给人惊喜的Scrum 第一卷 第一集

Scrum没有帮助你的公司变得更加敏捷的原因

曲解PO角色是如何对组织造成危害的，以及应该如何处理。

BY MJ

本故事早些时候是一段视频
https://youtu.be/cr2rjaGmUzo

以下のサイトにて、本ストーリーの日本語版ビデオおよび
コミック版をご覧いただけます。
故事视频的日语版本在这里
http://seattlescrum.com/jp/

MJ 感谢 *Shinya Shibusa, Benjamin Leffler, Tommi Johnstone*
以及*Yoko Hinoue*的帮助。

请将评论和纠错发到
mj@seattlescrum.com
或者使用*Twitter*
@michaeldotjames.

目录

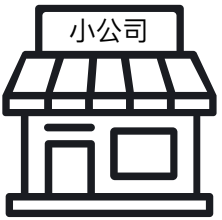
场景 1	PO这个角色应该如何工作?	1
场景 2	你所在的大型组织是如何曲解PO这个角色的?	3
场景 3A	曲解PO角色是如何延缓客户反馈的?	7
场景 3B	曲解PO角色是如何降低团队的开发积极性和客户同理心的?	11
场景 4A	一个真正的PO是如何交付最大客户价值的?	12
场景 4B	曲解PO角色是如何降低价值交付的?	15
场景 5	作为一个团队产出负责人糟糕在哪儿?	18
场景 6	我们如何在增加团队自组织和跨职能的同时帮助深陷这一角色的人?	20
场景 7	为何引入类似于“首席PO”这样的新角色并无必要?	23

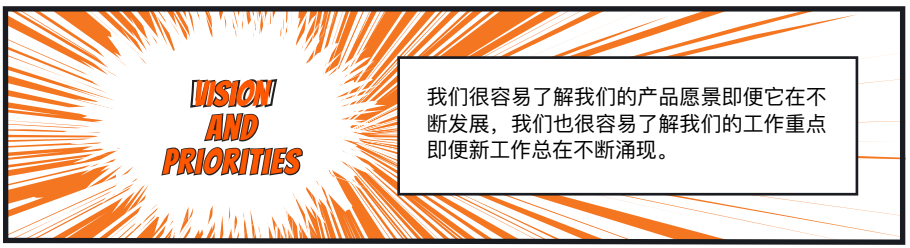
场景 *1:*

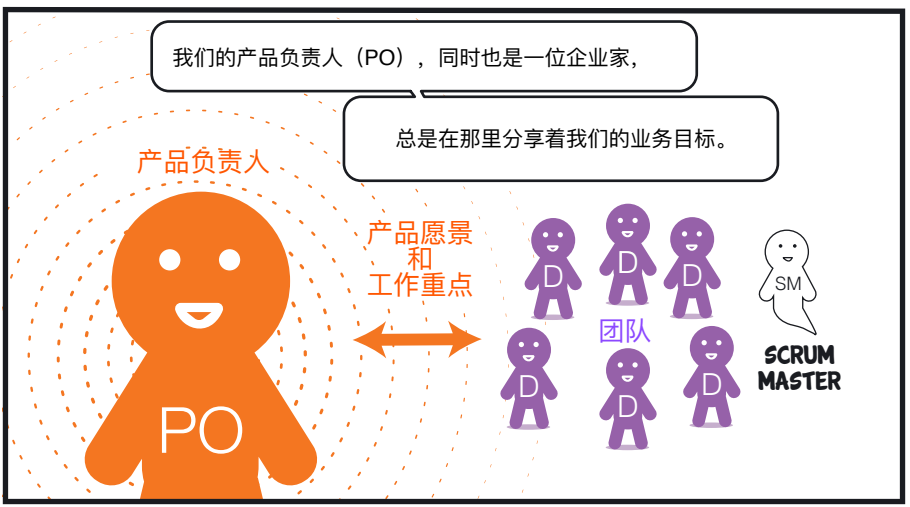
PO这个角色应该如何工作?

