spotify是如何设计产品的

产品开发并不简单。事实上，大多数产品开发努力到最后都失败了，并且最常见的失败原因就是开发了错误的产品。

Spotify是一个瑞典的精益创业项目，它同时保持着一个很棒的产品交付记录。他们的产品广为用户和艺术家喜爱，并且像病毒一样传播开来：他们有超过2000万活跃用户，500万付费用户，并且用户数量增速迅猛。举一组数字说明问题，Spotify在美国这样一个已经充斥着不少音频传播软件提供商的海外市场，只用了1年时间，就把付费用户数从0上升至100万。 Spotify的愿景是在任何时候给你带来对的音乐。这意味着它将无限地接入全世界的音乐，并且在Spotify中分享音乐会十分容易；并且音乐被分享和播放得越多，那么音乐的创作艺术家们就可以获得越多的钱。几年前，Spotify以一个音乐播放器的身份诞生，如今，他们的产品演变成一个发现新音乐和在艺术家和粉丝间建立直连的广袤的平台。 这个产品的设计理念是简单、个性、有趣。甚至连Metallica（美国乐队名），这支长期以来被认为是音乐流服务的死对头的乐队，现在都称Spotify是“目前最好的流服务”并且“被它的方便所震精。” 但仍旧存在一个悖论：就是像Spotify这样成功的公司当然只希望产出人们喜爱的产品，但是只有在产品上线之后，他们才知道人们到底喜不喜欢这个产品。 那么他们是怎么做的呢？ 这篇文章的目的就是给Spotify的产品开发方法做一个高度的概括总结。

