第 7 章
两个故事

《诗经·小雅·鹤鸣》:“他山之石,可以攻玉。”

作为本书的最后一章,本章主要介绍了传统敏捷方法(如 Scrum+XP)所采用的用户故事技术,并对两个故事——用户故事与用例故事之间的异同做了剖析。

用户故事真的比用例故事更好吗?通过对两者的细致分析与比较,我们得出的结论是:

① 与用户故事一样,用例故事同样也适用于敏捷开发;

② 两个故事之间存在着偏等价性,即用例故事几乎可以取代用户故事,反之则不行。

7.1 用户故事简介

由极限编程(XP)方法的创始人 Kent Beck 所发明的用户故事(User Story),可以说是过去十多年来敏捷运动中知名度最高、应用最广泛的一项敏捷需求技术。它几乎是 Scrum+XP 开发过程中的一项标准实践,可用于辅助制定各种开发计划,以及促进各方为了澄清和细化需求而展开的对话等方面。

然而,业界对用户故事多年来也一直存在着不少误解。例如,认为用户故事的优点就是用例的缺点,敏捷开发只能用用户故事而不能用用例等,而且出于一些原因,坊间也不太愿意谈论用户故事的缺点。

为此,本章将对用户故事的优缺点进行比较深入的剖析,并详细地比较用户故事与用例之间的异同点,以此说明两者事实上具有一定的相似性或等价性,用例其实是比用户故事更成熟、更强大的需求技术。

下面让我们先从“什么是用户故事”开始吧。

Scrum与用户故事专家Mike Cohn对用户故事的定义是：

“极限编程首创了以用户故事的形式来表达需求的实践，它们从用户的视角，以一种简短的形式描述了对软件的用户（或客户）有价值的功能。”

简而言之，用户故事代表（描述）了对用户有价值的一个产品（或系统、软件）功能。

在Scrum＋XP开发过程中，典型的做法是把一个用户故事（通常只有简单的一两句话）以手写的方式记录在一张纸质小卡片（如索引卡、标签卡等）之上。

例如，以下是一个典型的用户故事（取自Cohn的著作《用户故事与敏捷方法》）：

故事卡1.1
用户可以在网站上发布简历。

其实，该用户故事对应于用例方法中主用角“用户”的用例“发布简历”。

用户故事（卡片）一般主要用于描述系统的功能需求，但有时用户故事这种形式也可以描述非功能需求。例如，Cohn在他的书中就给出了几个例子：

故事卡18.11
老顾客必须能够在90秒内找到书和下订单。
（约束）

故事卡18.27
系统必须能够支持50个并发用户。
（约束）

以上两个故事描述了对系统的易用性、性能等方面的非功能需求，如“90秒内、50个”等。

关于到底什么是用户故事，还有一种更为全面和重要的定义。XP知名专家、创始团队成员之一的Ron Jeffries建议用户故事应该由“卡片、对话与确认”这三个部分组成，即著名的3C（Card，Conversation，Confirmation）定义：

（1）卡　片

该部分是指写在故事卡片上的文字说明，通常比较简单，只有一两句描述。

故事卡片主要用于开发过程中的发布计划和迭代计划，以及作为一种提示物来启动和促进（以下第2部分的）对话。用来做计划和作为提示物，可以说是用户故事的两个基本用途。

故事卡片是用户故事的一种最直观、明显的物理表现形式，然而它们却不是最重要的；就像一种临时的占位器（Placeholder），卡片只代表了客户需求，而不是（真正地）记录需求。

（2）对　话

该部分是指用户与开发者通过拿着故事卡片，相互之间开展对话与沟通，以获得和澄清用户故事的具体细节的过程。

(3) 确　认

该部分主要是指编写(和执行)针对用户故事的各种测试(主要是验收测试),这些测试中记载的大量细节可用来确认一个故事是否真正地完成了。

这说明除了前面两部分的卡片、口头沟通与对话之外,用户故事所代表的需求的更多细节,应当以大量的测试(包括测例声明等)方式来进行书面的表达和记载。Cohn建议在故事卡片的背面记录测试要点(或验收测例的简述)。然而一张故事卡片的容量小,可记录的测试信息非常有限,对于一些复杂的故事测试,不如采用测试文档更为有效。

关于以上3C的三个组成要素之间的联系,Cohn总结到,这是对用户故事的最佳诠释：卡片包含了对故事的简短文字描述,然而需求细节要在"对话"中获得,并通过"确认"部分得以记录。

可见,从获得和记录需求的细节(主要内容)看,用户故事的"对话"与"确认"这两部分远比只含简单说明的故事卡片本身更重要,如此设计的一个效果就是基本避免了编写或制作详细的需求文档(或模型),因为所有需求的复杂细节似乎都可以间接地通过频繁对话和测试反映出来。

可是这样做,真的都好么?

7.2　两个故事比较

既然我们知道,特性、用例等技术也同样可以表示系统的功能,那么用户故事与这些技术之间有什么明显的区别呢?

下面就在介绍用户故事的优缺点之前,先来分析一下用例与用户故事这两个故事之间的异同点。

Mike Cohn认为用户故事与用例的区别主要有如下4点：

- 粒度不同;
- 完全性不同;
- 生命期不同;
- 用途不同。

以下将分4个小节分别对这些不同点进行分析,然后再介绍两个故事之间的共同点(与等价性)。本节所引用的Cohn的各种观点和论述主要译自他的文章 *Advantages of User Stories for Requirements*。

7.2.1　生命期

关于用户故事与用例故事在生命期上的不同点,Cohn是这么说的：

"用例与用户故事一个重要的不同点是两者的生命期。

用例经常是一种永久(或长期性)的工件,只要产品的开发或维护没有终止,它们就会持续

存在。

而用户故事有所不同，它们的生命期一般不会超出其所产生作用（或当它们被分配到软件上）的某个迭代。存档故事卡片有时确实也可行，不过许多团队的做法是用完就直接把它们撕掉了。”

