteration of card, conversation, and confirmation.

第 1 句，提到了用户故事的两个基本功能：用于做计划，以及需求的提示物；

（这里为啥没提需求的描述与分析？）

第 2 句，说明一个用户故事的具体需求细节应该通过用户和开发人员之间的对话交流来填补；

第 3 句，说明除口头沟通之外，更多的故事细节，应由大量的测试来表达和记载，并以此判断一个故事的需求描述是否完整。

从以上用户故事的定义和组成看，与用例有啥异同呢？

用例故事主要是 UP（统一过程）采用的需求技术。其实，一个用例至少也由三个部分组成：用例描述（文本与图形），用户与开发人员的对话交流，以及完整的测试等。

在对后两点（用户对话与确认测试）的要求上，用户故事与用例故事其实没啥本质区别，而且对于第一点，用例同样可以用于项目计划和作为需求的提示物。

两者的区别主要在于第一点。同样是文字描述，用例的需求描述内容通常比用户故事要更加丰富和完善，除了一段话的简述，还有前后态、触发事件、基本流、扩展条件与扩展流、业务规则、非功能需求等等字段。由于用例文本提供了更加丰富和结构化的需求描述信息，对于提高后两点（用户交流、测试编写）的质量和效率反而有更大的帮助。

此外，用例方法还建议结合采用 UML 用例图、活动图、序列图、状态图等等直观生动的可视化模型来描述复杂的系统需求，在需求描述与分析的技术手段上可谓更加完备。

优点

与用例故事和传统需求技术相比，Cohn 认为用户故事的不同之处和优点主要是以下三点：

1）强调口头交流，比文本描述更准确；

2）更适用于项目计划和估算；

3）简单易学，更有利于迅速启动敏捷开发。

那么用户故事的这三个优点是否就是用例的缺点或不足之处？答案是否定的。

优点 1、用户故事更强调口头交流，可消除文本需求的多义性。

User stories emphasize verbal communication. Written language is often very imprecise, and there's no guarantee that a customer and developer will interpret a statement in the same way... We act as though written words are precise, yet they often aren't.

以下我们分两个部分来反驳。

1a. 用户故事比用例故事更适合口头交流？

Cohn 说由于用户故事更强调口头交流，所以比用例更有优势，这个理由并不充分。根据我十多年的行业企业用例需求教学经验，其实用例文本比用户故事能更好地促进用户和开发者之间的沟通。

我们应该拿什么、怎样去和客户沟通需求？是用一种简单的、只有寥寥几笔的故事卡片好呢，还是用有更多系统、规范的书面细节和直观、生动图形展示的用例模型更好？

用例方法提供了规范的需求模板和编写规则，有了这些结构化、类程序的、详细的模板字段、规则和技巧建议（如下图所示），开发者和客户就能有的放矢，更加系统、准确、有条理地，更好地进行沟通，从而达成一致意见，消除模糊性，促进系统需求的尽快稳定和收敛。

在促进用户需求的有效沟通方面，与用例故事所具备的文本、图形等丰富描述手段相比，用户故事卡片所能提供的建议内容太少。

事实是，有了更加完善的需求模板和内容脚本的支持，文本、图形描述加口头沟通，用例故事比用户故事能更加有效地提高开发者与用户之间交流需求的质量和效率。

用例 下订单

简述

顾客通过网店订购宠物，下订单（包括商品的种类和数量），采用送货、信用卡支付方式。

范围 宠物店系统（JPetStore） **层次** ！

后态

- 保持用户状态为已登录。
- 保存了订单（包含用户名称、订单日期、商品名称、编号、数量、价格；用户信用卡信息、账单地址和送货地址等）。
- 修改了商品存货（可售）数量。
- 清空了购物车。

前态

用户已进入网店页面，可访问到购物车。

触发事件

a. 用户选择把商品“加入购物车”（Action:Add to cart）

b. 用户直接打开购物车

基本流

CALL(使用购物车)