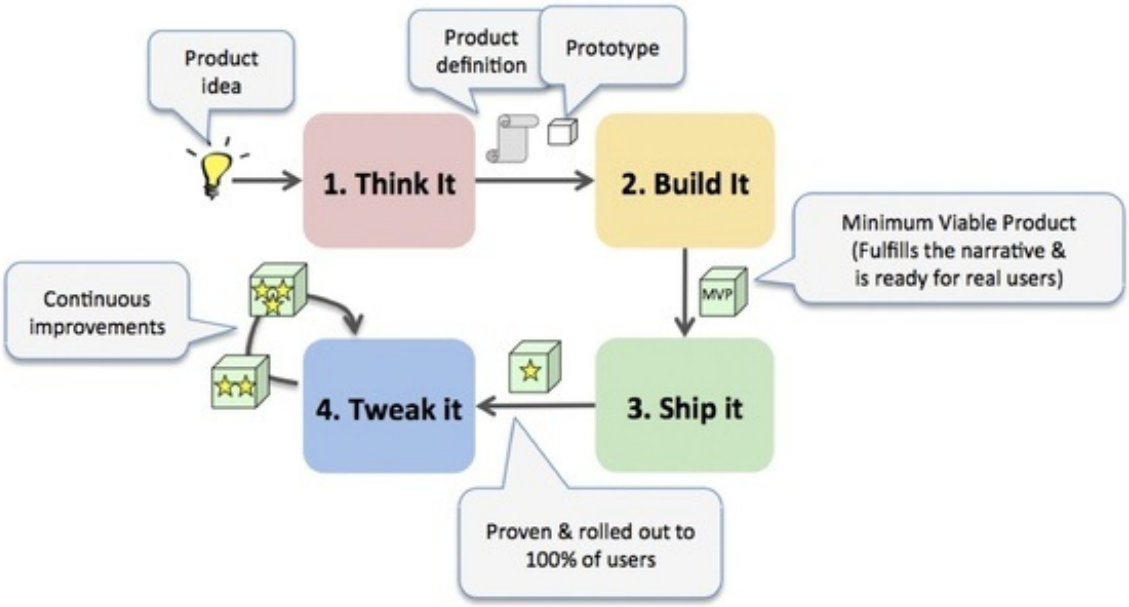
概要

我们的核心理念是：

- 我们创造革命性的产品，同时通过早期低成本的原型设计来控制风险。
- 我们直到品质过关了才会发布产品，即便已经错过了发布日期。

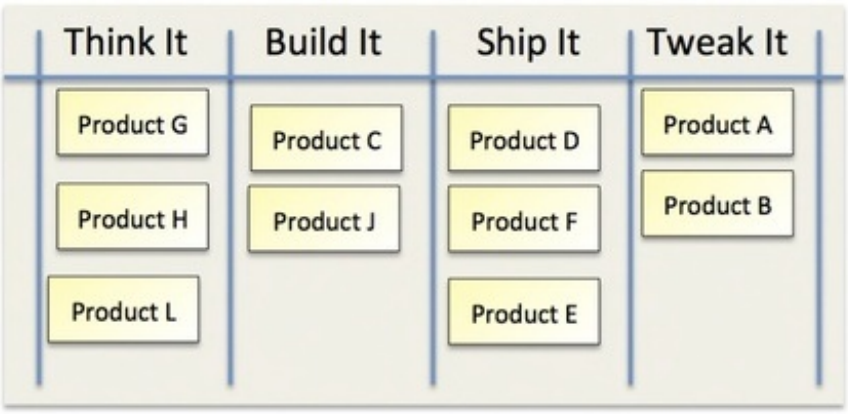
- 我们通过产品发布后虐心地一次次tweak（可理解为“调整优化”）产品，来确保我们的产品从发布起就表现优异，并且到后来惊艳得令人称奇。

所有主要的产品计划都经历4个阶段–“Think It（思考）”，“Build It（构建）”，“Ship It（发布）”，“Tweak It（优化）”。下方为一个关于从产生灵感到形成产品的整个流，以及过程中的各个阶段会产出什么玩意儿的图示。



- Think It（思考）= 整明白我们在打造何种产品，为什么。
- Build It（构建）= 开发出最小可行性产品(MVP)
- Ship It（发布）= 将产品向全部用户逐步慢慢铺开，同时进行数据检测并不断改善。
- Tweak It（优化）= 持续不断地提升产品。这是产品的最终状态，产品不断优化直到生命周期终止或产品重构（= 回到Think It）。

Spotify 拥有超过30个 squads（可理解为“小分队”，下同）和许多不同的产品，为了让公司的其他人都了解公司正在发生什么，我们用一种产品状态图来表示每个产品分别处于哪个阶段。大致如下：



我们同时也面向一些squad试行预测机制，这些squad对产品何时将会到达下一个阶段有一个日期时间上的预期，并对提供的这个阶段晋级日期范围（日期X-日期Y）负责。

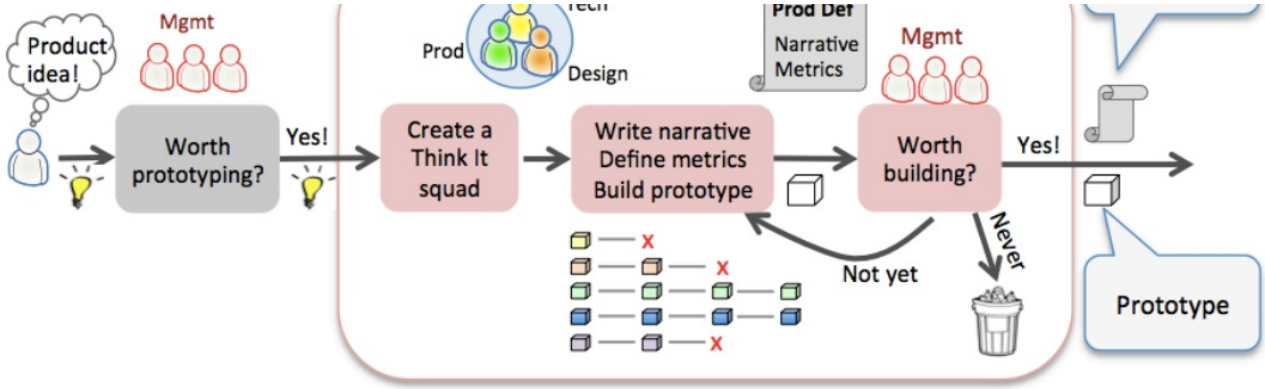
为什么是这4个阶段？

构建一个错误的产品是最具风险的事情–错误的产品无法取悦我们的用户，同时无法提升用户数以及用户留存等好的指标。我们称这个后果为“product risk（产品风险）”。 这个4阶模型帮助我们压低风险，并且快速做出产品。下面这个图表可以看出在每个阶段产品风险是如何被降低的，同时可以看到每个阶段是如何地成本密集。

我们可以看到，Think It这个阶段可以以很低的成本降低风险。同时我们也看到我们为什么要尽可能缩短Build It这个阶段（因为它消耗很高的运作成本却几乎无法带来风险的降低）。而在Tweak It阶段逐渐降低的运作成本表明，随着时间推移，产品并不需要进行尽可能多的更新，squad们可以开始继续去做其他事情。 每个阶段的周期变化多端，上面的这个比例只是一个例子而已。总的时间同样也是会变的；有些产品从孵化到产出也就是几个月的事情，而另一些产品可能要花去大半年甚至更多的时间。但是在每个阶段里，产出（即便只是内部的）都是在一个可持续性的基础上完成的。 好，现在我们仔细来研究一下每一个阶段。