赞同 Cohn 的以上看法，用例的生命期通常要远大于用户故事。

他强调用例通常是一种“永久性”的工件，其生命期几乎与一个产品的开发与维护期一样长。而一个用户故事的生命期，却通常不超过一个迭代，一旦功能实现且通过测试了，用完就可以扔掉，Cohn 还说“对话比卡片更重要”，这说明用户故事卡片所起到的作用一般仅具有临时性和提示性。

那么，为了让开发机构、甲方或乙方团队能够长期地保留产品开发中的重要知识资产——需求工件，用例与用户故事（传统的、非电子版）这两种记录需求的不同形式，一个持久、一个临时，哪个更为重要、更有价值呢？

不言而喻。

7.2.2 完全性

关于用户故事与用例故事在完全性（Completeness）上的不同，Cohn 是这么说的：

“用户故事与用例在完全程度上也有所不同。

专家 James Grenning 认为：一个用户故事卡片上的文字，加上它的验收测试，基本上就等同于一个用例。这意味着，一个用户故事（卡片）对应于一个用例的基本流，而它的验收测试则对应于该用例的扩展流。”

（**注：**有关用例的基本流和扩展流等概念，请参阅本书第 3 章）

大体上同意以上的说法。

这说明用户故事（卡片）本身不是一种比较完整的需求描述，尤其对于一个比较复杂的系统功能或需求而言，它缺少对很多重要需求细节的书面记载。

这也是为什么 XP 专家 Ron Jeffries 提出一个用户故事至少需要 3C（卡片、对话与确认）相结合才是完整的，而 3C 中的第 3 个 C（即确认）主要就是指 Grenning 所说的验收测试，其中包含了许多针对功能使用中可能发生的各种扩展和异常等情况的测例（Test Cases）。

那么，一个用户故事卡片加上它的验收测试就真的与一个用例完全等价了吗？

事实上，情况没那么简单，只能说两者大体上对应，而不是完全等价（参见后面 7.2.6 小节“偏等价性”）。

例如，用户故事与用例的基本流之间还是有着明显的区别的，关键是用户故事一般只描述了用户想得到的一个目标，而缺少像用例基本流那样，对如何达到这个目标的基本动作步骤（交互流）的清晰、完整的书面描述。

说“用户故事的验收测试对应于用例的扩展流”，有一定道理。然而，用例的扩展

流是需求描述，而用户故事的验收测试则是一些测例描述或声明。虽然两者之间确有联系，因为每一个扩展流都代表着一种需测试的特殊情况，与一个或多个验收测例相对应，但是需求归需求，测试归测试，两者之间还是有着根本的区别。当然，除扩展流外，验收测试也应该包括针对用例基本流的测例。

而且，通过一个功能的测例或测试脚本来反推、了解需求的细节，常常是比较困难的，还不如直接阅读用例的交互流文本来得更直接和方便。

那么，用户故事是如何来细化、澄清一个复杂功能的需求细节的呢？主要靠3C中的第2个C(对话)与第3个C(确认)，即靠现场用户代表与开发者不断地对话沟通，并编写测例和测试程序来最终搞清楚一个系统功能或需求的复杂细节。

所以，单纯作为需求描述的工件，用户故事(卡片)肯定是达不到用例描述可以达到的那种需求完整(或完全)性的。除了用例名称、简述以外，通常用例的文本描述(如大纲、详述等)，甚至UML动态图也都比一张极其简单的用户故事卡片的内容要更加丰富和全面。

显然，对于能够更快地澄清复杂需求的细节，高效地驱动后续的设计、编码与测试，信息量更多、更丰富的书面用例故事的优势是用户故事所无法比及的。

7.2.3 粒 度

关于用户故事与用例在粒度大小上的区别，Mike Cohn在《用户故事与敏捷方法》中是这么说的：

> 用户故事与用例一个最明显的区别就是粒度不同(**注**：原文用的是Scope，主要是指一个需求的内容在完成开发用时或工作量多少的“范围”，不同于本书中用例的属性“系统范围”所指的当前系统所涉及的具体内容有多少的那个“范围”，为了避免混淆，故此处译成与其涵义更接近的“粒度”)。
>
> 两者无论大小如何，均可以提供业务价值，但是用户故事的粒度通常要更小一些，因为我们对它们的尺寸有具体的限制(例如“所有故事都不应该预计含有超过10天的开发工作量”)，这样就可以把它们用于排程，而一个用例的粒度几乎总是要大于用户故事。

Cohn的这些分析既对又不全对。

首先，他对用户故事粒度的介绍基本上是对的。确实是这样，为了用于迭代计划与任务排程，一个用户故事的粒度大小通常应该控制在可于一次迭代之内完成。因此，Scrum+XP推荐在一个迭代内用大小合适的用户故事来驱动开发。

这里需要补充的是，用户故事不全都是小粒度的，后来又发展出了“史诗(Epic)”级别的大粒度用户故事，这种故事的粒度与普通用例或大用例相似，通常可以跨越多个迭代。

其次，Cohn关于用例粒度也只说对了一半。确实，用例为了要能完整、准确地反映真实的用户目标和需求，粒度通常比普通的用户故事要大一些，实现一个用例(大粒度需求)往往需要花费一两个或多个迭代的时间。

然而，用例的粒度总是比用户故事大吗？不是的。读过Cockburn著作的Cohn

可能忘了,正如用户故事可分为史诗故事与普通的用户故事,用例的粒度大小也是分层的,既有开发完成时间通常超过一个迭代的大用例(概要目标层)、普通用例(用户目标层),也有粒度更小的小用例(子功能层),而这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。

例如,Cohn 在他的《用户故事与敏捷方法》第 18 章“一些用户故事”中一共列举了约 27 个用户故事,下面我们就从中挑选几个比较典型的故事加以说明。

故事卡 18.1
用户可以用作者、书名或 ISBN 搜索书籍。

以上用户故事刚好对应于一个用例“搜索书籍”(用户目标层):

