17讲程序员也可以“砍”需求吗

我们前面讲的任务分解，主要是在讲开发任务的分解。今天我们换个角度，看看需求的分解。是的，需求也要分解。

有一次，我和一个做开发的同事聊天，他给我讲了他近期的烦恼。

> **同事：**我们现在就是需求太多，开发的人太少，再这么干下去，哪天觉得自己抗不住了，我就拍拍屁股走人。
> **我：**你没尝试着砍砍需求？
> **同事：**怎么没尝试？产品的人都不同意。这批功能他们都说是关键功能。
> **我：**你有没有尝试把需求拆开了再砍呢？
> **同事：**还可以这样？

同事很惊讶，我一点都不意外。我们都是在说需求，但彼此对需求的理解却是大不相同。我先来问个问题，提到需求这个词，你会想到什么呢？

以我们用了好多次的登录为例，如果我问你这个需求是什么，大多数人的第一直觉还是用户名密码登录。

基本上，闯入你脑海的需求描述是主题（epic），在敏捷开发中，有人称之为主用户故事（master story）。

如果你对需求的管理粒度就是主题，那好多事情就没法谈了。比如，时间紧迫的时候，我想砍需求，你问产品经理，我不做登录行不行，你就等着被拒绝吧。

但是，如果你说时间比较紧，我能不能把登录验证码放到后面做，或是邮件地址验证的功能放到后面，这种建议产品经理是可以和你谈的。

这其中的差别就在于，后者将需求分解了。

大多数人可以理解需求是要分解的，但是，分解的程度不同，就是导致执行效果差异极大的根源。

以我的经验而言，**绝大多数问题都是由于分解的粒度太大造成的，少有因为粒度太小而出问题的。**所以，需求分解的一个原则

是，粒度越小越好。

需求要分解

“主题”只是帮你记住大方向，真正用来进行需求管理，还是要靠进一步分解出来的需求。这里的讨论，我们会继续沿用前面专栏文章中已经介绍过的需求描述方式：用户故事，它将是我们这里讨论需求管理的基本单位。

如果你的团队用的是其他方式描述需求，你也可以找找是否有对应的管理方式。

上一个模块介绍“以终为始”，我们对用户故事的关注点主要在：用户故事一定要有验收标准，以确保一个需求的完整性。而在“任务分解”这个模块，我们看用户故事，则主要关注它作为需求分解的结果，也就是分拆出来要解决的一个个需求点。

在前面的讨论中，我们已经知道了用户故事的“长相”，但更重要的问题是，划分需求的方式有无数种，就像一块蛋糕，你可以横着切，也可以竖着切。如果你一刀不切，那就是拿着主题当用户故事。你也可以快刀飞起，把主题切碎。

每个人都会有自己喜欢的拆分方式，我相信知道拆分的重要性之后，你总会有办法的。这里，我主要想和你聊聊怎样评判拆分结果，毕竟我们要把它当作需求管理的基本单位。

只有细分的需求才能方便进行管理。什么样的需求才是一个好的细分需求呢？我们先来看看用户故事的衡量标准。

评价用户故事有一个“ INVEST 原则”，这是六个单词的缩写，分别是：

- **Independent，独立的。**一个用户故事应该完成一个独立的功能，尽可能不依赖于其它用户故事，因为彼此依赖的用户故事会让管理优先级、预估工作量都变得更加困难。如果真的有依赖，一种好的做法是，将依赖部分拆出来，重新调整。
- **Negotiable，可协商的。**有事大家商量是一起工作的前提，我们无法保证所有的细节都能100%落实到用户故事里，这个时候最好的办法是大家商量。它也是满足其它评判标准的前提，就像前面提到的，一个用户故事不独立，需要分解，这也需要大家一起商量的。
- **Valuable，有价值的。**一个用户故事都应该有其自身价值，这一项应该最容易理解，没有价值的事不做。但正如我们一直在说的那样，做任何一个事情之前，先问问价值所在。
- **Estimatable，可估算的。**我们会利用用户故事估算的结果安排后续的工作计划。不能估算的用户故事，要么是因为有很多不确定的因素，要么是因为需求还是太大，这样的故事还没有到一个能开发的状态，还需要产品经理进一步分析。
- **Small，小。**步子大了，不行。不能在一定时间内完成的用户故事只应该有一个结果，拆分。小的用户故事才方便调度，才好安排工作。
- **Testable，可测试的。**不能测试谁知道你做得对不对。这个是我们在前面已经强调过的内容，也就是验收标准，你得知道怎样才算是工作完成。