Think It

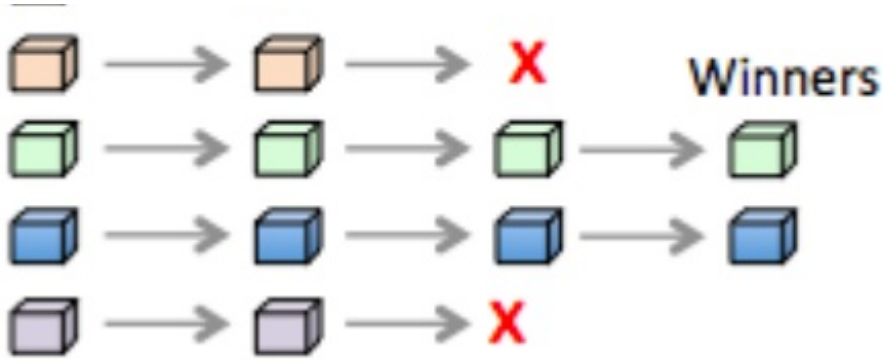
产品灵感在任何时间，在公司的任何人身上都可能诞生出来。大部分灵感都是去提升现有的产品（也就是“tweaks”），这种情况squad们只需自己实施和发布即可。 这里说的“Think It”阶段指的是某人想出了一个全新的产品创意，或者说去重构一个现有产品。



如果管理者也认为这个想法是值得付诸实践的，那么一个小型的“Think It”squad随即成立。典型的“Think It”squad一般包括一个开发者，一个设计师和一个产品经理。他们的工作就是去完善产品描述，同时构建一个足够吸引人的产品原型。

- 产品描述通常是一个用来回答如下问题的一个简短的文档：
- 我们为什么要去构建他？谁会从中受益，如何受益？
- 我们期望这个产品去提升哪些关键指标？这些指标可能关于播放了多少流音乐，下载量有多少，注册量有多少等等。
- 我们的预期是怎样的？我们如何去判断这个产品是否成功？
- 产品会带来“阶段性的改变”（阶段性改变指的是，在预期中这个产品将带来至少双倍的既选指标上的提升）吗？如果在我们的期望中，这个产品只是较小地提高了指标，那么要去构建它，最好有更强有力的理由，比如一些战略方面的原因等。

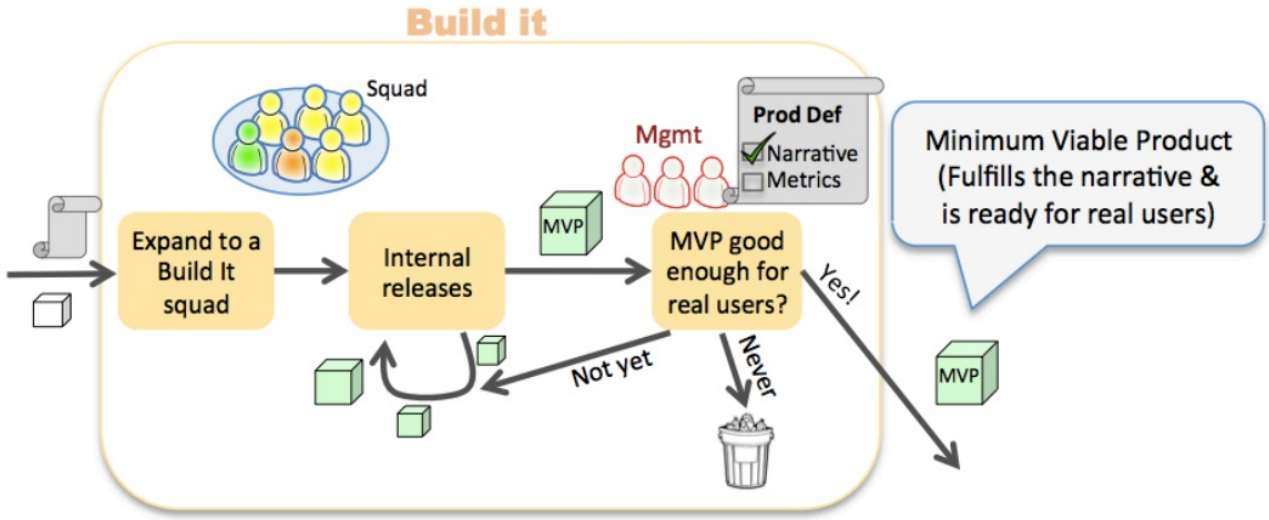
产品描述不是必备的文档，也不是所谓的项目计划。它不包括特性清单、预算、资源计划等等。它更像是一个用数据说话（数据驱动）的意愿陈述。 产品描述中最重要的部分就是故事性描述。我们要向世界讲什么故事？新闻稿又是什么样的呢？ 举个栗子，Spotify的“Discover（发现）”标签是最近的一个产品。介绍一种发现音乐的更好方式。看！你最喜爱的艺术家刚刚分享了一首歌给你。我们让艺术家们和粉丝们从未如此靠近过。喜欢一个艺术家？那就去follow（关注）他，并与朋友们分享你的新发现吧。 另一个例子是有“Radio you can save（你可以保存的电台）”说法的免费移动电台。这种情况下，我们会用谷歌的关键词广告去尝试几种不同的描述，看看哪种描述最吸引人。关键在于，这个故事性描述在产品构建前就写好了！这样我们可以在产品构建前就确定这个产品足够吸引人。 另外，“Think It”squad会构建许多不同的原型来传递产品的感官上的体验–同时会有“低保真”的纸面原型和“高保真”的可运行的原型（上面跑伪数据源之类）。这时几个内部焦点小组会用来辨别哪一个原型最好地传达了它的产品精神（那个故事性描述），直到我们不断缩小范围，最后只剩下几个胜出的原型。