用例:搜索书籍
简述:用户可以用作者、书名或 ISBN 搜索书籍。

用例的粒度总是比用户故事大吗?不一定。以上这个用户故事与用例的粒度就是完全一致(相同)的。

再来举一个小粒度用例和与其相对应的用户故事的例子。

故事卡 18.4
在完成订单前,用户可以从她的购物车中删除书籍。

以上故事对应于一个小用例(从购物车中)“删除书籍”(子功能或鱼虾层):

用例:删除书籍
简述:在完成订单前,用户可以从购物车中删除书籍。

“删除书籍”这个小用例其实只是上级用例“使用购物车”当中的一个动作步骤(参见例 6-14)。在使用购物车时,用户不但可以删除书籍,还可以执行修改(增加或减少)购买书籍的数量等操作,而这些不同的操作同样可以对应于(或提取为)一个独立的用户故事。

可见,并不是所有用例的粒度都比用户故事大,许多小用例的粒度也和用户故事差不多,基本上也都能在一个迭代之内完成开发。只不过在做用例分析时,一个个的小用例通常都隐含在其上一级用例的内容(交互流)描述当中,而且一般我们不提倡在开发过程中过早地提取出小用例,以免出现“只见树叶不见森林”的现象。

以上实例说明,一个用例通常可以分解为(或对应于)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(或史诗大故事)。

用户故事与用例粒度分层的大致对应关系如表 7-1 所列。

表 7-1 用户故事与用例粒度层级的对应关系

用户故事类型	用例层级
史诗故事	大用例(概要目标层) 用例(用户目标层)
(一般的)用户故事	用例(用户目标层) 小用例(子功能层)

此外,用例故事的分解其实有多种灵活的方式,不仅可以分解为用户故事,还可以拆分成多个特性、用例片段(块)、被包含或扩展用例等。

7.2.4 用 途

关于用户故事与用例在使用目的(或用途)上的不同,Cohn是这么说的:

用例与用户故事的另一个不同点是,两者具有不同的写作目的。

用例的书写采取了一种让客户和开发者都易于接受的格式,以便他们阅读并对用例的内容达成一致。编写用例的目的是记载客户与开发团队之间达成的一份协议。

而用户故事,它的编写目的是促进发布与迭代计划的制定,以及作为占位器(Placeholder)来引导各方为进一步细化用户需求而展开讨论。

Cohn关于用户故事编写目的和用途的说法没错,它们主要是用于发布计划和迭代计划,以及作为一种启发和引导细化用户需求的各方对话、讨论的占位器。正是因为主要作为一种占位器或信息提示卡来用,用户故事在卡片上的书写内容自然就可以很简略,通常只有简单的一两句话。

然而,这里Cohn对用例目的或用途的理解是片面的。他认为编写用例好像就是“为了记载客户与开发团队之间所达成的一份协议”,这其实只是用例编写的多个目的或用途之一。看了他的结论,有人可能会产生误解,以为作为项目计划和需求讨论占位器是用户故事的特长,而这两项用途用例技术都无法提供,这是一种偏见。

不要忘了,用例技术也有多种应用场景和多种表现形态。

确实,对于某些有正式、详尽需求文档要求的开发项目,用例采用了一种客户与开发者都能接受、便于阅读的文本格式,其详细描述的需求内容(主要是功能交互)可以说反映了客户与开发团队之间达成的一种需求协议(或契约),因而一般不能随意变更、修改。这种形态的用例(以文本详述为主)通常主要适合于比较传统的文档驱动型的工程开发类项目,如需要客户与开发商正式签约,在开发过程中需要正式签署、确认需求文档,变更需求则需要走正规的审批流程等场景。

然而,除了详尽的用例文本描述这一种形态以外,用例表示本身还有多种简易的中间形态,如用例名称、用例简述、UML用例图中的相关符号、用例大纲等,这些形式的用例照样可用于不那么强调契约式、正规的需求文档,而与用户故事的适用场景相类似,走简易、频繁反馈型流程的敏捷开发。

以上分析可用表7-2来概括。

表7-2 能起到与用户故事类似作用的多种用例形态

用户故事的用途	可起到相同作用的用例形态
驱动发布计划和迭代计划	各层大小的用例名称或其简述 (概要目标层、用户目标层、子功能层)
讨论需求细节的占位器(提示卡)	用例名称 用例简述 用例大纲 用例图 活动图、交互图等UML动态图

表7-2说明UML用例图、用例的名称、用例简述等也都可以起到与用户故事相类似的讨论需求细节的占位器、提示物等作用。

如果要做发布计划或迭代计划,那么通常小用例(即子功能层的用例)与用户故事的粒度大小差不多(即预计开发完成时间不超过一个迭代),同样也可以起到类似于后者的作用。这主要是因为用例与用户故事在所描述需求的内容实质上存在着一定程度的等价性(参见7.2.6节)。

需要补充和强调的是,关于促进讨论需求细节的作用,事实上含有更多有价值书面信息的用例(包括UML图与脚本)往往可以比用户故事做得更好。

此外,Cohn还提到了在用例编写中经常出现的几个问题或误区,他可能认为这些是用例方法的缺点:

> 首先,用例编写经常要耗费大量纸张,而且在缺少其他合适的地方用来存放用户界面(UI)需求的情况下,结果是它们往往出现在了用例描述当中;
>
> 其次,用例的编写者经常过早地聚焦于软件的实现,而非业务目标。

然而,以上现象是用例方法或技术本身的缺陷吗?

不是。

在产品功能需求描述当中混入(大量)UI需求或软件实现等内容,这两个问题其实是在用例技术的实践当中,过去常见的应用缺陷或错误,是由于一些应用者使用不当、对用例技术理解有误造成的,而并不是用例方法本身的缺点,出自用例专家的正确用例编写方法与技巧一般都不建议这么做(Cockburn)。

同样,我们也不能把应用者对用户故事的误解和误用,当作用户故事方法本身的问题。

7.2.5 与用例简述比较

……

7.2.6 偏等价性

以上我们比较了用户故事与用例故事的不同点，下面再来说说两者的共同点。

尽管有些专家出于某些原因始终不肯承认，但其实用户故事与用例故事之间存在着事实上的(偏)等价关系。

所谓“偏等价”，有两层基本的涵义：一，既然是偏等价，就说明不是全等价，如果两者(全)等价，那么这两种故事中的任一项技术就完全失去了其价值，可以被另一项技术所取代而无需继续存在；二，“偏”是指在需求工作中，用例故事基本上可以取代用户故事，而反之不行，即用户故事无法或很难取代用例故事。

下面是用户故事的另一种常见书写格式(根据敏捷联盟的介绍，此格式也被称为“角色—特性—原因”或 Connextra 模板，发源于一家英国公司 Connextra。而本书其他地方出现的用户故事写法源自 XP 创始团队，相比之下书写比较自由，更接近于一般的特性描述)：

As a *role*, I want *goal/desire* so that *benefits*.