检查购物车 {

1. 用户选择结账（Action:Checkout）。（PAGE:Shopping Cart）
2. 系统检查购物车。
3. 如用户尚未登录，CALL(登录)。
4. ASSUME(购物车非空 AND 用户已登录)
5. 系统显示购物车汇总信息（含总价）让用户确认。（PAGE:Checkout Summary）
6. 用户选择继续。

}

填写交易信息 { //填写订单

1. 系统读取并显示用户缺省账单地址和送货地址（来自注册信息），提供信用卡表

单供用户填写卡号、类型和密码。（PAGE:New Order）

2. 填写支付方式

用户填写信用卡信息，并确认送货和账单地址后提交。

3. 验证

系统验证用户提交的交易信息（包括地址、信用卡等）。

4. 系统显示完整的交易信息（包括用户已填写的信用卡信息、送货地址和账单地址等）让用户确认。（PAGE:Confirm Order）

5. 用户选择继续。

}

处理订单 {

1. 创建订单

系统为用户创建一张新订单，并导入购物车中的内容。

2. 保存订单

系统分配唯一的新订单号，并在该用户名下保存该订单，以便今后查询。

3. 更新存货数量

系统根据订单中的所有条目（LineItem），修改已订购商品的存货数量。

4. 系统清空购物车。

5. 系统显示订单成功的完整确认信息。（PAGE:View Order）

}

END

扩展流

:[用户继续挑选商品] { ABORT }

:检查购物车 [购物车为空] {…}

:填写交易信息/验证 {

[用户要求变更送货地址] {

1. 系统显示空白的送货地址表单。（PAGE:ShippingForm）

2. 用户输入新送货地址并提交。

RESUME(0) //重新执行父步骤

}

[信用卡信息无效] {

```
      1.    系统显示信用卡的错误提示信息。
      RETURN(../填写支付方式)
   }

   [账单地址无效 Or 送货地址无效] {
      系统提示用户重新输入地址。
      RETURN(../BEGIN)
   }
}//验证

:处理订单 {
   [系统处理失败] {
      1.    系统保留|恢复购物车内容。
      2.    系统显示订单处理失败原因。//…
      3.    系统让用户选择处理办法（重新尝试，或放弃购买）。
      4.    用户选择重新尝试。
      5.    GOTO(/检查购物车)
      } EXTENSION {
         :4[用户选择放弃] {
            1.    系统清空购物车。
            2.    系统显示本次购物失败。
            ABORT
      }
   }

   [用户取消处理] {…}
} //处理订单

:*[超时 & 已登录] {
   系统清除购物车和用户的其他会话信息，自动注销。
   ABORT
}

…
```

用 UUCM/UCL 模板编写的“下订单”用例

1b. 口头交流比文字描述需求更准确？

Cohn 说，文本语言往往很不准确，我们无法永远保证客户和开发者对一个句子有相同的理解。这么说有点片面和夸大。

的确基于自然语言的文本描述很可能不准确，往往存在多义性。所以，需求文本需要正式的用户评审，通过严格的模型分析、测试、演示和实际使用等多种软件工程手段来验证需求，保证开发的系统确实符合用户的需要（The right thing）。

然而，难道口头交流需求就比文本描述需求更准确，就不存在表达上的多义性？

其实表意不准的问题，口头交流也存在。而软件工程的实践表明，需求口头约定比文本约定隐匿的风险和问题更大，而规范的文字交流往往比口头交流更可靠、更准确。

造成需求描述不准确的原因，主要并不在于沟通形式采用的是口头还是文字，口语还是书面语，关键是因为人们采用了自然语言！人类语言的天生缺陷，会造成表意不精准，存在多义性，这是一种客观现象。

如何才能尽量避免采用自然语言描述需求的天生缺点？答案是：用更加科学、系统和规范的办法，比如形式化/半形式化等方法。

形式化或半形式化的需求描述方法通常比口头交流需求更精准。而用例模板语言正是一种介于完全形式化和非结构化自然语言之间的一种半形式化描述方法，在追求需求语义的精准性和可读性之间作出了更好的权衡。