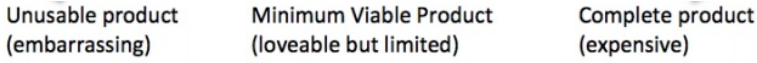


这是一个没有截止日期的迭代过程。只有当我们可以拿出一个足够吸引人的故事性描述和能够传达出它的可运行的原型，这个产品才是值得去构建的。我们无法决定这个产品前期会花去多少时间。完成的定义：Think It阶段直到管理者和squad共同认同这个产品是值得构建的（或者这个产品永远都不值得构建，故应该被舍弃）则标志完成。 这是一个主观上的决定，它并没有硬数据作支撑。Ship It阶段才会产生硬数据，所以我们希望尽可能快地到达Ship It阶段。

Build It

在这时，Think It squad开始扩张，以组建一个更加长时间存在的squad（有时是好几个squad），这个squad具备开发、测试、发布一个真实产品的所有需要的能力，这个squad会长期负责这个产品，不仅仅是在Build It这个阶段。 Build It阶段的目标是构建一个MVP（Minimum Viable Product，最小可行产品，注释见上文），即一个对于发布给外部用户，传达某些产品理念来说已经足够好的最小可行产品。这个最小可行产品利用一些例如Scrum、Kanban以及eXtreme Programming的敏捷开发方法迭代构建。