这句标准格式的用户故事很容易转换成如图 7-1 所示的用例图。

图 7-1 用用例图来表示一个用户故事

从以上用户故事的格式内容与所对应的用例图来看，显然用户故事与用例故事这两者之间存在着如下简单的一一对应关系：

一个用户故事的角色(Role)就相当于一个用例的用角(Actor)，而用户故事的目标或期望(Goal/Desire)也很容易转换成一个反映用角(或用户)目标的用例名称。

此外，用户故事中角色所期望获得的好处或利益，可以在用例图中用一个 UML 标签或者在相应用例的文本模板字段(如“简述”)中加以说明。

事实上，除了小(或子功能层)用例以外，每个用例(名称)都反映了一个真正或主要的用户目标和意图(Intention)。而本书前面已经介绍过，从用户角度出发，或以用户为中心(User-Centered)、基于目标驱动，这些也正是用例技术由来已久的一个优

点和特点。尤其在反映用户目标和业务价值这点上，两种故事其实是基本一致的。

那么，既然用户故事与用例故事两者不是全等价，它们的区别又在何处呢？

除了前面已介绍过的生命期不同以外，主要还在于表现形式以及内容的丰富程度（即完整或完全性）不同。

表达一个用户角色针对系统的目标及其具体操作和交互，一个只用简短文字表述，外加口头叙述（用户故事），而另一个既可以画图，也可以用文字简述，或者采用文本模板进行详述，当然同时也不排斥口头交流（用例故事）。

虽然这两种故事技术表达需求所采用的具体形式与内容丰富程度各有不同，但是它们所反映或代表的需求本质上几乎是一致的。

与“本质用例”的共同点

Cohn还提到了Constantine与Lockwood两位专家提出的Essential Use Case（EUC，本质用例）方法，这也是用例方法的一个流派。

关于EUC的特点，Cohn是这么说的：

> 本质用例去掉了对技术与实现细节的隐含假定……另外，关于本质用例有趣的一点是，其中所反映的用户意图可以直接转译成用户故事。

以上EUC的特点（去掉了技术和实现细节、反映用户意图等）其实已经被如今的Cockburn、Jacobson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对技术和实现细节的描述，确保用例步骤的编写反映了用户的真实目标和意图等，并且本书在其他章节中也对这些特点做了介绍。

关键的一点是，既然反映了用户意图的用例（或其中的内容、步骤）可以直接转译成用户故事，或者与某些用户故事相对应，那么这也从一个侧面（尤其像Cohn这样的用户故事专家的角度）佐证了当代用例与用户故事在所反映内容的实质上具有某种等价性。

小结如下：

参照用户故事的3C定义（7.1节），其实一个用例至少也可以由三部分组成：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。

在对后两部分（对话与确认测试）的要求上，用户故事与用例故事其实没有本质的区别，而且对于第一部分，用例同样可以用于项目计划和作为需求对话的提示物。

两者的区别主要在于需求描述的形式与内容详尽程度。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、扩展流、业务规则、非功能需求等字段。由于用例文本提供了更加丰富的流程化和结构化的需求描述信息，对于提高后两部分（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、序列图等直观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与形式、内容上可谓更加

完备。

通过以上分析，可以得出如下几点结论：

- 无论是用例故事，还是用户故事，都没有发明除功能需求与非功能需求以外新的需求类型。
- 一个用户故事（或史诗）所能表达的需求（主要是功能需求），也都能用一个与其相对应的用例（无论是大用例、普通用例，还是小用例）来表达。两者主要只是在需求表现、描述的具体形式上有所不同，而它们分别所要反映的内容实质是基本相同的。用例所提供的书面、有价值的信息通常比用户故事要丰富得多。
- 任何一个系统功能既可以用用例故事来表示，也可以用用户故事来表示，应根据不同应用场景的特点、情况来取舍。
- 一个用户故事的书面内容与一个用例（的用角、用例名称及其简述）相等价。

7.3 用户故事的优点

在2004年10月发表于*Informit Network*上的*Advantages of User Stories for Requirements*这篇著名文章中，Mike Cohn分析和列举了用户故事相较于用例等其他需求技术的一些主要优点，类似的观点在Cohn后来出版的《用户故事与敏捷方法》一书中也有所反映。

Cohn认为，与用例和其他传统需求技术相比，用户故事的不同之处和优势主要可以归纳为以下三点：

- 对话优先（强调口头交流，比文本描述更准确）；
- 适宜做计划（更适用于做项目计划和估算）；
- 推迟确定细节（简单易用，延迟细化需求，有利于迅速启动敏捷开发）。

那么，用户故事的这三项主要优点是否就是用例的缺点或不足之处呢？

答案是否定的。

同时，我们也并不赞同他认为总体上用户故事比用例更好、更适合敏捷开发的观点。下面是我们对以上Cohn观点的评论、分析与反驳。

7.3.1 优点一：对话优先

Cohn认为用户故事的第一个优点是用户故事相比其他技术更强调口头交流，可消除文本需求的多义性，他是这么说的：

> 用户故事强调口头沟通。书面语言经常是非常不准确的，而且无法保证客户与开发者对同一句书面陈述拥有相同的解释。我们总以为书面语言是准确的，并据此开展工作，然而它们常常并非如此。