最初我使用Scrum的时候是在一家小公司做产品开发人员。

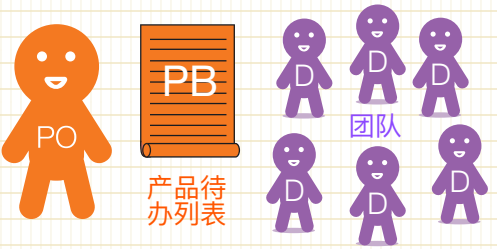
开发人员



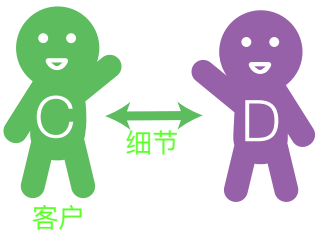


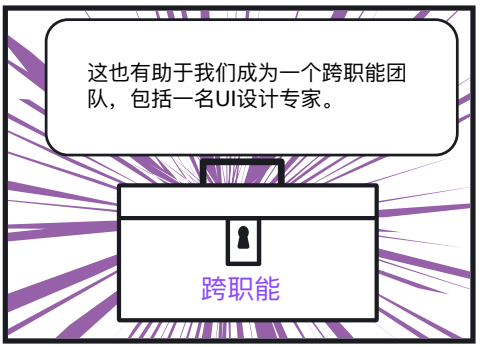


他维护着一份产品待办列表的优先级，然后我们一起对它进行完善。

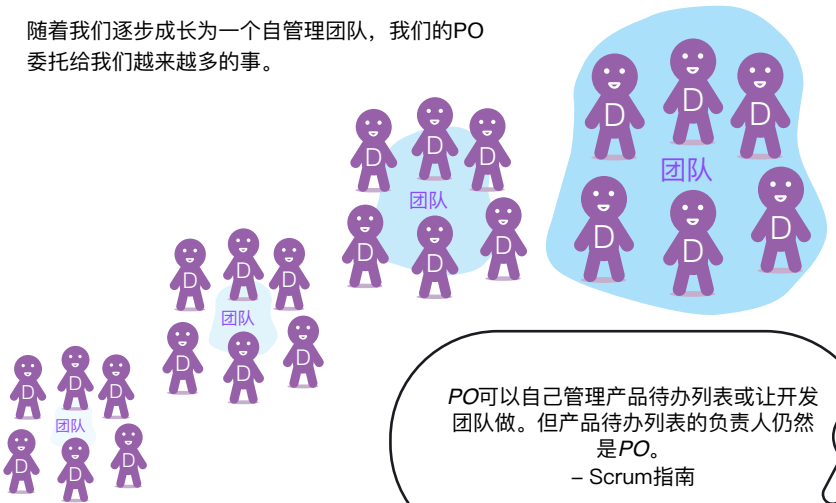


与客户的互动帮助我们了解了该做的细节部分。





随着我们逐步成长为一个自管理团队，我们的PO委托给我们越来越多的事。



*PO*可以自己管理产品待办列表或让开发团队做。但产品待办列表的负责人仍然是*PO*。
– Scrum指南

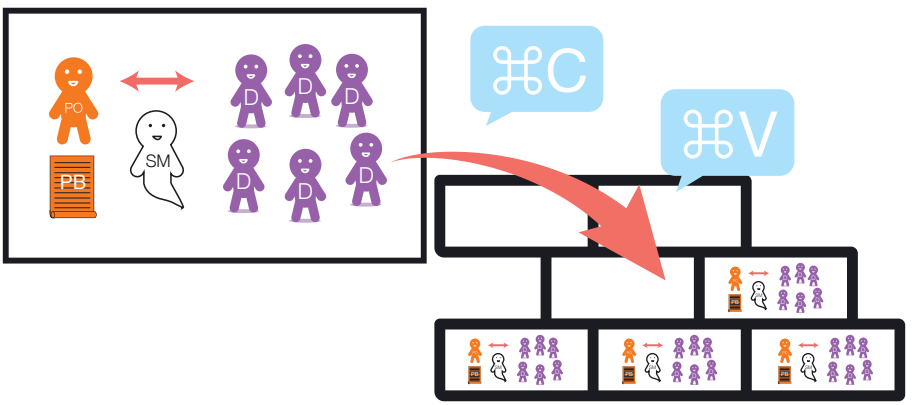
场景 *2*:

大型组织是如何曲解PO这个角色的?

当我作为ScrumMaster开始在大型组织工作时，我犯了一个错误：聚焦于团队生产力。

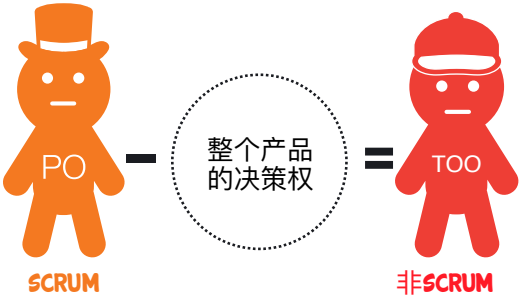
我可以在一家小公司将Scrum应用得很好，

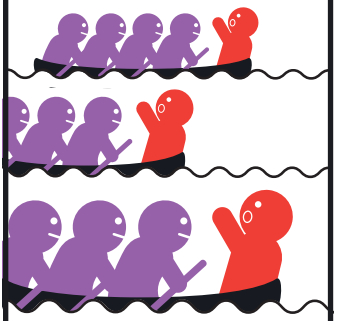
那么为什么不将相同的模式拷贝粘贴到一个更大的组织中呢?



只考虑团队生产力的话，似乎每个团队都应该有自己的PO。

但跟我曾经的PO有所不同，这些人都无权对整个产品做出重要业务决策。

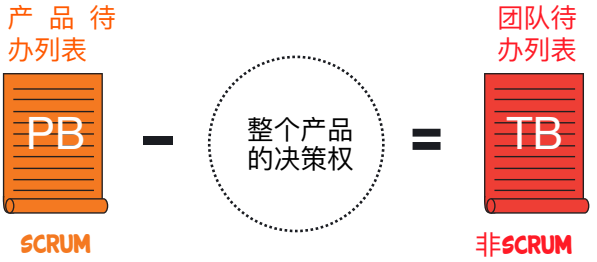




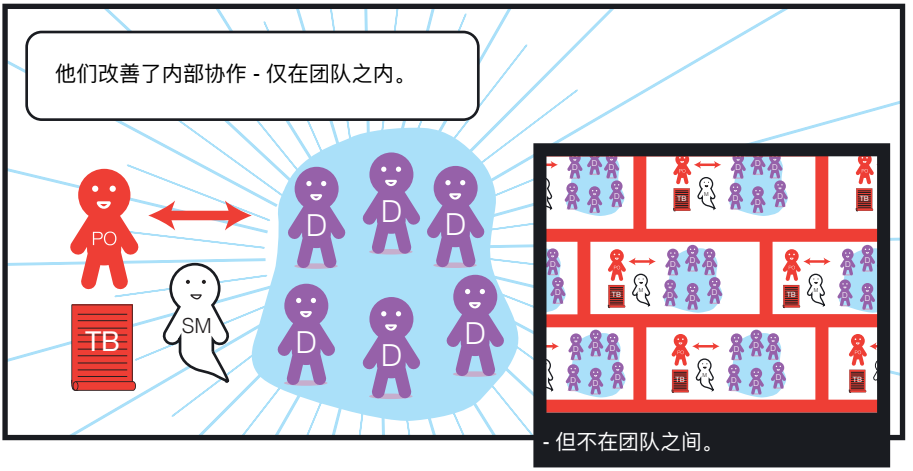
组织希望他们聚焦在团队的交付上。

这是一种常见的Scrum误用，我将他们称为团队输出负责人（TOO，Team Output Owner），因为这名字真实地反映了组织对他们的期望。

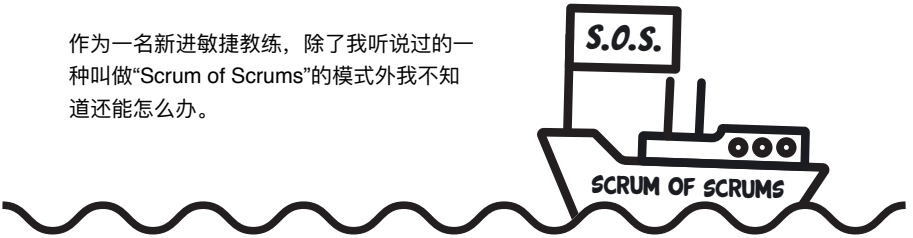
当我说团队输出负责人（TOO）的时候，他可能就是您所担当的角色或您称之为“PO”的人。