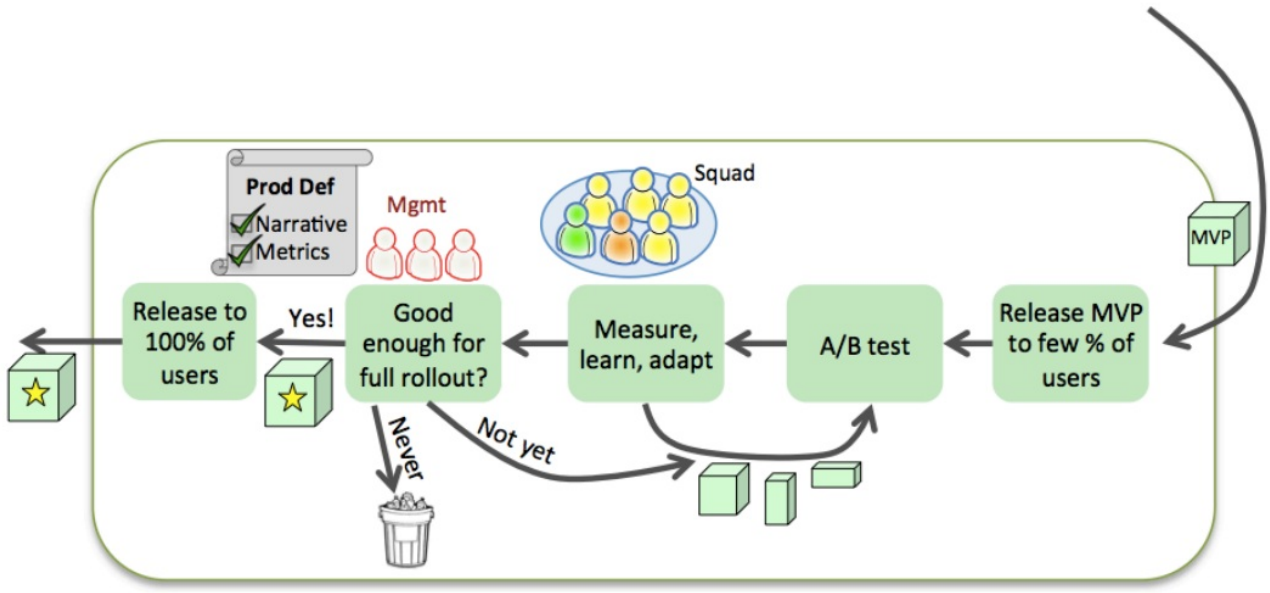




一方面，我们不希望在发布产品前构建一个十分完备的产品，因为这个过程会延迟我们获取数据的时间。在我们把真实的软件发布给真实的用户之前，我们是无法确定我们是否处于正确道路的，所以我们需要尽可能快速到达Ship It阶段。另一方面，我们不希望产出无用的或令人沮丧的产品。人们总是期待Spotify产出优秀的软件，并以此来给我们打分，即便我们说目前软件仅仅是beta版本或alpha版本。 于是squad需要找到可以实现最基本的narrative（故事性产品描述，产品精神），并且可以取悦用户的他们可以做的“最小的可能的玩意儿”。或许形容它的一个更贴切的词是Minimum Loveable Product（最小可爱产品）。自行车对于没有更好的交通工具的人来说是可爱的有用的产品，但是距离它的升级版，摩托车的差距还很大。但我们的确需要实现基本的产品描述，产品精神。否则，我们的判断标准就会被误导：“嘿，我们做出了一个轮子，并且没有人去用它，所以说这个产品是失败的，我们不应该去打造自行车的剩余部分了！” Think It和Build It阶段的关键不同在于，在Think It中，我们尽可能快，可以走遍各种捷径并且不用担心技术上实现的质量；而在Build It中，我们要写产品级的代码并且需要保障质量。 完成的定义：Build It阶段，直到管理者和squad共同认为目前这个产品已经实现了最基本的产品定义，并且对于开始发布给真实用户已经足够好的时候标志着结束。 面对Moment Of Truth（真理到来的时刻），我们已经准备好了！

Ship It

Ship It阶段的目标是逐渐将产品铺开给所有用户，同时进行数据检测，确保产品在自然环境下，也能够履行它的设计初衷。

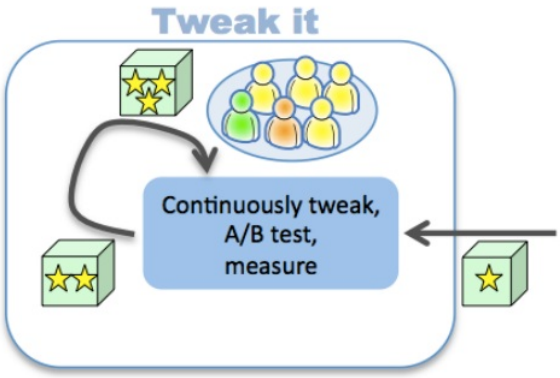


Squad一开始只将产品发布给全部用户中的一小部分（一般1-5%），以便收集数据。 如何将这些用

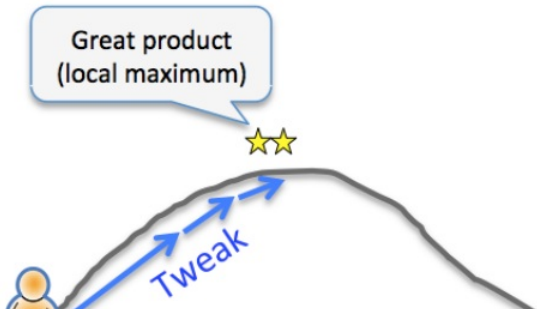
户的行为，相比于其他的95-99%呢？ 还记得吗，我们在Think It阶段定义了一些关于这个产品的预期，现在我们可以最终测试一下这些预期是否依然保持正确，并且对产品进行一些必要的迭代提升。一开始我们应该不太容易一下就做对，在这个模型中花的力气也有不少是不必要的。 当管理者和squad共同认为产品正在小范围的用户群中发挥预期的效果，我们就可以逐渐地在更多的用户中铺开产品，同时仍旧需要做数据监测和产品提升。这可以给我们时间去处理一些业务方面的事物，例如硬盘容量，监测，脚本部署，扩展性等等。 完成的定义：当产品对所有用户都可用时，Ship It阶段完成。 注意一下，这时并不意味着产品已经“feature complete（特性、功能完全）”，完成了Ship It阶段只是意味着产品（最小可行产品+必要的改进）已经被100%铺开而已。其实并没有所谓“feature complete”的说法，因为产品即使在Ship It阶段之后还会持续进化。