用户故事卡片的文字内容非常简单，通常只有简单的一两句话，那么对于大量复杂的需求细节，不写下来怎么办？只好主要靠与现场的用户代表对话来澄清了。所以，鼓励用户代表与开发者之间面对面的口头交流，口述故事，强调对话比卡片更重要以及对话优先，这确实是用户故事的一个特点和优点（在某些特定情况下）。

然而，以上观点也可能会让人产生这样一些看法和误解：口头沟通一定好于书面沟通，用例采用书面记录需求的方式不鼓励对话、沟通等。事实真的如此吗？

以下我们分两个层面来分析。

（1）用户故事比用例故事更适合口头交流

XP 的支持者认为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。

其实，用例技术比单纯的用户故事卡片能更好地促进用户与开发者之间的沟通，在促进对话这方面至少不必用户故事差。

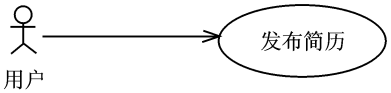
例如，这是 Cohn 书中的第一个用户故事：

故事卡 1.1
用户可以在网站上发布简历。

拿着这张卡片，我们并不知道用户到底应该如何发布简历，显然具体的操作和使用细节需要用户代表与开发者之间的对话以及编写测试来填补。

而如图 7－2 所示的用例图、用例简述与以上故事卡完全等价，而且内容简短，同样可以起到提示、促进对话与沟通的作用，甚至如果需要，我们也可以像用户故事那样把它们画（写）在卡片上。

a)



b)

用例：发布简历
简述：用户可以借用现成模板，填写并发布个人简历，以供用人单位查询和检索。

图 7－2 “发布简历”的用例图与用例简述

有了如此简单的用例图和用例简述，还能说“促进对话”是用户故事独有的优势吗？

当然，用例促进对话的手段还不只有用例简述、用例图这两种。在与用户沟通一个复杂的系统需求时，如果讨论仍然不够清晰、明确，那么可以马上在白板或电脑工具上画出用户使用该功能的流程图（活动图）、交互图、状态图等更加直观的图形，或者快速写出相应用例基本流的步骤大纲，再者还可以配上界面原型、线框图等来强化

说明效果。

可见在促进与用户进行有效对话、沟通需求方面，与用例方法所具备的多种文本加图形的丰富描述手段相比，用户故事卡片则显得内容太少、形式太单薄了。

所以，我们应该思考，到底拿什么东西、怎样去和干系人面对面地对话沟通、澄清需求，才是一种更好、更加高效的方式？

只是用一种简单的、只有寥寥几句话的故事卡片好呢，还是用有更多更规范的书面描述细节（如用例大纲）以及直观图形展示（如各种 UML 动态图）的用例模型更好呢？

前者可能主要适用于一些需求相对简单的小产品，而后者更适合需求比较复杂的大中型产品和系统（或者小产品中的一些复杂功能）。

（2）口头交流需求就一定比文字描述需求更好吗

前面 Cohn 说到，写下来的文字语言往往很不准确，甚至“无法永远保证客户和开发者对同一个句子有着相同的理解”，这种说法未免有点片面和夸大了。

的确，基于自然语言的需求文本描述常常不够准确，可能存在着多义性、不完整、不一致等问题。所以，需求文档的正式评审，除了分析师、架构师等内部人员参加以外，常常还需要用户代表、领域专家等外部人士的参与。并且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、确认需求，才能保证当前开发的系统需求确实符合用户的需要。

然而，难道口头交流需求就比文本描述需求更加准确，人们交流所用的口语就不存在表达上的多义性了吗？

其实表义不准的问题，口头交流中也经常存在，而且实践表明，口头约定需求比文本约定隐匿的风险和问题常常更大，而专业、规范的文字或图形描述往往比口头交流更为可靠和准确。例如，在日常工作中存在着这样一种普遍现象：当遇到系统需求或设计中一些一时说不清楚的问题时，大家经常会利用办公室里随处可见的白板来画一些图形或者书写一些文字，以展开深入讨论与沟通。这也充分反映了针对理解和澄清复杂问题，书面交流的效力往往要高于口头交流。

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，主要是因为人们采用了自然语言。例如，随便翻开一本语言词典，就会看到大量字或词语都至少有一种以上的不同解释。人类语言的天生缺陷会造成表义不精准，存在多义性，这是一种客观现象。

用户故事提倡对话优先，减少甚至不写需求文档，那么仅仅依靠拿着故事卡片，进行口头交流就可以完全消除沟通上的多义性、模糊性等现象吗？

显然也难以做到。

沟通需求，无论是采用口语还是书面语，如何才能尽量避免采用自然语言描述需求所产生的各种问题？答案是：用更加科学、系统和规范的办法，如结构化、形式化（或半形式化）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为精准，当然这同时也增加了阅读理解的难度，不如用口语沟通那么直接、方便。形式化的需求描述方法主要流行于学界和科研界，在一般工业界的日常实践中并不多见。

而业界流行的用例模板(语言)是一种介于完全形式化和自然语言之间的一种“半形式化”与结构化描述方法，在追求需求语义的精准性、可读性和可理解性、书写的便利性等因素之间做出了较好的平衡。

小结一下：

用简单文字描述需求，以促进用户与开发者之间的对话、沟通，这并非用户故事独有的优点，其他具有类似形态的技术，如特性、用例简述、用例图等也同样可以发挥类似的功能。

“对话优先”并非适合所有类型的项目开发，单纯的口头对话也并非在任何情况下都是一种最佳的沟通方式。

另一方面，有了更加完善的需求模板和交互脚本的支持，适当的文本、图形描述加上口头沟通，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。

对于提取和分析复杂的产品需求，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、详细的模板字段、规则和技巧等建议，开发者和用户代表们就能有的放矢，更加系统、准确、有条理地进行沟通，从而尽早达成一致意见，消除各种模糊、多义性，促进系统需求的尽快稳定和收敛。

此外，基于规范文本与图形描述的用例模型，比强调口头交流的用户故事卡片集，内容更准确、细节更丰富、形式也更完备，不仅能更好地促进与客户沟通需求，还能有效提高测试编写的效率和质量。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，并通过编写测试(试图用各种测试声明来全部取代需求模型)以补充细节，这对于许多软件开发项目而言其实是一个不小的风险隐患和质量缺陷。