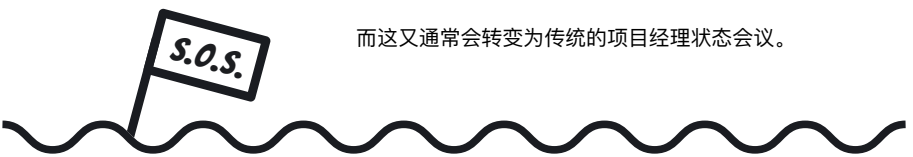


虽然没有权限确定整个产品待办列表的优先级，但每个团队输出所有者（TOO）都负责一个团队待办列表（我有充分的理由这不是一个Scrum工件）。



作为一名新进敏捷教练，除了我听说过的一种叫做“Scrum of Scrums”的模式外我不知道还能怎么办。

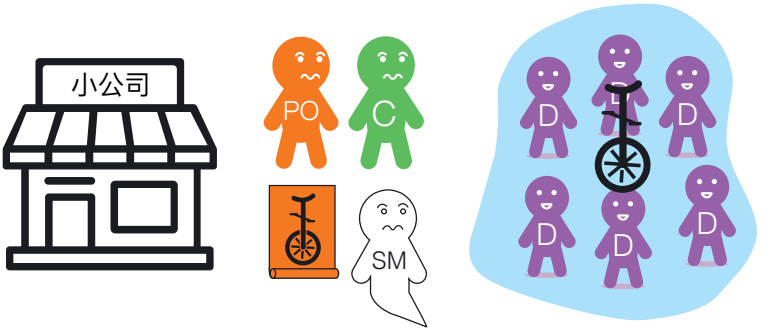




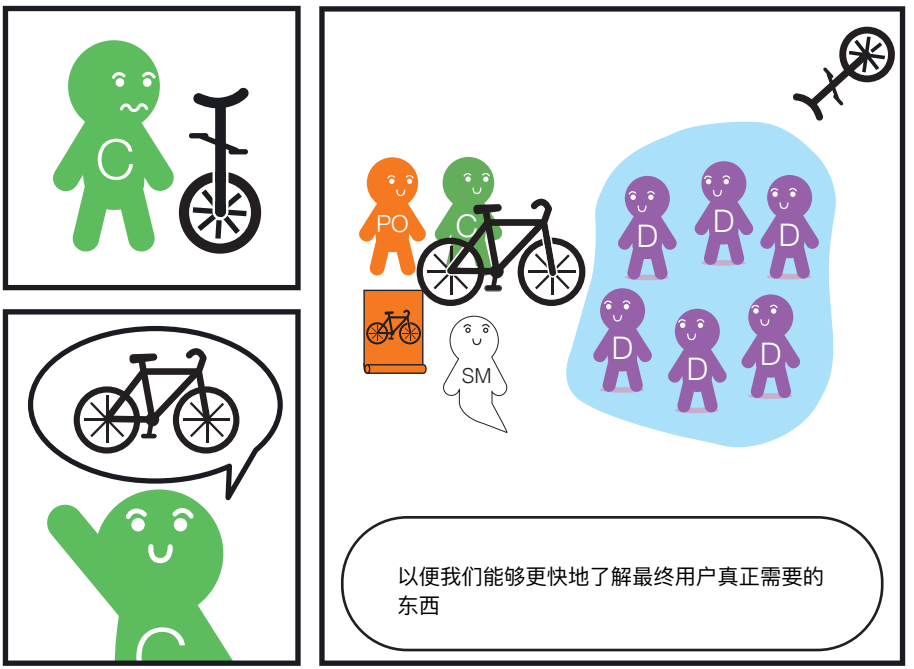
而这又通常会转变为传统的项目经理状态会议。

场景 *3A*：

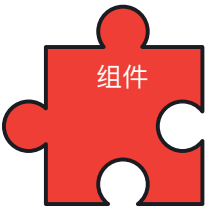
曲解PO角色是如何延缓客户反馈的?