俗话说，细节决定成败。舍弃详细的书面需求文档和高质量的需求模型，只提倡基于简略的用户故事卡片进行口头沟通和编写完备的测试（测试能否取代需求模型？），对于国内外的许多软件开发项目是一个很大的风险隐患和质量缺陷。

1c. 小结

基于规范的文本描述的用例需求模型，比强调口头交流的用户故事卡片，内容更准确，细节更丰富，格式更完备，能更好地促进与客户沟通需求，并且有效提高测试编写的效率和质量。

优点 2、用户故事更适用于项目计划

A second advantage of user stories is that they can be used readily in project planning. User stories are written so that each can be given an estimate of how difficult or time–consuming it will be to develop; use cases, on the other hand, are generally too large to be given useful estimates.

Also, a story is implemented all in a single iteration of an agile project, while it's common to split a use case across multiple iterations (even though those iterations are usually longer than on a story–driven project).

以下我们分两个部分来反驳。

2a. 用例不适合项目计划和估算？

Cohn 认为用户故事通常比用例的粒度更小，可以在一个迭代中实现，所以做项目计划时用用户故事来估算、排序更方便。那么，是否用例就不适合做项目计划和估算了？

非也。

每一个用例都是一个需求故事。用例故事与用户故事的关系，就像一棵大树的树干与树枝，树枝与树叶的关系。用例充当容器把各种琐碎的小用户故事拼接起来，可以让我们看到整个系统需求的全貌。

每一个用例都是对用户真正有价值（反映了用户目标）的需求单元，其粒度通常比用户故事大，这导致用例故事的总数往往比用户故

事少，因而更便于我们在做项目计划时把握系统需求的全局，而不是只见树叶不见森林。

基于用例图的用例分析技术并不是传统意义上的功能分解，而是需求或功能的聚合，可以让我们站在大量琐碎、关系错综复杂的功能细节之上，看清少数真正具有业务价值的关键用户目标（User Goal）和系统服务。

合理的项目计划既需要大粒度的需求，也需要小粒度的需求。而用粒度更大、数量更少的用例来做项目的发布计划，效率比普通的用户故事更高。

Cohn 还说“Use cases are generally too large to be given useful estimates”，意思是说因为用例太大了，所以通常不能给出有用的估算。果真如此？

非也。

Cohn 忽略了用例的切分（又叫分解、拆分）技术。用例估算的正确做法是：

分而治之，把粒度过大的用例分解为更小的需求单元进行估算。

用例的分解可以采用多种方式，既可以分解为用户故事，也可以分解为情节/场景（Scenario）、块（Block）、特性（Feature）等等。通常把一个用例所包含的所有需求块的估算结果累加起来，就可以得到这个用例的估算（工作量）。

在利用粒度更小的需求单元做计划与估算这点上，用例故事与用户故事这两种方法的基本主张和做法其实是一致的。

两者的区别主要在于：在需求细节的内容描述上，用例故事比用户故事的要求更多。用例模板提供了更多的需求细节（前后态、触发事件、基本流、扩展流、业务规则等等），从而有利于我们得出比忽略细节的用户故事更为准确、避免过于乐观的估算。

2b. 用户故事更适合估算？

Cohn 说“User stories are written so that each can be given an estimate of how difficult or time‐consuming it will be to develop”。意思是，针对每个用户故事（相比用例）更容易估算出它的开发难度和用时。然而仅靠如此简单的一张故事卡片，这么一点信息，得出的估算可靠吗？

估算的一个基本规律是，掌握的有效信息越多，估算的准确性通常就越高。

用户故事估算的主要缺点在于，由于故事卡片本身缺少大量的需求细节信息，缺乏需求的完整性，很容易让人忽视各种特殊情况的需求和潜在的技术风险，易造成估算不准，导致预测的结果往往偏乐观。

有人可能会辩解说，用户故事估算当然不是仅仅靠故事卡片，提高故事估算可靠性的办法是：与用户口头交流，以及针对每个故事编写全面的测试，三者结合（即上文提到的 3C）就能获得更可靠的估算。这有一定道理。