Tweak It

这是最为关键的阶段，因为产品们在这里抵达重点（除非在路上他们被抛弃），并且产品在这里花掉它生命周期中的大部分时间。



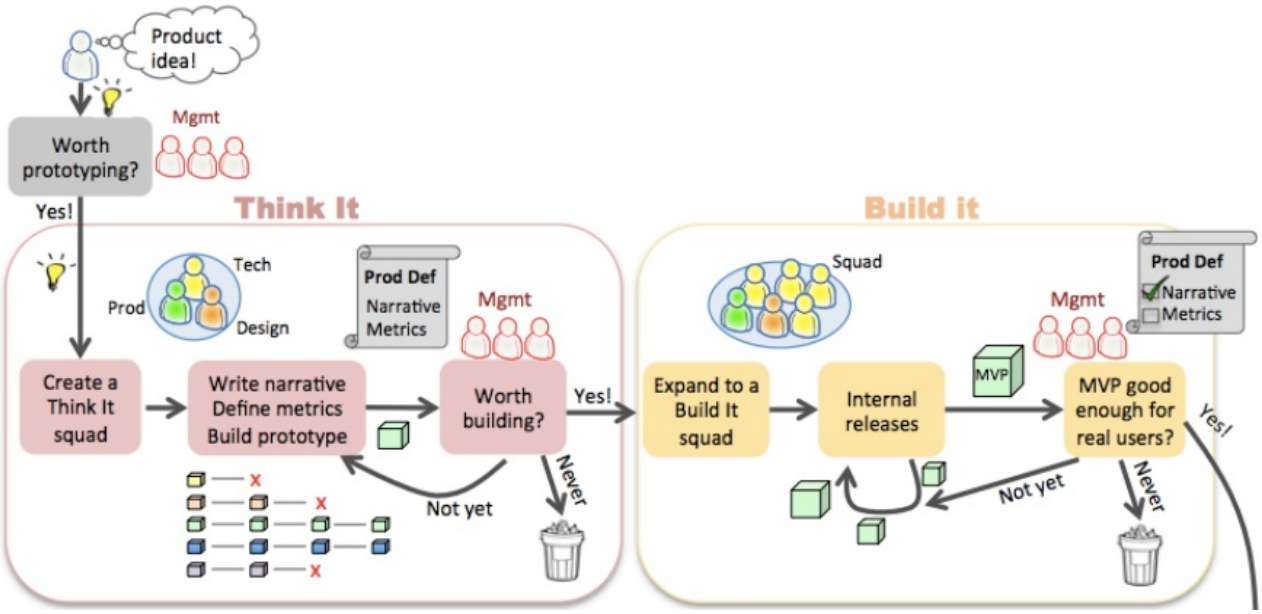
产品现在已经产出成果，并且对所有用户可用。虽然在某种程度上它已经在Ship It阶段证明了自己，但总还是有很多提高的空间。Squad继续展开实验，在跟踪数据的同时，进行A/B测试以及改善产品，这可以包括重要的新特性也可以是较小的调整。 然而，未来的某一天，squad可能会到达一个产品的收益递减的点。这时产品已经很好，最重要的改进已经完成，并且改进新特性带来的收益率也将不再那么吸引人。转看监测数据，新特性和改进也不见得会带来很大程度的飞跃。 那么这就意味着产品已经趋近于一个“极大值”了。