补充和完善需求细节，完全依赖对话驱动而几乎不写任何需求文档，这是一个极端；而编写了大量的需求文档，却很少进行面对面的有效沟通与对话，这又是另一个极端。无论坚持哪种极端做法，都是不明智的。

7.3.2 优点二：适宜做计划

Cohn认为用户故事的第二个优点是相比其他技术，它更适于做项目计划(包括发布计划、迭代计划等)。对此他是这么说的：

> 用户故事的第二个优势是它们很适宜做项目计划。用户故事所采用的编写方式，使得很容易估算它们的开发耗时或难度。而用例常常(粒度)太大了，所以很难获得有用的估算。
>
> 此外在敏捷项目中，一个用户故事通常可以在一次迭代内完成开发，而一个用例的完成通常都需要跨越多个迭代(尽管这些迭代的时间往往比故事驱动型项目的迭代还要长)。

以上观点的实质还是在说用户故事的粒度，因为用户故事的粒度一般比用例要小，且合适（完成时间通常不超过一个迭代），所以它比用例更适合于做计划和估算。

下面我们分两个层面来分析。

（1）用例不适合做项目计划和估算吗

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它们来做估算、排序更为方便。那么相比之下，是否用例就不适合做项目计划和估算了呢？

非也。

首先，同样描述系统的功能，为什么用例的粒度一般比用户故事大呢？这是有其内在道理的。

每一个普通用例其实都是一个比用户故事内容更为完整的需求故事。典型的用例与用户故事之间的关系，就像一棵大树的树干与树枝、树枝与树叶的关系。用例可以充当容器把各种琐碎的小故事拼接起来，让我们看到整个系统功能的全貌。

除了子功能层的小用例以外，每一个用例都是对用户真正具有价值（反映了用户目标）的需求单元，其粒度通常比用户故事大，这导致一个系统的用例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视图，而不是“只见树叶不见森林”。

因此，始于系统用例图的用例分析技术并不是传统意义上的功能分解，而更像是需求或功能的聚合，可以让我们站在大量琐碎、关系错综复杂的小功能（如用户故事）之上，看清少数真正具有业务价值的关键用户目标和系统服务。

制定出有效、合理的项目计划，有时既需要大粒度的需求，也需要小粒度的需求。而用粒度稍大、数量更少（如概要目标层和用户目标层）的用例来辅助做项目的发布（或全局性）计划，效率往往比小粒度的用户故事更高，效果也更好。

为什么用户故事方法后来也发展出了史诗级故事？其中的道理与用例是类似的。

所以，“因为用例的粒度一般比用户故事大，所以不适合做计划”，这个结论不能成立。

其次，Cohn还说由于用例的粒度常常过大了，所以不能给出有用的估算。果真如此吗？

非也。

Cohn忽视了用例的层级与切分（分解、拆分）技术。

用例估算的正确做法是分而治之。为提高估算的准确度，有时可以把一些粒度过大的用例分解为更小的需求单元进行估算。用例的分解可采用多种方式，一个用例既可以分解为用户故事，也可以分解为小用例、用例块、特性等多种形态。把一个用例所包含的所有需求单元的估算值累加起来，往往就可以得到这个用例总的估算结果（如估计工作量或耗时等）。可见，在利用粒度更小的需求单元做计划与估算这

点上，用例与用户故事的基本做法其实是一致的。

两者的区别主要在于：在需求细节内容的描述上，用例比用户故事提出的编写要求和信息量更多。基于文本模板的用例脚本提供了更多的需求细节（如前后态、触发事件、基本流、扩展流、业务规则等），从而与缺少大量书面细节作参考的用户故事相比，有利于我们做出更为准确、免于太乐观的估算。

（2）用户故事更适合做估算吗

Cohn 认为针对每个用户故事（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，估算的准确性往往就越高。

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难点或风险，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上的那一点提示，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 3C，参见 7.1 节）就能获得更可靠的估算。这种说法有一定道理。

然而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，会更有利于促进复杂需求的理解、澄清与交流。同时，有了对大量的用例扩展流所反映的特殊情况等重要需求细节信息的书面记载，也便于写出更加完善的测例。从需求用例到测例，是一个更自然的系统化思考与分析过程。

此外，即便在用户故事估算的过程中，适当编写每个重点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信度。

总之，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（如子功能层用例或小用例），因而用例同样适合做估算。

小结一下：

用户故事的粒度通常比用例小，这并非它特有的优势，因为一个粒度较大的用例同样可以分解成其他更小、适于做项目计划和估算的需求单元，如小用例、用例块、用户故事等。

用户故事粒度小且适中，所以适宜做计划，这确实是它的一个优点。然而与用户故事相比，用例不但可以提供与其粒度相似的功能小单元，而且还提供了更高质量、更丰富的其他书面需求细节信息，所以用例更适合做项目计划和估算，至少不比用户故事差。

7.3.3 优点三：推迟确定细节

Cohn 认为用户故事的第三个优点是形式简短，更有利于团队推迟收集需求的细节，从而可以迅速启动开发。

他认为,用户故事方法鼓励团队推迟收集需求的细节,一开始先用相当于占位器的故事卡片简略地描述一些重要需求,在无需编写详细需求文档的情况下就可以迅速投入开发,然后在适当和必要的时候再补充、完善更多的需求细节。因此与其他需求技术相比,用户故事尤其可以"完美地"适合进度紧张的项目开发。

Cohn 的观点有些道理,用户故事采取了一种看似不错的策略。但是,只可惜 Cohn 没有提及 UML 用例图与用例简述,而后者同样可以起到与用户故事类似的需求占位器乃至提供系统全局视图的作用。

例如,在本书宠物店案例的用例模型中,仅用一张 UML 用例图就直观、形象地勾勒出了系统的几个主要功能,以及用户与需求、需求与需求、用户与用户之间的多种关系,如图 7-3 所示。而采用用户故事简略地描述这些需求至少需要多张独立的故事卡片,不仅书写的文字可能有冗余,而且各个需求(故事)之间的关系也无法像用例图那样直观地表达出来。