然而前面我们讨论过，与单纯靠简略的故事卡片相比，其实借助文本用例模板、UML 图形等手段更有利于促进需求的口头交流，同时有了大量用例的扩展流、特殊情况等重要需求细节信息的书面记载，也便于我们写出更完善的测例（Test Case）。从需求用例到测例，是一个很自然的系统思考分析过程。

此外，即便在用户故事估算的过程中，适当编写每个关键用户故事所对应的文本用例（前后态、基本流、扩展流等），也更有利于提高估算的整体可靠性。

因为粒度小，所以更适合估算，这并非用户故事的专利。虽然一个用例的粒度通常比用户故事大，但是用例可以分解为与用户故事粒度基本相当的更小需求单元（如 Cockburn 方法中的“子功能用例”或 Taij 方法中的“小用例”），因而同样适合估算。

一个有趣的问题是，用户故事方法为什么始终要回避、忽略书面记录重要的需求细节信息这件事情呢？

2c. 小结

由于提供了更高质量的需求细节信息，用例故事比用户故事更适合做项目计划和估算。

用户故事的粒度通常比用例小，这并非它特有的优势，因为用例同样可以分解成包括用户故事在内的，其他更小的合适的需求单元，

以用于项目计划和估算。

优点 3、用户故事更有利于团队推迟收集需求细节，迅速启动开发。

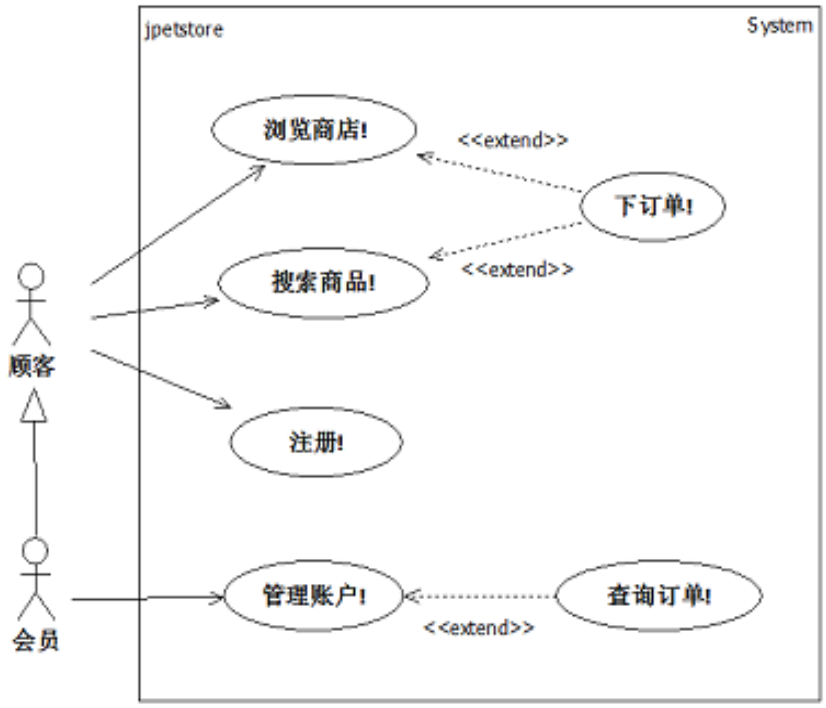
Cohn 认为，用户故事方法鼓励团队推迟收集需求的细节，先用具有占位器（Placeholder）功能的目标层故事描述简略的需求，迅速投入开发，然后在适当、必要的时候再补充更多的故事和需求细节，因此用户故事完美地（perfectly）适合进度紧张的项目。

User stories encourage the team to defer collecting details. An initial place–holding goal–level story (“A Recruiter can post a new job opening”) can be written and then replaced with more detailed stories once it becomes important to have the details.

This technique makes user stories perfect for time–constrained projects. A team can very quickly write a few dozen stories to give them an overall feel for the system. They can then plunge into the details on a few of the stories and can be coding much sooner than a team that feels compelled to complete an IEEE 830–style software requirements specification.