“INVEST 原则”的说法是为了方便记忆，我们这里着重讨论两个点。

第一个关注点是可协商。作为实现者，我们要问问题。只是被动接受的程序员，价值就少了一半，只要你开始发问，你就会发现很多写需求的人没有想清楚的地方。

在我的职业生涯中，我无数次将需求挡了回去，不是我不合作，而是我不想做一些糊涂的需求。我之所以能问出问题，一方面是出于常识，另一方面就是这里说的用户故事是否是有价值。**用户故事，之所以是故事，就是要讲，要沟通。**

还有一个更重要的关注点，也是这个模块的核心：小。无论是独立性也好，还是可估算的也罢，其前提都是小。只有当用户故事够小了，我们后续的腾挪空间才会大。

那接下来就是一个重要的问题，怎么才算小？这就牵扯到用户故事另一个重要方面：估算。

需求的估算

估算用户故事，首先要选择一个度量标准。度量用户故事大小的方式有很多种，有人用 T 恤大小的方式，也就是S、M、L、XL、XXL。也有人用费波纳契数列，也就是1、2、3、5、8等等。有了度量标准之后，就可以开始估算了。

我们从分解出来的用户故事挑出一个最简单的，比如，某个信息的查询。这个最简单的用户故事，其作用就是当作基准。

比如，我们采用费波纳契数列，那这个最简单的用户故事就是基准点1。其他的用户故事要与它一一比较，如果一个用户故事比它复杂，那可以按照复杂程度给个估计。

你或许会问，我怎么知道复杂程度是什么样的呢？这时候，我们前面讲过的任务分解就派上用场了，你得在大脑中快速地做一个任务分解，想想有哪些步骤要完成，然后才好做对比。

所以，你会发现，任务分解是基础中的基础，不学会分解，工作就只能依赖于感觉，很难成为一个靠谱的程序员。

估算的结果是相对的，不是绝对精确的，我们不必像做科研一样，只要给出一个相对估算就好。

同一个用户故事，不同的人估算出的结果可能会有差别。怎么样尽可能在团队中达成一致呢？这就需要团队中的很多人参与进来，如果团队规模不大，全员参与也可以。

如果多人进行估算，你就会发现一个有趣的现象，针对同一个用户故事，不同的人估算的结果差异很大。

如果差别不大，比如，你觉得3个点，我觉得2个点，我们协调一下就好。但如果差异很大，比如，你认为2个点，我认为8个点，那绝对是双方对任务的理解出现了巨大的差异，这个时候，我们就可以把刚才在脑中进行的任务分解“摆”到桌面上，看看差异在哪。

通常情况下，是双方对需求的理解出现了偏差，这时候负责用户故事编写的同事就要站出来，帮助大家澄清需求。所以，**一般来说，估算的过程也是大家加深对需求理解的过程。**

估算还有另外一个重要的作用：发现特别大的用户故事。一般而言，一个用户故事应该在一个迭代内完成。

比如，你预计大小为1点的用户故事要用1天完成，而你团队的迭代周期是两周，也就是10个工作日，那13点的任务是无论如何都完不成的。那该怎么办呢？很简单，把它拆分成多个小任务，这样一来，每个小任务都可以在一个迭代中完成了。

所以，一般来说，用户故事有可能经过两次拆分。一次是由负责业务需求的同事，比如，产品经理，根据业务做一次拆分。另外一次就是在估算阶段发现过大的用户故事，就再拆分一次。

当我们有了一个合适的用户故事列表，接下来，我们就可以安排我们的开发计划了。只要厘清用户故事之间的依赖关系，安排工作是每一个团队都擅长的事情。

我在这里想回到我们开头讨论的话题。我们常说，需求来自产品经理，但需求到底是什么，这是一个很宽泛的话题。到这里，我们已经有了一个更清晰更可管理的需求，用户故事。这时候我们再说需求调整，调整的就不再是一个大主题，而是一个个具体的用户故事了。