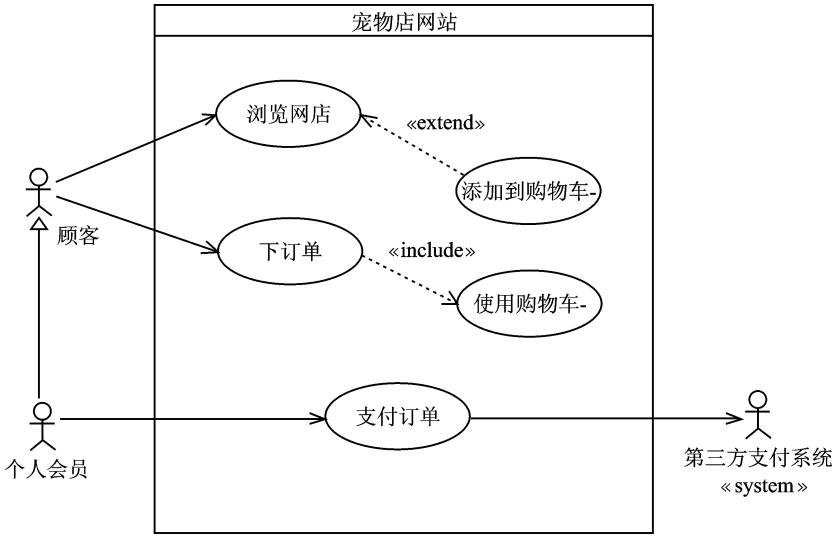


图 7-3 宠物店网站系统的一些核心用例示意图

其实,画用例图比书写用户故事卡片更简便,用例图中的每个用例也同样起到了需求讨论占位器的作用。

UML 用例图往往比一堆用户故事卡片更简单、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,用例图还可以直观地提供系统需求的全局、局部与关系等视图,这些是用户故事方法难以做到的。可见,用例技术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。

7.3.4 其他优点

……

7.4 用户故事的缺点

以上我们分析了用户故事的优点，那么用户故事有哪些缺点呢？

Cohn在其书中也列举了用户故事的几项缺点，不过主要针对的是大项目、大团队（如不可追溯等）。

其实，用户故事的缺点也是显而易见的，如果处理不好，对小团队、小项目或小产品的开发也可能产生负面影响。下面就对用户故事的几个主要缺点分别进行简要分析。

7.4.1 缺点一：不完整

用户故事的第一个缺点是不完整（或不完全，Incomplete）。

所谓的“不完整”，就是指用户故事所描述的需求信息是（非常）不完整的。

一张用户故事卡片所能记载的信息量是非常有限的，常常只能简单地记下几句话。故事卡片的尺寸小，必然导致可记录的书面信息少，因而往往缺少关键、复杂的需求细节，澄清需求大部分要靠现场用户代表与开发者之间的频繁口头交流。

这么做的好处是简单、易用、容易上手，但缺点也是很明显的：

缺少关键的需求细节信息，会导致需求描述的质量低、精准度差，所以必须通过后续不断的用户与开发者之间的现场沟通、口头解释来细化、明确用户故事卡片背后所代表的那些真正的需求信息。

从需求完整性的角度看，用户故事（卡片）是无法取代用例故事的。与一个系统功能的用例描述相比，用户故事卡片缺少了该功能需求的包括其主干（具体行为描述的交互流，如前态、后态、基本流、扩展流等）在内的绝大部分内容，而仅有一两句概述。

用户故事另一个明显的不完整性，体现在它们缺少明确的上下文（Context），且彼此之间具有分散性。

一张张的故事卡片是独立、离散的，很难一眼看出它们彼此之间的联系。而用例的主干交互流提供了一个很好的需求上下文，可以把众多零散的用户故事组织到一个完整的执行流程当中，使得各个小功能彼此之间的联系更加清晰可见。

对此Cohn是这么说的：

在拥有大量用户故事的大型项目中，故事之间的关系可能错综复杂，难以捉摸……面对大量的需求时，用例固有的层级性会使需求收集比较方便。一个单一的用例，通过其主成功场景（注：即基本流）与扩展流，可以把相当多数量的用户故事汇聚成为一个整体。

Cohn的这段话同时也清晰地表明了用例相较于用户故事的优点和价值，以及两者之间的内在联系，值得称赞。

7.4.2 缺点二：不正规

用户故事的第二个缺点是不正规（或非正式，Informal）。

这里的“不正规”，主要指的是用户故事在需求分析方面不是一种比较成熟、正规的技术。

前面已经介绍过了，用户故事的价值主要体现在项目管理上。在敏捷开发中它们起到了一种令牌（Token）或提示、占位器的作用，用来提醒大家需要细化和沟通某个需求，并作为代表系统需求的一个物理（或电子）的实物来驱动、辅助开发工作的计划、管理与跟踪。

然而，在产品的需求分析方面，与用例技术相比，用户故事只能算是一种比较初级、粗略的需求分析（或提取）技术，故事卡片上的那几句话所表达的常常只能是一种低精准度、模糊和不完整的需求。

对于现实中的大多数（尤其复杂的）软件开发，如果一个项目做完了只留下一大堆故事卡片，甚至按照用户故事的经典做法，在一个迭代内用完就把卡片上仅存的一点需求信息也给撕掉了，而没留下任何更加规范、完整、清晰的需求文档（或模型），这常常是不可想象的，也无法令人接受。

对此，Cohn是这么说的：

虽然故事在一个团队内部能大大促进隐性知识的积累，但还是不适用于特大规模多团队的结构。这时，确实需要把有些交流记录下来，不然难以保证信息在大型团队中充分共享。

显然，Cohn承认需求文档有时还是有价值的。不过Cohn在这里用的词是“特大规模”，意思是遇到特大规模的团队了，才可能需要编写需求文档，把口头交流、易忘的需求内容记录下来。

