06 | 大厂都在用哪些敏捷方法？（下）

宝玉 2019-03-09

00:00 19:33

讲述：宝玉 大小：17.91M

你好，我是宝玉，我今天继续与你分享大厂的敏捷方法应用。

在上一篇文章中，我们一起看了一下大厂和敏捷相关的一些流程规范，同时也为你留了一道思考题：

> 如果每周一个 Sprint，怎么保证每周都有交付，还能保证产品质量？

所以在这一篇中，我们就以每周一个 Sprint 的小项目组为例，看看它的日常是怎么应用敏捷开发的。

一个应用敏捷开发的小组日常

这个小组是做网站开发的，基于微服务负责网站的某一个小模块。标准配置 7 人，4 个程序员（至少有一个资深程序员，有架构能力），1 个产品经理（Scrum 里面叫 Product Owner），1 个测试，1 个项目经理（Scrum 里面叫 Scrum Master）。主要负责网站某模块的日常维护。

在分工上：

产品经理：写需求设计文档，将需求整理成 Ticket，随时和项目成员沟通确认需求；

开发人员：每天从看板上按照优先级从高到低领取 Ticket，完成日常开发任务；

测试人员：测试已经部署到测试环境的程序，如果发现 Bug，提交 Ticket；

项目经理：保障日常工作流程正常执行，让团队成员可以专注工作，提供必要的帮助，解决问题。

在敏捷开发框架下，已经形成了一些很好的敏捷实践，这个小组也是基于 Scrum 方法做过程管理，基于极限编程做工程实践，看板可视化。每周一个 Sprint。

如何完成需求和修复 Bug?

这个小组的日常工作，也是围绕 Ticket 来开展的。所有的需求、Bug、任务都作为 Ticket 提交到项目的 Backlog，每个 Sprint 的任务都以看板的形式展现出来。

每个人手头事情忙完后，就可以去看板上的“To Do”栏，按照优先级从高到低选取新的 Ticket。选取后移动到“In Progress”栏。

每周一部署生产环境

没有人愿意星期五部署，那意味着如果部署后发现故障，可能周末都没法好好休息了。所以即使程序早已经测试好了，除非特别紧急，否则都会留在下一周再部署。所以部署放在上半周，这样后面遇到问题还有足够的时间去应对。

部署很简单，按照流程执行几个命令就可以完成生产环境部署。部署完成后，需要对线上监控的图表进行观察，如果有问题需要及时甄别，必要的话对部署进行回滚操作。**但轻易不会打补丁马上重新上线，因为仓促之间的修复可能会导致更大的问题。**

像敏捷开发这样一周一个 Sprint 的好处之一就是，即使这一周的部署回滚了，下周再一起部署也不会有太大影响。

每周二开迭代回顾会议，总结上个 Sprint

每周二的早上，这个小组一般还会预留一个小时的时间，因为常规的站会完成后，还有一个**迭代回顾会议 (Sprint Retrospective)**会议，目的是回顾一下在迭代中，团队有哪些做的好的地方，有哪些做的不好的地方。

对于需要后续改进的，需要创建相应的 Ticket，加入到 Backlog 中，在后续迭代中改进完善。

例如会议上，测试人员反馈说，上一个 Sprint，开发人员上线前几个小时还往预部署的分支里面更新代码，导致测试需要重新做回归测试，但因为时间不够了，没来得及测试完整，导致上线后不稳定，建议以后不要随意在上线前，在部署分支更新代码。

对于这样的问题，可能不止一次发生，说明流程上还是存在问题。所以最后大家商定，以后如果不是紧急的修复，就不要在预部署的分支上更新，确实要加，需要和测试先确认。

如果会议中要形成涉及项目的决策，最好是通过集体表决的方式决策，尽可能避免独裁式决策。因为敏捷的原则之一是要**善于激励项目人员，给他们以所需要的环境和支持，并相信他们能够完成任务。**

每周四迭代规划会，计划下周工作

每周四早上，也需要一个小时来组织会议。因为常规站会完成后，还有一个**迭代规划会（Sprint Planning Meeting）**。这个会议是要大家一起讨论下一个 Sprint 的内容。

在开会之前，产品经理和项目经理会商量好 Ticket 的优先级，会议上，大家一起按优先级高到低从 Backlog 中选出下个 Sprint 的内容。

团队每个成员都要对候选的下个 Sprint Backlog 中的 Ticket 从 1-5 分进行打分，1 分表示容易 1 天以内可以完成的工作量，2 分表示 2 天内可以完成的工作，5 分表示非常复杂，需要 5 天以上的工作量。

这里需要注意，打分时，要大家一起亮分，而不是挨个表态，不然结果很容易被前面亮分的人影响。

评估每条 Ticket 工作量的大概流程如下：

1. 会议组织者阅读一条 Ticket，可能是用户故事，可能是 Bug，可能是优化任务。同时会询问大家对内容有没有疑问。
2. 大家一起讨论这个 Ticket，确保充分理解这个 Ticket。
3. 每个团队成员在心中对 Ticket 进行工作量估算。
4. 会议组织者确认大家是否都已经确定估算结果，确认后，开始倒数：“3，2，1”，大家一起伸出一只手，亮出代表分数的手指头。
5. 如果估算结果存在分歧，出分最高的和最低的各自说明理由，讨论后达成一致。