许多团队真正的困境在于，在开发过程中缺少需求分解的环节。在这种情况下，需求的管理基本单位就是一个主题，既然是基本单位，那就是一个不可分割的整体。团队就被生生绑死在一个巨大的需求上，没有回旋的余地。

如果团队可以将需求分解，需求的基本单位就会缩小，每个人看到的就不再是“铁板”一块，才能更方便地进行调整，才会有比较大的腾挪空间。

总结时刻

总结一下今天的内容。软件开发中，需求管理是非常重要的一环。在需求管理上常见的错误是，需求管理的粒度太大，很多团

队几乎是在用一个大主题在管理需求，这就让需求调整的空间变得很小。

结合用户故事，我给你讲了一个好的需求管理基本单位是什么样子的，它要符合“INVEST原则”。其中的一个关键点是“小”，只有小的需求才方便管理和调整。

什么样的需求才算小呢？我给你介绍了一种需求估算的方式，每个团队都可以根据自己的特点决定在自己的团队里，多大的需求算大。大需求怎么办？只要再进行分解就好了。

如果你对用户故事这个话题感兴趣，推荐阅读 Mike Cohn 的两本书《User Stories Applied》和《Agile Estimating and Planning》。

如果今天的内容你只能记住一件事，那请记住：**想要管理好需求，先把需求拆小。**

最后，我想请你分享一下，你的团队在需求管理上还遇到过哪些问题呢？欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言

西西弗与卡夫卡

需求管理中印象最深的是刚进入开发行业时的一段经历。当时要显现某种单据，单据本身还嵌套子单据，原始需求是单据按层次显示，但某变量等于某个值时，某层次的单据就隐藏，但它的再下一层子单据需要显示。解决方案简单可以理解成页面要用递归方式才能正确显示。当时一心想显示自己牛鼻，因为无法用通常的页面模版机制，就自己写工具递归生成页面代码，看起来代码很巧妙，但别人难以维护。后来有段小插曲，若干年后项目移交到另外一个研发中心时，还需要有人专门去讲这段代码，否则接手人很难理解。再说回到之前，开发完后有一次和业务分析员再次聊起，说这个很不好实现但我啃下来了等等，他说那可以不用隐藏，显示空白层也可以，真如果这样实现就简单多了，也好理解多了。事不大，但在职业生涯里印象深刻。需求不光是拆解，更可以讨论后寻找简单解决方案，而不是用自以为牛鼻的代码实现。以更合理的成本实现需求交付价值，这其实是用户故事里Negotiable的意义所在

2019-02-06 00:38

作者回复

多谢大年初二的学习与分享！

程序员的锤子是代码，到处找钉子是直觉。

2019-02-06 19:38

WL

可衡量才可以讨论，跟老师学习受益匪浅

2019-02-12 13:20

毅

我对需求的理解分两个层面：用户需求和开发需求。在尽可能全面了解客户意图的基础上突出价值，通过开发需求框定范围。以往会形成两份不同的文档，主要是考虑到双方的知识背景不同而有针对性的准备，至于开发需求的误差程度往往得不到客户的有效确认，因为客户不习惯以程序员思维和工作模式去阅读开发文档，常常会给基本上是这样、差不多吧、先这么着看看这样的反馈。所以我在想应该可以用一种更“活泼”的方式去提高双方的沟通效率，如果能够以故事的方式去撰写用户故事，把问题拆分說透，用一个个能让对方身临其境界的场景故事去沟通，而非格式化的冷冰冰的文档去消除双方认知上的不足与分歧，这样除了可测试之外其他方面应该都能兼顾了吧。

2019-02-06 23:40

稳

老师，今天的内容，能不能理解为需要先将用户故事切分的足够小，然后以此来做需求合理性、工时评估、需求的降低标准等事情呢?

2019-02-06 09:26

作者回复

先将需求分成用户故事，只有到达可以评估的大小，你才能更好地理解，才能开始做后续的工作。

2019-02-06 12:04