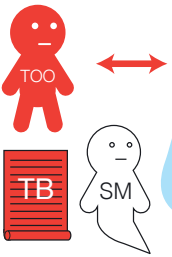


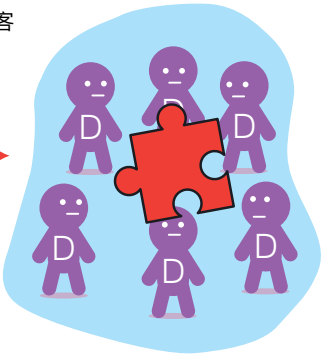
真正的Scrum团队会努力在每个迭代内开发可交付的产品



大型组织中的团队有时只会开发组件，而非端到端及以客户为中心的功能。

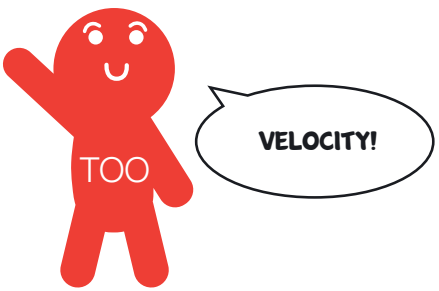






聚焦于实现中层管理者的内部目标而非更大的业务目标可能会导致团队去尝试增加其输出，

这个输出所对应的度量有时被作为“开发速率”。



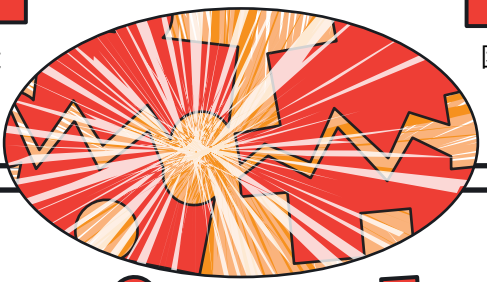
组件

团队1制造

但后来如果发现不同团队制作的组件彼此不匹配会发生什么？

组件

团队2制造



COMPONENT

团队1制造

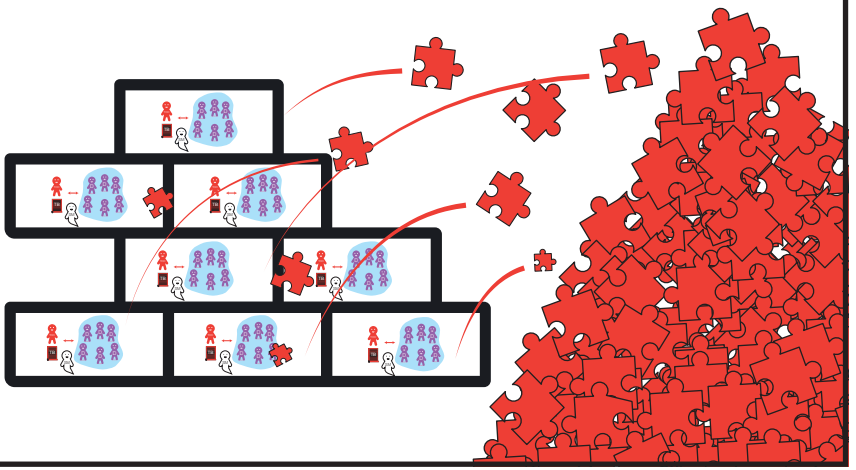
COMPONENT

团队2制造

人们有时认为团队应该聚焦于增加其产出。聚焦点很容易转移到“开发速率”这一类度量项上。

但总体而言，聚焦于此可能会增加集成问题，事实上还会延迟我们获得客户响应的能力！

如果我们的工作必须与其他东西集成才是可交付的功能，那么我们可能需要更长时间才能获得真正的客户反应，并从中学习及相应地调整方向。



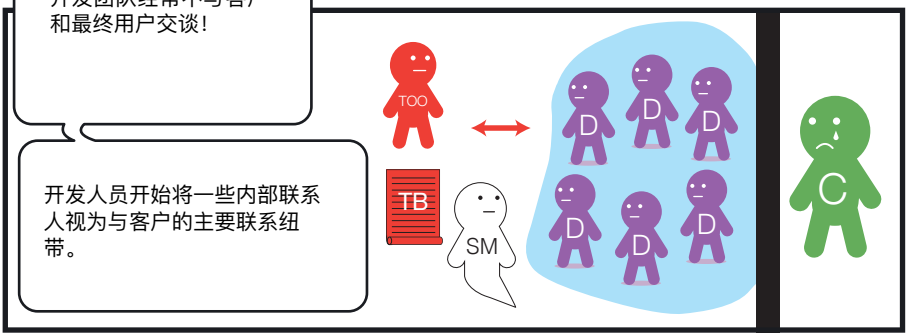
更多地聚焦组织层面的内部目标意味着更少地聚焦业务目标。

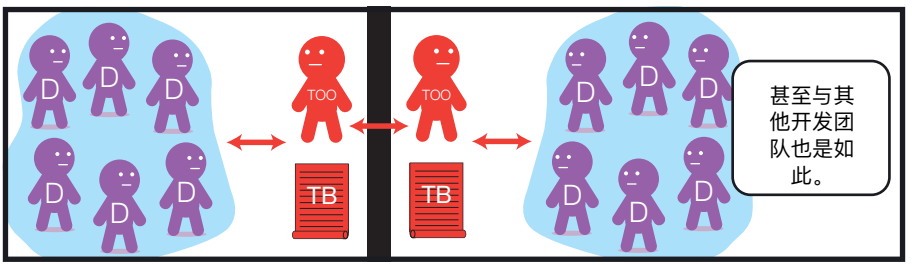
场景 3B：
曲解PO角色是如何降低团队的开发积极性和客户同理心的?

大型组织的另一个怪象是

开发团队经常不与客户和最终用户交谈!

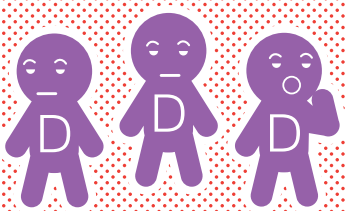
开发人员开始将一些内部联系人视为与客户的主要联系纽带。





甚至与其他开发团队也是如此。

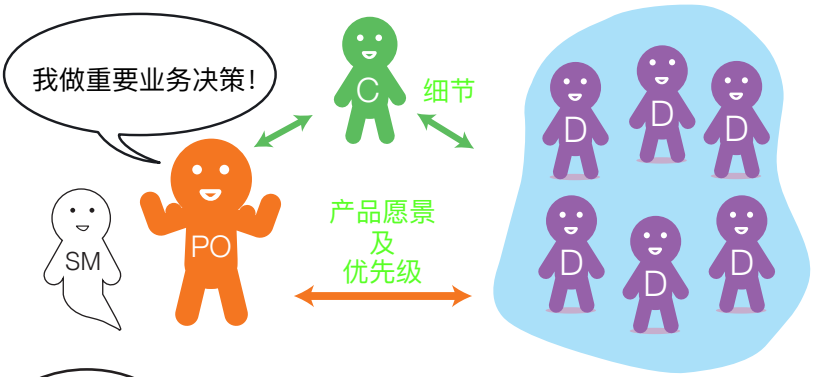
但是使这些内部联系人满意并不像与使用我们产品的真实客户交谈那样有趣或有效。



场景 *4A*：

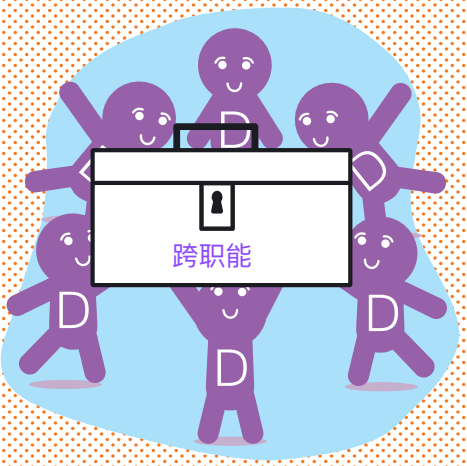
一个真正的PO是如何交付最大客户价值的?

真正的PO可以做出重要业务决策，并在我们了解了越来越多的信息时逐步发展产品愿景。



他们不仅仅只会对着需求文档码代码。

开发团队是跨职能的，拥有创建产品增量所需的所有技能。
-*Scrum*指南



架构

分析

测试

设计

编码

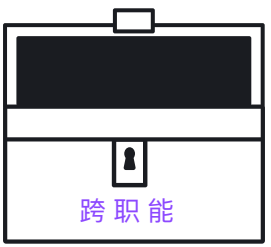
集成

用户体验

开发

业务领域

作为一个跨职能团队，他们应该拥有或学习所有必要的技能来完成高级工作。



Scrum将跨职能与简化角色结合在一起。

+

无角色

消除角色将流程所有权置于团队手中。

有些人认为Scrum添加了诸如燃尽图，斐波纳契数列，开发速率之类的东西。

SM

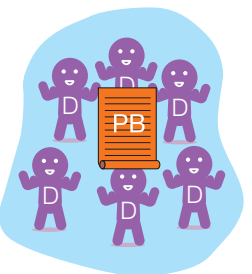
*Scrum*不认可开发团队成员的头衔，不管承担哪种工作他们都叫开发人员。
–Scrum指南

这些东西在某些场景下可能有用，但它们并不是Scrum的一部分。

Scrum的主要好处在于消除了流程和角色。

当PO做出重大业务决策时，他或她可以对产品待办列表进行相应的修改。

组织没有结构性变化，团队已经养成了工作在高阶问题上和学习新事物的习惯。

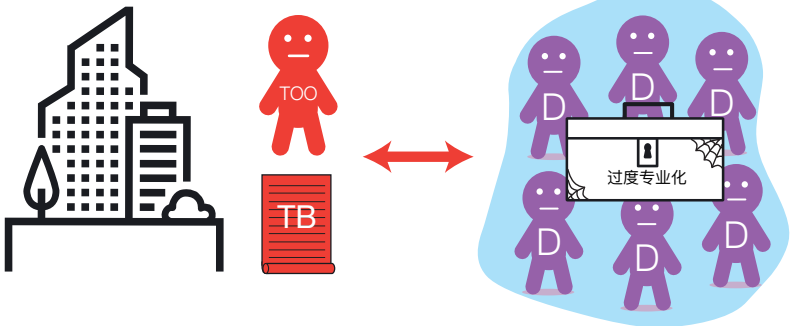


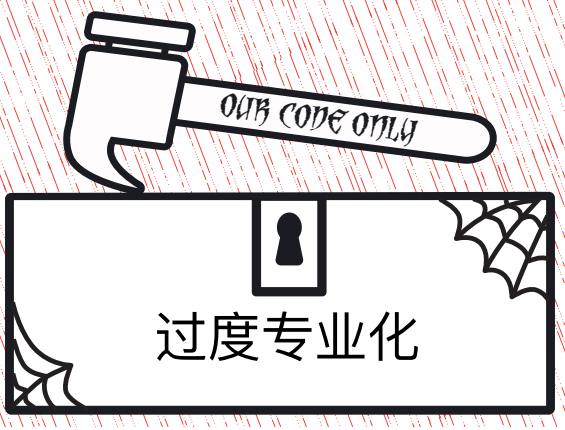
场景 *4B*:

曲解PO角色是如何降低价值交付的?