这些说法有一定道理，但可惜在这里 Cohn 没有提及 UML 用例图以及用户目标层的用例。

用例图



用户故事

1）作为顾客，我要浏览商店，可以查看各类商品的说明、图片和价格。
2）作为顾客，我要注册，以便成为网站会员（可以下订单）。
3）作为顾客，我要通过关键词搜索任意商品。
4）作为顾客，我要下订单（含商品种类、数量、价格）。
5）作为会员，我要管理账户，以便修改个人资料（密码、地址、信用卡信息等）。
6）作为会员，我要查询历史订单。

© 张恒 www.zhangxun.com

在以上宠物店的用例模型中，我们仅用一张 UML 用例图就直观、形象地勾勒出了系统的 6 个主要功能，以及用户与需求、需求与需求、用户与用户之间的多种关系，而采用用户故事描述这些需求至少需要 6 张独立的卡片，不仅文字上有冗余，而且各个需求之间的关系也无法直观表达。

其实，画用例图比书写用户故事卡片更简便，用例图中的每个目标层用例也同样起到了需求讨论占位器的作用，而用例（边界）图更

好地反映了系统功能的全局视图。所以，用例同样可以，甚至比用户故事更好地促进进度紧张项目的敏捷开发。

事实是：

用例（图）比用户故事更简单、更敏捷，能更好地促进团队推迟描述需求的细节，迅速启动开发。此外，用例图还直观地提供了系统需求的全局、局部与关系视图，这是用户故事方法所缺少的。

比较

Cohn 认为用户故事与用例故事的区别主要有 4 点：

粒度不同，完整性不同，生命期不同，以及用途不同。

1、粒度不同

One of the most obvious differences between stories and use cases is their scope. Both are sized to deliver business value, but stories are kept smaller in scope because we place constraints on their size (such as “no story can be expected to take more than 10 days of development work”) so that they can be used in scheduling work. A use case almost always covers a much larger scope than a story.

一个用例的粒度通常比用户故事大。

为了用于迭代计划，一个用户故事的大小通常应该控制在一次迭代内完成。而用例为了要能完整、准确地反映真实的用户目标和需求，粒度通常要比用户故事大，实现一个用例（大粒度需求）往往需要几个迭代的时间。当然，有些小粒度的用例也可以像用户故事那样在一个迭代内完成。

这说明，一个用例通常可以分解为多个用户故事，而几个相关的小用户故事合起来就可以是一个用例（大故事）。

此外，用例故事的分解其实有多种灵活的方式，除了可以分解为用户故事，还可以拆分成多个特性、用例片段、包含或扩展子用例、块、情节等等。

2、完整性不同

User stories and use cases also differ in the level of completeness. James Grenning has noted that the text on a story card plus acceptance tests "are basically the same thing as a use case." By this, Grenning means that the story corresponds to the use case's main success scenario, and that the story's tests correspond to the extensions of the use case.

敏捷专家 James Grenning 认为：一个用户故事加上它的验收测试（包含许多针对扩展和异常情况的测例），基本上就等价于一个用例；一个用户故事就相当于一个用例的基本流，而它的验收测试则相当于这个用例的扩展流。

这种说法基本准确。这说明用户故事（卡片）本身不是完整的需求描述，它缺少很多必要、复杂的需求细节，这也是为什么 Ron Jeffries 提出一个用户故事至少需要 3C（Card, Conversation, Confirmation）才是完整的，而这里的 Confirmation 相当于 Grenning 所说的验收测试。

3、生命期不同

Another important difference between use cases and stories is their longevity. Use cases are often permanent artifacts that continue to exist as long as the product is under active development or maintenance.

Stories, on the other hand, are not intended to outlive the iteration in which they're added to the software. While it's possible to archive story cards, many teams simply rip them up.

Cohn 强调用例通常是一种“永久性”工件，其生命期几乎与一个产品的开发期与维护期一样长。而一个用户故事的生命期，却通常不超过一个迭代，功能实现了，用完就可以扔掉，起到的作用是临时性的。

由此可见，作为企业的一种需求工件和知识资产，用例故事和用户故事哪个更重要？不言而喻。

4、用途不同

Another difference is that use cases and stories are written for different purposes.