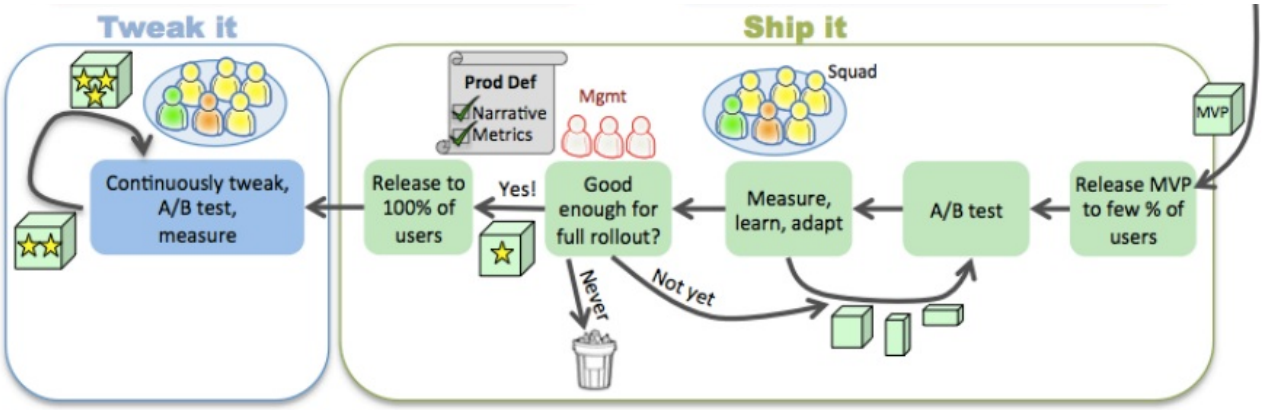


在这个时候squad和管理者就会讨论：我们是不是甘于止步于这座山的山顶，或者去寻找一个更高的巅峰？在前一种情况下，squad可能会逐渐地转移到其他产品的工作上去。在后一种情况下，squad可能会回到“Think It”阶段去考虑重构这个产品或者让这个产品去开拓国际化道路（或者至少是一个更高的山峰…）。

这种情况的一个实例就是spotify.com这个网站。该网站在2012年夏天我们决定去重构它之前已经修修补补了4年。现在这个网站已经在以一种完全不同并且出奇高效地方式来传达Spotify的愿景。