大型组织中的团队存在在某个代码领域中过度专业化的风险。

其他团队经常无法理解他们的代码。





有时他们的技能会过时，他们也不知道还有其他有用的方法和技巧。

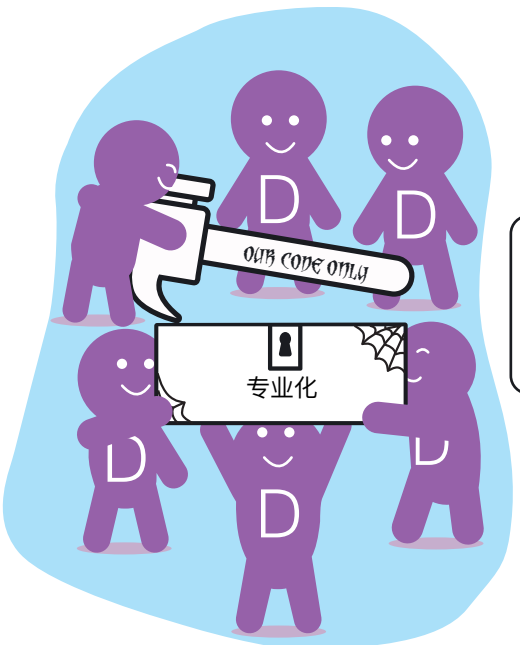
团队输出负责人将努力对团队待办列表进行排序以提供最大价值。

$5 ITEM
$1 ITEM
$8 ITEM
$5 ITEM
$7 ITEM
$10 ITEM
$5 ITEM

WE'LL HAVE THE ILLUSION THAT SCRUM IS WORKING.

$10 ITEM
$8 ITEM
$7 ITEM
$5 ITEM
$5 ITEM
$3 ITEM
$1 ITEM

TOO



团队也会认为这是最佳优化，

可能是因为过于聚焦于输出，也可能因为工作在“你说啥我做啥”模式下过于舒服。

即使是ScrumMaster们，管理者们，教练和培训师们也可能被这种**本地优化**所欺骗。

但是客户价值却是少于应该实现的，因为在我们的团队没有学习到的其他待办列表中隐藏着更重要的工作！

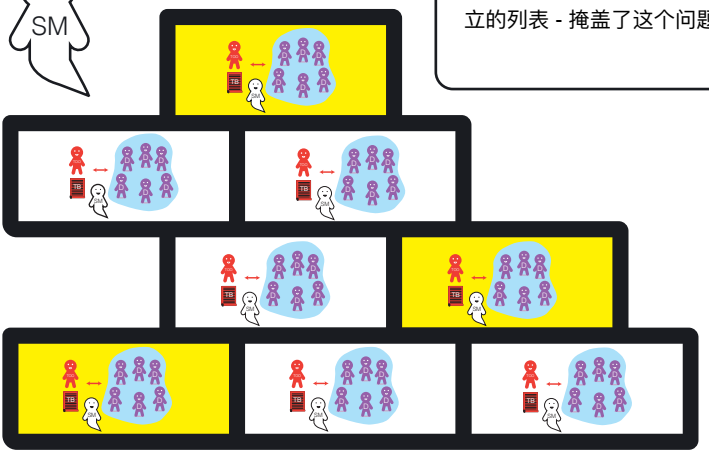
$8000 ITEM
$8000 ITEM
$7000 ITEM
$7000 ITEM
$7000 ITEM
$6000 ITEM
$6000 ITEM

只有少数团队正在做最有价值的工作。

SM

我们团队的最重要的工作可能不如其他团队没有时间开始的工作重要！

保持各自的团队待办列表 - 各自独立的列表 - 掩盖了这个问题。



我们对客户的影响也受到限制。

对于这种组织来说，根据环境变化改变方向（这也是敏捷性的关键点）将是困难的。

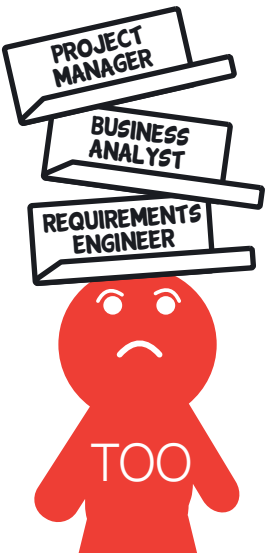
场景 5:

作为一个团队产出负责人糟糕在哪儿?