Use cases are written in a format acceptable to both customers and developers so that each may read and agree to the use case. The purpose of the use case is to document an agreement between the customer and the development team.

Stories, on the other hand, are written to facilitate release and iteration planning, and to serve as placeholders for conversations about the users' detailed needs.

编写用户故事的目的主要是用于发布和迭代计划，以及用作启发、细化用户需求的对话和讨论的占位器。

用例故事则采用了一种客户与开发者都能接受、便于阅读的格式，其内容本身反映了客户与开发团队之间达成的一种需求契约。

看了这些，有些人可能会产生误解，以为用作项目计划和需求占位器是用户故事的特长，用例不能用于项目的发布和迭代计划，也不能用作与用户讨论需求细节的占位器。前面我们已经讨论过，事实并非如此。

小结

关于用户故事与用例故事的这 4 点不同之处，Cohn 分析的结论比较准确。然而，粒度不同、完整性不同、生命期不同以及用途不同，恰好说明了用例是比用户故事更重要、更完整、更强大的一种需求技术。

用例简述

Cohn 还介绍了用户故事与用例简述的区别：

Use case briefs differ from user stories in two ways.

First, since a use case brief must still cover the same scope as a use case, the scope of a use case brief is usually larger than the scope of a user story. That is, one use case brief will typically tell more than one story.

Second, use case briefs are intended to live on for the life of a product. User stories, on the other hand, are discarded after use.

Finally, use cases are generally written as the result of an analysis activity, while user stories are written as notes that can be used to initiate analysis conversations.

Cohn 认为区别主要是三点：

1）用例简述的范围通常比用户故事大，内容更多。

没错，一个用例简述描述内容的粒度大体与用例相同，通常包含或可分解为多个更小的步骤（如用户故事）。

2）用例简述的生命期通常跨越一个产品的整个生命期，而用户故事通常用完后就扔了。

对，用例的文档记录功能其实是个优点，不应随便丢弃需求的描述。

3）用例通常是需求分析的结果，而编写用户故事的一个主要目的是用做启动需求分析与对话的提示记录。

这说明了两者的时序差，一前一后，用户故事（卡片）可用来启动需求分析的对话，而用例描述记载了需求分析的结果，直接体现了需求分析的内容。但启动需求分析也未必一定要通过用户故事，比如可以用 UML 用例图或用例简述。

使用误区

Cohn 还提到了用例使用中的几个问题和误区。

First, use cases often lead to a large volume of paper, and without another suitable place to put user interface requirements, they end up in the use cases.

Second, use case writers focus too early on the software implementation rather than on business goals.

在功能需求描述中混入用户界面需求或软件实现信息，这两个问题其实是用例应用中常见的缺陷或错误，是使用者使用不当、对用例理解有误造成的，而并不是用例方法本身的缺点，正确的用例使用方法和技巧不建议这么做。同样，我们也不能把使用者对用户故事的误解和误用，当成是用户故事方法本身的问题。

核心用例

Cohn 还提到了 Constantine and Lockwood 的 Essential Use Case（EUC，核心用例）方法。EUC 也是用例方法的一个流派。

An essential use case is one that has been stripped of hidden assumptions about technology and implementation details...

What's interesting about essential use cases is that the user intentions could be directly interpreted as user stories.

EUC 的这两个特点（去掉技术和实现细节、反映用户意图等）其实已经被如今的 Cockburn、Jacobson、Bittner 等主流用例方法所吸收和采纳，包括在用例的主干部分去掉技术和实现细节，确保用例的步骤描述反映用户的真实目标和意图等。

需求陈述

Cohn 分析了传统的 IEEE 830 标准样式需求规范的不少缺点，例如没有聚焦用户目标。这点我赞成。

IEEE 830–style requirements have sent many projects astray because they focus attention on a checklist of requirements rather than on the user goals.