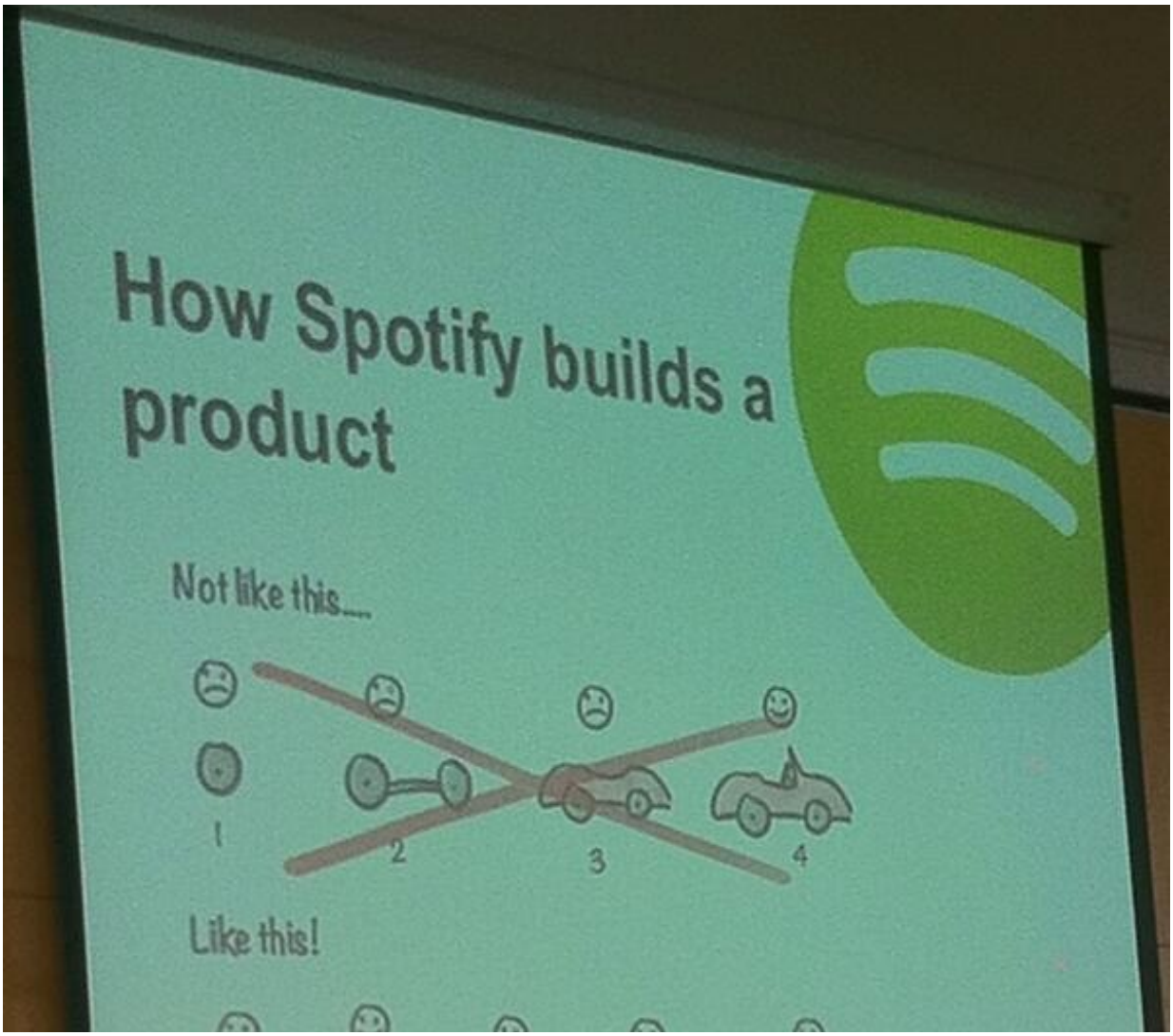
总览图

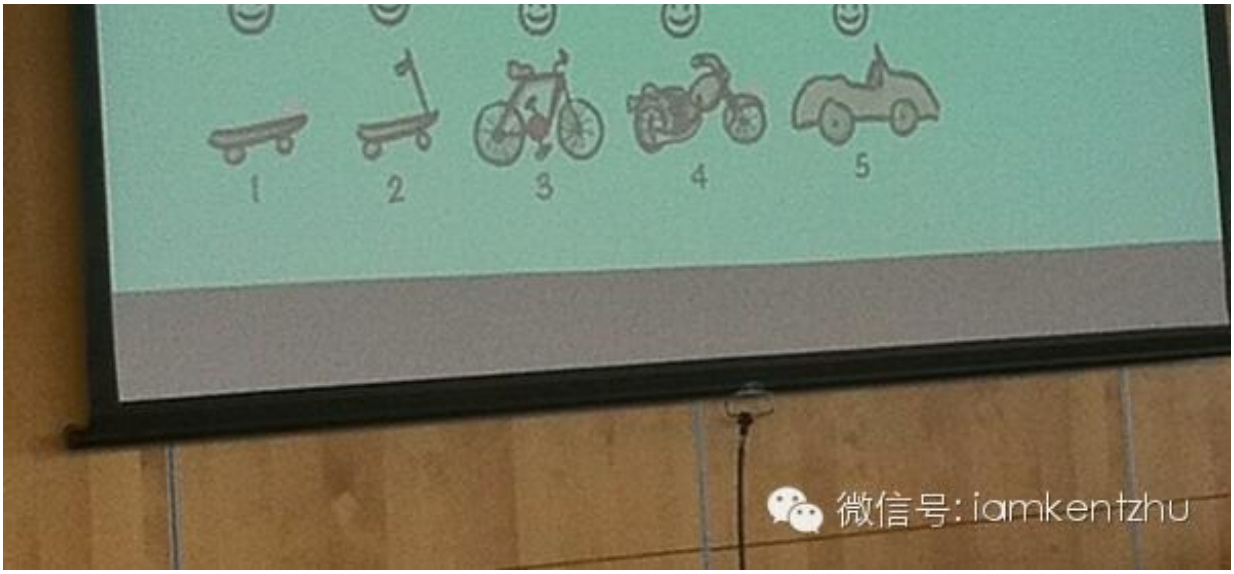




最后的话

希望你能享受这篇文章！ 如果模型中的某些部分让你觉得“我去，我已经早就造这些东西了，我们已经酱紫做几十年了好伐”，那么你八成是对的。这个模型所述并不是新玩意儿，屌玩意儿。它只是在讲述那些有用的东西–这玩意儿新老其实并不重要。我发现这种实例的结合还是非常振奋人心充满能量的，我也希望你可以在期中找到在你的环境中也有用的东东。 – 在Twitter上，有位哥们po了一张PPT的图，应该是这个主题的演讲，其中用一个更形象的图总结了什么叫「敏捷开发」，附录如下：





原文：How Spotify builds products 翻译：@Omnibingo

人人都是产品经理（woshipm.com）中国最大最活跃的产品经理学习、交流、分享平台