团队产出负责人被困在中间。

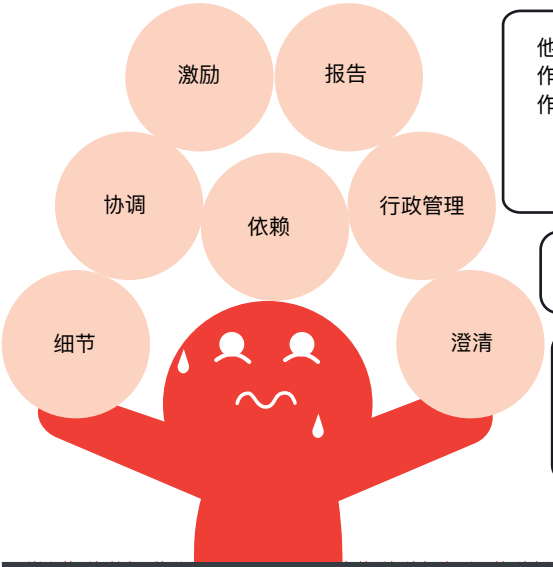
他们是无权做出重大商业决策的中间人。

有时，他们对自己无法控制的事情负有“责任”。



他们承担着可能成为需求工程师，业务分析师或项目经理的风险。

他们可能会努力将完美的“用户故事”写成详细的需求，然后在它们不完美时做澄清。



他们可能会去协调与其他团队的工作，跟踪依赖关系，执行管理工作，状态报告，

或推动团队获得更多产出。

团队产出负责人承受着来自其他地方的压力，无法成为一名企业家。

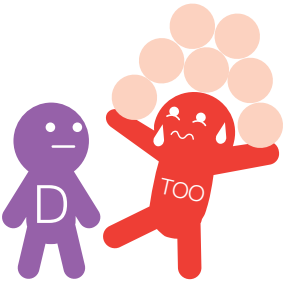
我从未想过让*PO*成为负责需求的业务分析师。
-Ken Schwaber

https://kenschwaber.wordpress.com
/2011/01/31/product-owners-not-proxies/

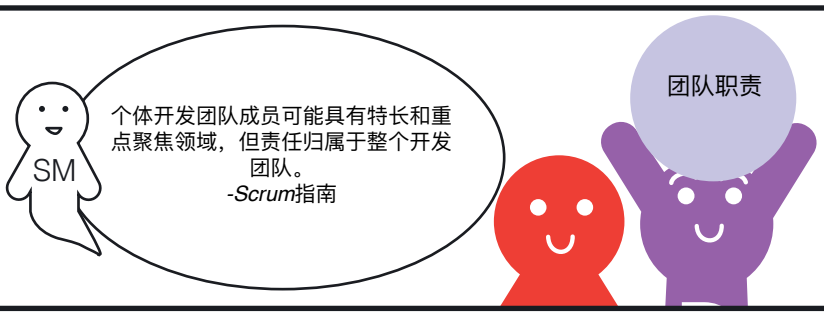
场景 6:
我们如何在增加团队自组织和跨职能的同时帮助深陷这一角色的人?

我尊敬那些认为TOO这个角色是适合他们情况的实践者。

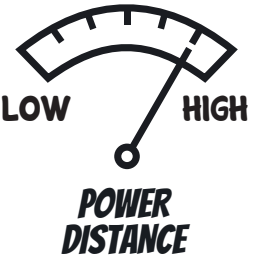
我尊敬那些认为TOO这个角色是适合他们情况的实践者。他们为了实现让某个人可以能够立即给出反馈，已经做了很多。当然我们都想要这样的效果。

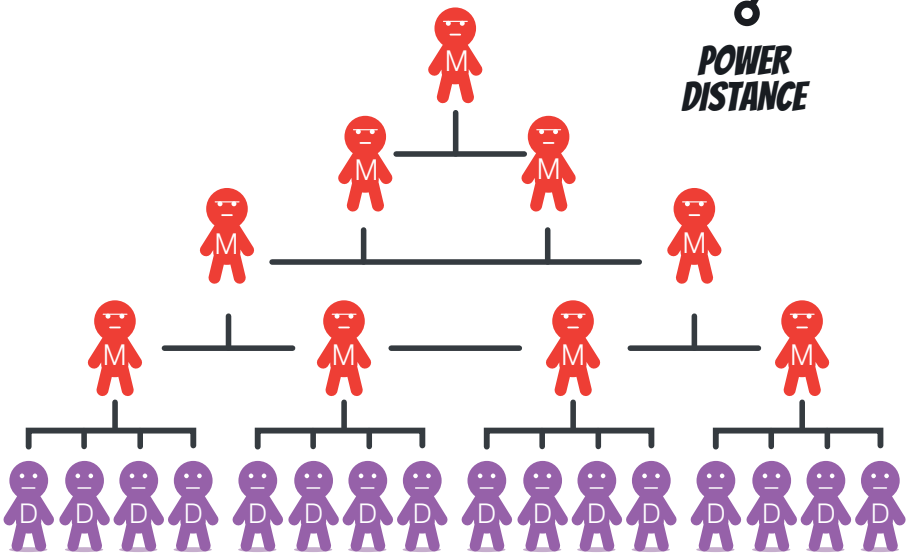


拥有这样效果的地方已经拥有了最小权力差异的社会环境，人们自然地分担责任，就像无角色团队被要求做到的一样。



但设定团队产出负责人这样的角色对组织有害，因为它们非常等级化。





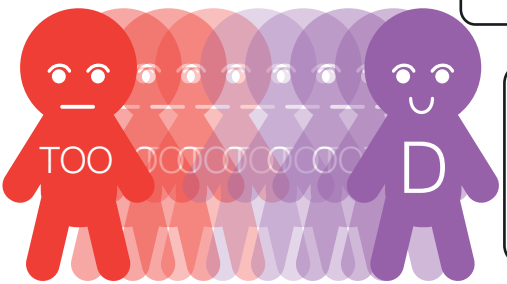
对于他们来说，与角色相关的身份差异是事实存在，

我不会告诉他们去假装所有角色都是平等的，但是一个角色比其他角色更平等。

如果一个角色比其他角色更平等，那么所有角色如何做到平等？

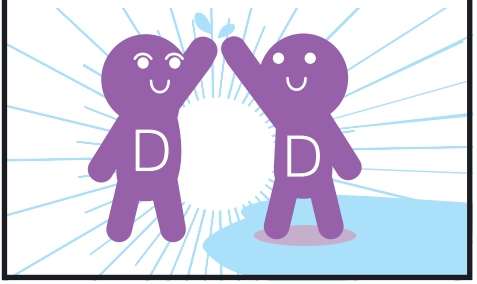
为了降低添加不必要等级结构的风险，我想强调Scrum只有三个角色的原因，以及无角色团队自组织的力量，

… 当 Scrum Master和管理层创建一个允许它的环境时。

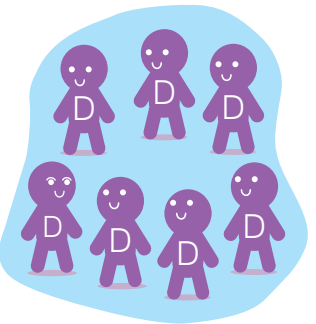


为什么我们需要担心这个人的加入而没有赋予他任何特殊的角色以与他人区分开?

然后，团队自己就可以弄清楚谁会在每种情况下做什么，而不是受到外部强加的过程的限制。



团队自管理在没有角色的时候发生。这有什么可怕的?