Cohn 强调了聚焦用户目标的重要性：

By focusing on the user’s goals for the new product, rather than a list of attributes of the new product, we can design a better solution to the user’s needs.

事实上，每个用例故事都反映了一个真正的用户目标，用户目标驱动开发正是用例分析的一大优点。

总结

以上我们从用户故事的定义、用户故事的优点、用户故事与用例故事的比较等几个方面，对“用户故事比用例更好，更适合敏捷开发”的观点进行了分析和反驳。

Mike Cohn 大师在这篇文章中没有谈及用户故事的缺点和用例的优点，没有谈及 Application Context，即用户故事在什么情况下适合，什么情况下不适合。这并不是一篇很严谨的科学论文，存在一些逻辑缺陷和认识上的误区，得出的许多结论和判断也是片面的，长期研究需求工程和软件需求的专家们很容易发现这些问题。

俗话说，细节决定成败。敏捷方法主要适用于复杂系统的软件开发，而复杂系统往往具有大量复杂的需求细节。用户故事（卡片）缺少足够的书面细节，传统做法是用完就扔，因此很难基于用户故事建立高质量的系统或产品需求模型。用户故事的粒度小、形式简略，所以主要用于敏捷项目的计划、估算和管理，但却不是一种成熟、完善的需求分析技术... 这些缺点对于国内外的很多项目和产品开发来说是一个较大的风险隐患。

20 多年的大量实践已证明，用例是迄今为止现代软件工程中最成熟的一种需求分析与管理技术。写好文本用例可以一举多劳、事半功倍，帮助开发团队更好地促进与用户沟通，建立质量更高的需求模型和测试模型，从而尽最大可能避免复杂软件系统设计、开发与测试中的大量折腾和浪费。

由于 XP 与 UP 两种方法流派的竞争，10 多年来在敏捷社区中形成了一种错觉，不少人以为需求 Story 就是用户故事。而事实上，用例作为一种需求故事的描述和分析技术手段比用户故事出现得更早，也更成熟。用户故事则从主流用例方法中学习借鉴了不少好东西（如业务价值、用户目标、粒度分层、聚焦意图等等），而且在市场营销上做得更好，取了一个更好听、更大众化的名字——用户故事。

需求故事至少有两种。用例故事和用户故事两种技术都可以描述系统功能需求，而每一个用户故事都有一个与其相对应的用例故事，两者之间存在着一定的等价、对应关系。除了适用情况、表现形式、内容粒度等方面有区别外，两者并无多大的本质区别，一般认为用户故事大体相当于简化、缩小版的用例，而且主要适用于小微项目或产品的开发。

用户故事是不完整的使用故事，用例故事是完整的使用故事！（User stories are incomplete use stories, while use cases are complete use stories.）

参考

Alistair Cockburn, Structuring use cases with goals, 1997.

Alistair Cockburn, Why I still use use cases, 2008-1.

Mike Cohn, Advantages of User Stories for Requirements, informit.com, 2004-10.

Mike Cohn, Advantages of User Stories for Requirements, MountainGoatSoftware, 2004-10.

作者简介

张恂，资深软件需求与敏捷开发专家，OU3（OOAD、UML、Use Case、UP）专家，15 年软件工程研发管理与咨询教练经验。

自 1998 年起开始研究和传播基于 Use Case 的需求分析与管理技术。2003 年起提出整合主要用例流派，用例、用户故事与特性等需求技术的“统一用例方法”。2006 年创作了用于指导 UML 和 Use Case 建模的“太极建模口诀”：由外而内，层次分明，动静结合，逐步求精。

十多年来服务的客户遍及国内通信、银行、保险、证券、税务、外贸、互联网、电子商务、企业管理等各种软件开发、系统集成和信息化领域，已为全国各地数百家知名软件开发机构和企事业单位的数千名学员传授了当代国际先进的软件研发管理方法与工程技术。

敏捷资源：

http://www.zhangxun.com/agile

用例资源：

http://www.zhangxun.com/usecase

欢迎参与本文讨论：

http://www.zhangxun.com/?ucoverus