我们的看法与他有点不同：

其实不只是“特大规模”，而是一般的大、中型团队，甚至开发比较复杂一点的产品、系统的小团队，恐怕都需要编写、制作一些正规和专业一点的需求文档；而仅靠口头交流，缺乏（高质量的）需求文档（或模型），大概往往都是那些软件工程水平不成熟、缺乏开发经验的团队所为。

7.4.3 缺点三：不鼓励建模

用户故事的第三个缺点是不利于（或不提倡、不鼓励）创建与保存高质量、可视化

的需求模型。

复杂产品的需求分析需要运用更加科学、系统化的工程技术方法与手段，而Scrum＋XP的需求分析过程所缺少的一种重要实践正是“建立产品（或系统）的需求模型”。用户故事的3C（卡片、对话加确认，参见7.1节）无法从根本上取代需求模型。

建立需求模型的常见技术包括本书所介绍的UML与用例建模等，而且最好是图形与文本等多种描述手段相结合，这样才能获得最佳的分析、建模效果。

此外，在需求分析时编写用例脚本和绘制UML等图形的建模方式，对于随后复杂功能的测试分析与设计也大有帮助。20年来多位UP方面的专家（如Leffingwell、Zielczynski等）都曾经公开介绍过如何系统地、按部就班地从用例描述中分析、提取出测例的科学方法，基本思路是根据用例脚本画出对应的反映功能交互执行流程的活动图，然后为其中不同的流程分支设计不同的测例，以实现对测试路径的完整覆盖。这种方法清晰而有条理，可以说是对软件工程中传统测试分析与设计方法的一种自然继承与演进。

相反，在做复杂功能、交互的测试分析时，如果一点都不画任何的流程图或其他动态图，则相应的测例设计工作将会变得很困难，也往往难以保证质量。可见如何从简略的用户故事卡片与口头对话准确、高效地最终获得高质量的测例声明与测试脚本，这其实也是极限编程等不重视图形建模的传统敏捷方法的一项弱点和缺陷。

7.4.4 缺点四：不可追溯

……

7.5 小 结

关于用户故事与用例故事的4点不同之处，Cohn分析的结论比较准确。然而存在着这些不同，恰好说明了用例是比用户故事更重要、更强大的一种需求技术。

此外，鉴于用户故事与用例（简述）之间存在着事实上的偏等价性，可以发现Cohn所列举的几项优点（如对话优先、适宜做计划与推迟确定细节等）并非用户故事所独有，相应的等价物（如特性、用例简述、小用例、用例图等）也可以发挥几乎同样的价值和作用。

基于本章列举的用户故事的几项缺点，加上用户故事目前尚缺乏像用例与UML建模那样统一、有效支持上游业务分析的描述办法和手段，我们认为：其实用户故事更适合作为一种敏捷项目管理和计划的工具，如作为需求的占位符、提示器，驱动迭代开发等，这方面用户故事确实有它的长处，然而与本书所重点介绍的用例故事相比，用户故事算不上是一种成熟、完善的需求技术。

结 束 语

到这里本书就结束了，非常感谢您的阅读！

回顾全书，我们重点介绍了在当代敏捷开发过程中，采用基于图形符号的 UML 与基于文本模板的用例故事建模来进行产品的业务分析与系统需求分析的一些基本方法、步骤与技巧，而贯穿始终的是太极建模口诀——“由外而内，层次分明；动静结合，逐步求精”。

对于复杂的系统功能需求分析，利用用例文本模板进行详述往往是非常合适的，可以比采用特性、用户故事等其他简略技术所描述的需求具有更好的质量。有了具有更高精准度、更加稳定和全面的需求描述，就能更加顺畅、敏捷地驱动后续的交互设计、架构设计、编码以及测试等多项开发活动，以减少各种不必要的麻烦、浪费和扰动。

本书所采用的用例模板书写格式，吸收、借鉴了 Jacobson、UP、Cockburn 等流行用例模板的优缺点，引入了关键词、可嵌套执行块等多个创新的语法特征，从而使得用例文本看上去像是一种更加清晰、易读的“需求程序”，并且在此基础上有可能形成一种统一、规范的用例描述语言（UCL，暂定名）。我们正在研发免费的基于 Web 平台的 UCL 编辑工具，计划在不久的将来正式发布，敬请关注。

除了本书所介绍的利用 UML 描绘业务流程、功能需求以外，在日常的需求分析与建模活动中，还可以采用同为 OMG 国际标准的 BPMN（主要用于描述业务流程）、SysML（主要用于描述软硬件联合开发的系统需求）等技术，而这两种语言（或标记法）也都是在 UML 这个相对成熟、稳定的建模技术统一内核的基础之上发展而来的。

多年以来，坊间一直存在着许多贬低、忽视 UML 的看法和意见，而其中许多观点是片面、主观、不科学的。应用于软件工程，UML 的最大价值（之一）就是通过抽象、规范的建模，帮助分析师“化繁为简、抓住本质”，因此正确、聪明的 UML 建模实

践也可以是非常敏捷的。这其实与建筑工程、机械工程等许多传统行业中所广泛应用的图形建模(画图)实践相类似,都具有基础而重要的科学、工程价值和意义。

传统的敏捷开发方法论大多是基于用户故事驱动,也很少提及图形建模,而这远非什么完美、成熟的最佳实践。相信读完此书,您一定会对为什么“包含书面、详细的用例故事与动态图在内的产品需求模型,比一大堆用完即可抛弃的、主要用作口头沟通提示物的简略用户故事卡片具有更高的需求质量与实用价值,而且在敏捷开发中前者完全可以有效地取代后者”,获得一些更加深刻的理解和认识。

全面的需求分析,除了包含主要描述系统功能与动态行为的用例分析以外,非功能需求分析以及数据需求分析(如数据建模、领域建模等)是另外两个比较重要的内容。囿于篇幅限制,本书对于后两者只是简略提及,未展开深入介绍,请感兴趣的读者自行参阅有关著作。

最后,若您希望获取更多有关敏捷需求方面的方法、模板、工具、技巧等工程技术信息和知识资源,欢迎访问网站 umlgreatchina.org 和 zhangxun.com。