他们是自组织的。没有人（即使是 *Scrum Master*）有权告诉开发团队应该如何把产品待办列表变成潜在可发布的功能增量。
-*Scrum*指南

请将你想象到的可能会发生的什么坏事情写信告诉给我，我会制作关于这些问题的后续视频。

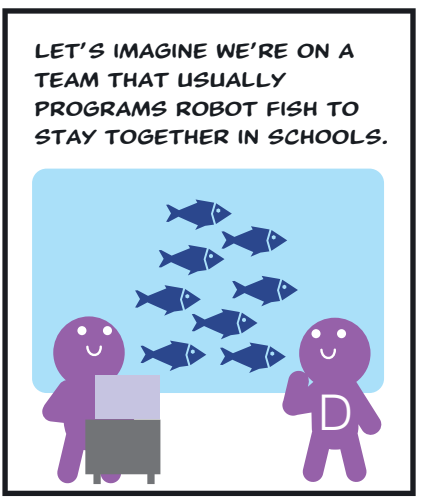
场景7:

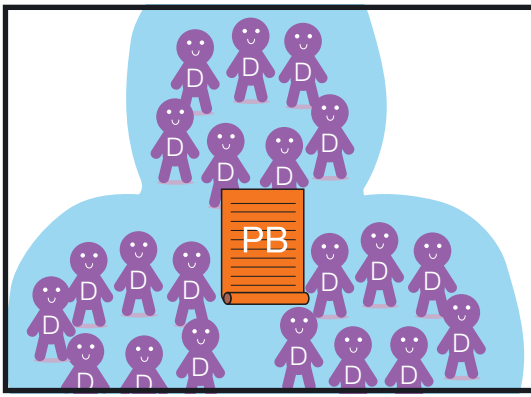
为何引入类似于“首席PO”这样的新角色并无必要?

从长远来看，你的组织可持续的竞争优势就是具有比对手更快的学习能力。
- 彼得圣吉

如果我的公司试图与这些人一起开发产品，并且我们的长期生存依赖于学习如何聚焦于最重要的事情，我会考虑将最重要的问题记录在一个真实的产品待办列表中，并要求产品开发人员聚焦于那份待办，

而不是过去对他们来说最简单的事情。





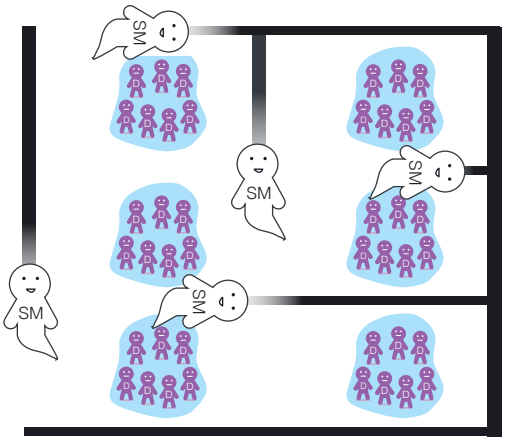
当然，团队不会互换。

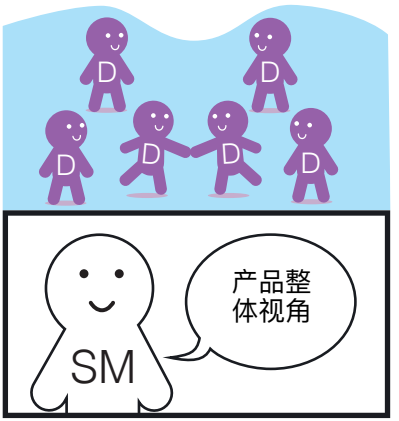
这是他们可以互相解决的问题，因为他们自己决定哪些团队将在每个Sprint中执行哪些产品待办列表项。

一个产品只有一个产品待办列表用于描述下一步产品开发工作。
*-Scrum*指南

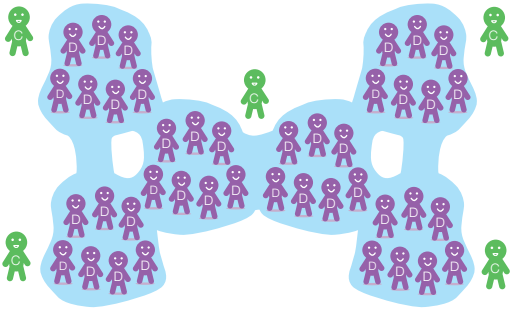
为了帮助培养整体产品观念，我要求Scrum Masters帮助消除团队之间的障碍，

并教导产品开发人员，跨团队边界的协作现在是产品开发人员的责任。



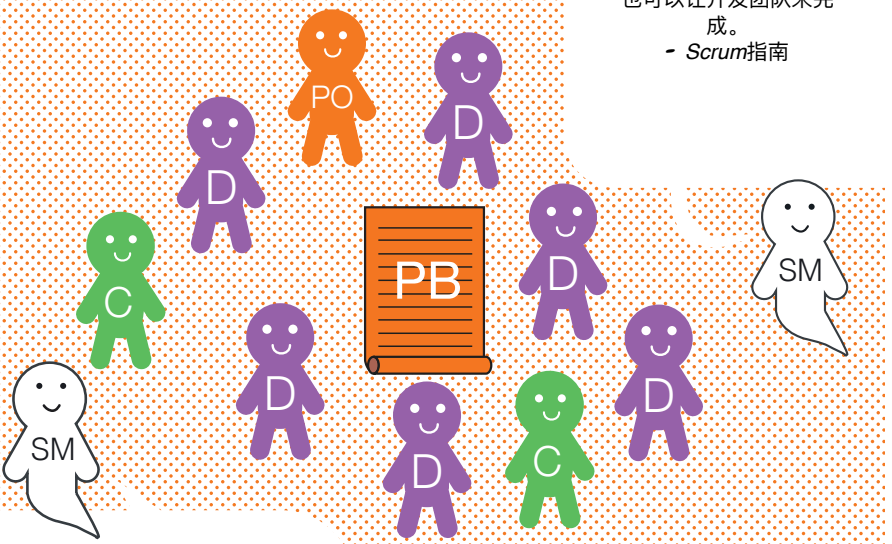


我还希望消除阻止开发人员从客户那里获得详细说明的障碍，这样我们就不再需要一勺一勺地喂他们了。



事实上，现在开发人员将帮助管理产品待办列表。

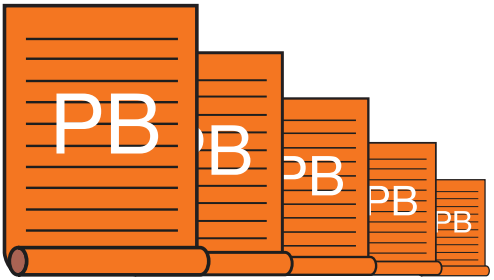
产品负责人可以亲自完成产品待办列表管理工作，也可以让开发团队来完成。
- *Scrum*指南



每个人必须与客户和用户面对面沟通以建立共情并获得洞察。
- JEFF PATTON

所有这些都可能需要数年时间才会按照我描述的方式进行，但这是被证明过的。

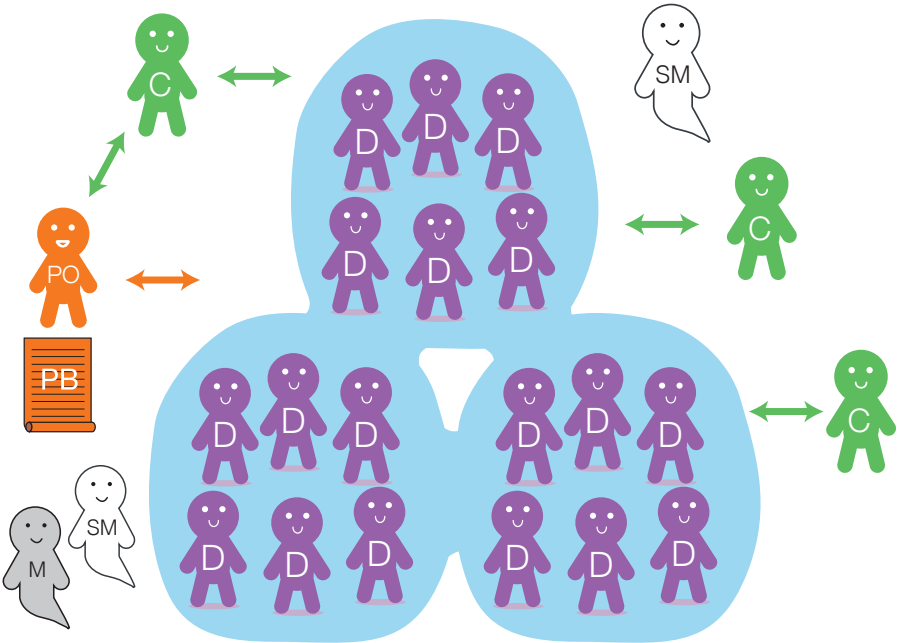
我们的产品待办列表应该会持续不断地变化



我们需要有真正权威和愿景的人通过调整这些待办项的顺序来最大化我们开发工作的价值。

我们不需要为我们工作优先级的最终来源来发明一个新名字，因为它真的并不是一个新角色。

我们称这个人为**产品负责人**。



PO

一个产品负责人和
一份产品待办列表
描述**做什么**。

C

开发人员可以直接跟客户沟通，但要避免撒手不管。

沟通的主要内容包括产品方向、优先级、业务策略和产品愿景。

开发人员与客户沟通的主要方面是用户的需求细节。

D

开发团队成员可以同时担任业务领域专家、UI/UX设计人员、前任团队输出负责人（TOO），等等。

团队间的协调是流动的、无固定角色的。团队们决定**怎么做**。

M SM

现在ScrumMaster和管理人员工作在另一个层面上，不再做协调其他人的工作。他们废除了有害的政策，并开始传播系统思考。

如果你想学习更多如何去应对这个故事里描述的问题，请：

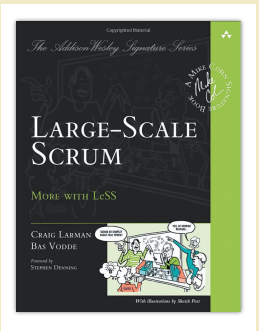
观看

https://youtu.be/1BZf_0a7W94.
(链接中的那个字幕是0, 而不是零.)

访问

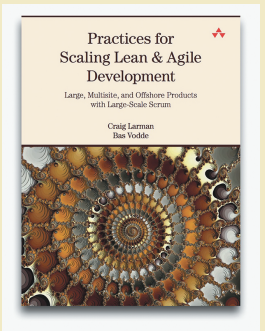
https://less.works.

先读这一本



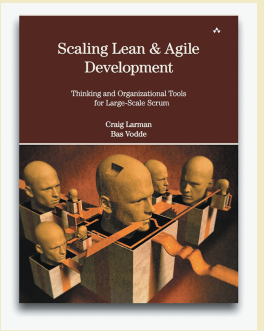
Large Scale Scrum: More With LeSS, Larman/Vodde (2016).

再读这一本



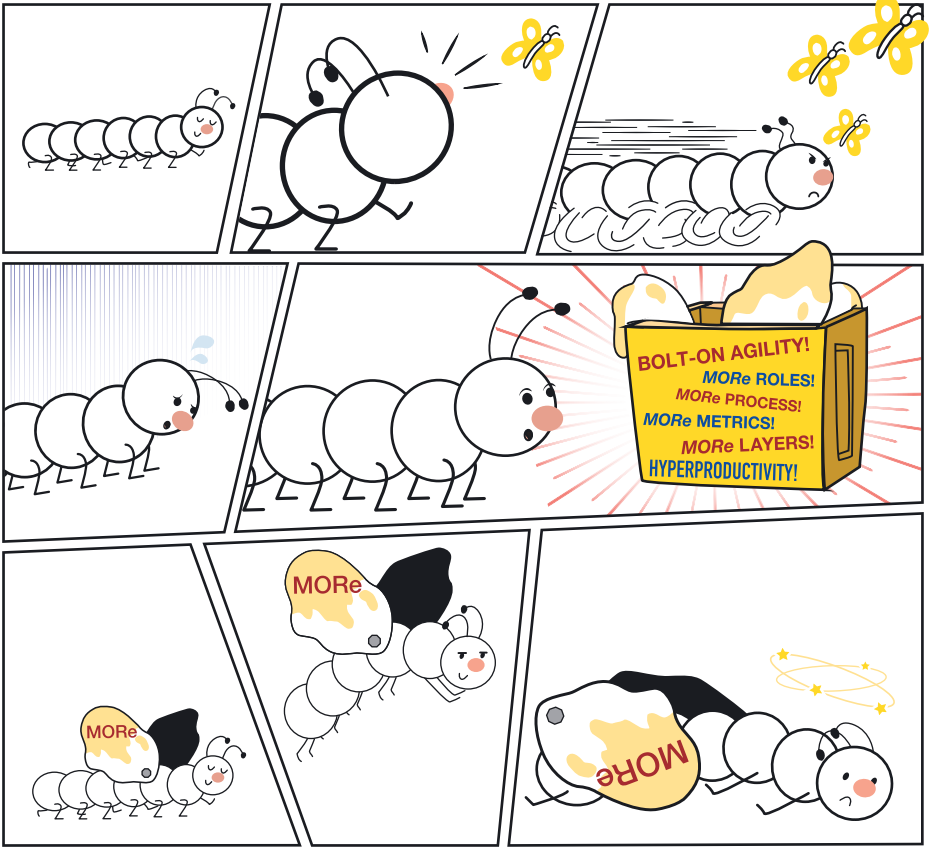
Scaling Lean & Agile Development, Larman/Vodde (2008).

再读这一本



Practices for Scaling Lean & Agile Development, Larman/Vodde (2010).

MJ (MICHAEL JAMES)
写了不少东西，
包括那份著名的Scrum Master Checklist
http://ScrumMasterChecklist.org 。
可以给他写email: mj@seattlescrum.com
或者tweet他 @michaeldotjames.



fans of LeSS

fansofless.com

你如果已经厌烦了MORe，试试LeSS吧。