这种估算工作量的方法有个名字叫估算扑克，因为亮分时用扑克牌亮分而得名，但并非一定要用扑克牌。

用这种方式评估工作量有几点很明显的好处：

1. **大家积极参与，详细了解需求。**相比以前，可能只有当某个功能模块分配到自己头上的时候，才会去详细了解那部分需求，而其他开发人员可能都不了解这部分需求。
2. **工作量是由实际参与开发的成员作出评估，往往更准确也更容易被接受。**以前项目经理代为估算的模式，很容易不准确，或者让开发人员抵触。
3. **促进成员的交流和经验分享。**我们知道一般经验浅的新手估算工作量都会偏乐观，而经验丰富的老手则会更准确，通过这种方式，新手可以向老手学习到很多工作量估算甚至技术实现的经验。

所以，在经过几个 Sprint 的磨合后，一般一个团队在每个 Sprint 的产出是比较稳定的。比如说这样一个 7 人的小团队，一个 Sprint 预计可以完成 20-30 分的 Ticket。

每周五分支切割

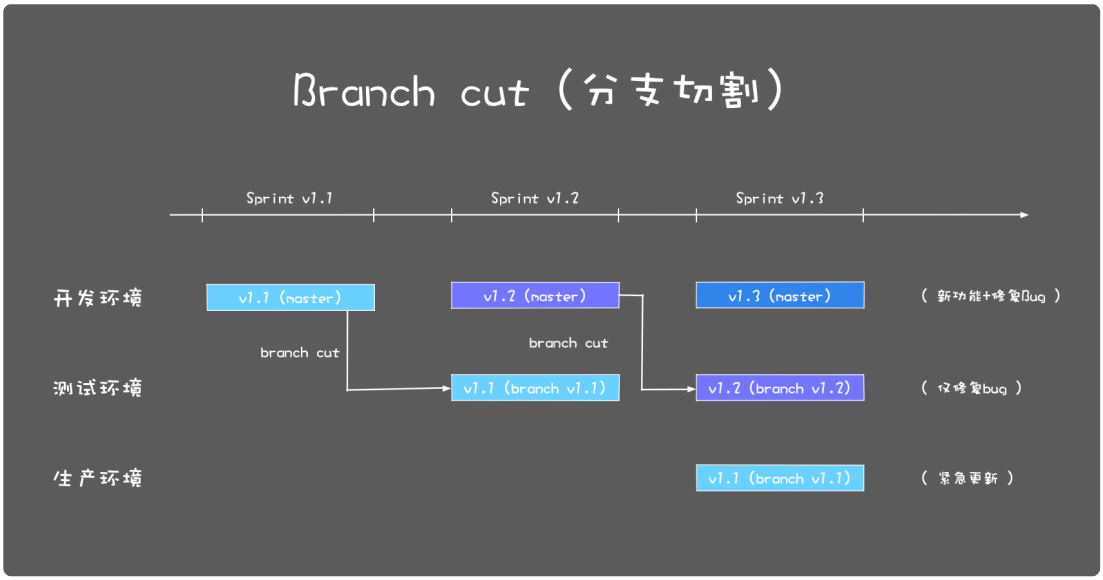
周五标志着一周的工作要结束了，所以下班之前（4 点左右），要做 branch cut（分支切割），也就是要把当前主干上的代码，克隆到一个分支（branch）上。

为什么要做分支切割这一步操作呢?

经过一周的开发，master （主干）已经合并了不少新的 PR（Pull Request，合并请求），但是如果你直接把 master 的代码部署到生产环境，肯定还是不放心，毕竟自动化测试还是不能完全代替专业测试人员的测试。

所以我们需要把 master 上的代码部署到测试环境进行测试，并且对测试出来的 Bug 进行修复，直到稳定下来为止。由于 master 还需要一直合并新的功能，所以最好的方式就是每次 Sprint 结束，从 master 创建一个分支版本出来，然后基于这个分支部署和修复 Bug。

所以需要基于主干做一个 branch cut，创建一个预部署的分支，将预部署分支的代码部署到测试环境，这样下周测试人员就可以测试新的版本。测试验收通过后，预部署分支的代码会部署到生产环境。



每周轮值

小组里面除了日常开发工作以外，其实还有不少琐碎的事情，比如每周部署生产环境，每天部署测试环境，每周的 branch cut（分支切割），回答其他小组的问题，主持每日会议（不一定需要项目经理），这些事情如果都是一个人做难免会有些枯燥。

在敏捷开发中，鼓励发挥每个成员的主动性，所以每周轮值是一个不错的方式，可以让每个人都有机会去体验一下，帮助团队完成这些事情，更有集体荣誉感和责任感。

一些问题解答

上面只是选取的一个项目小组的日常，所以估计你看完还会有些疑问，在这里我把可能的问题列一下，先行解答一下。

1. 基于这种敏捷开发的方式加班多吗?

其实加不加班，绝大部分时候和是不是敏捷开发没关系的，还是看项目组的情况。

通常来说，基于敏捷开发一个 Sprint、一个 Sprint 迭代，节奏还是比较稳定的，这个 Sprint 做不完的任务也可以顺延到下个 Sprint，不影响发布。不像瀑布模型那样前松后紧，后期加班可能性大一些。

2. 一周一个迭代怎么保证质量?

以前我在使用迭代模型开发时，一般是 4 周左右的迭代周期，2 周就是极限了，所以最开始看敏捷开发用 1 周的迭代周期，心中也有疑惑，1 周时间又要开发又要测试，怎么保证质量?

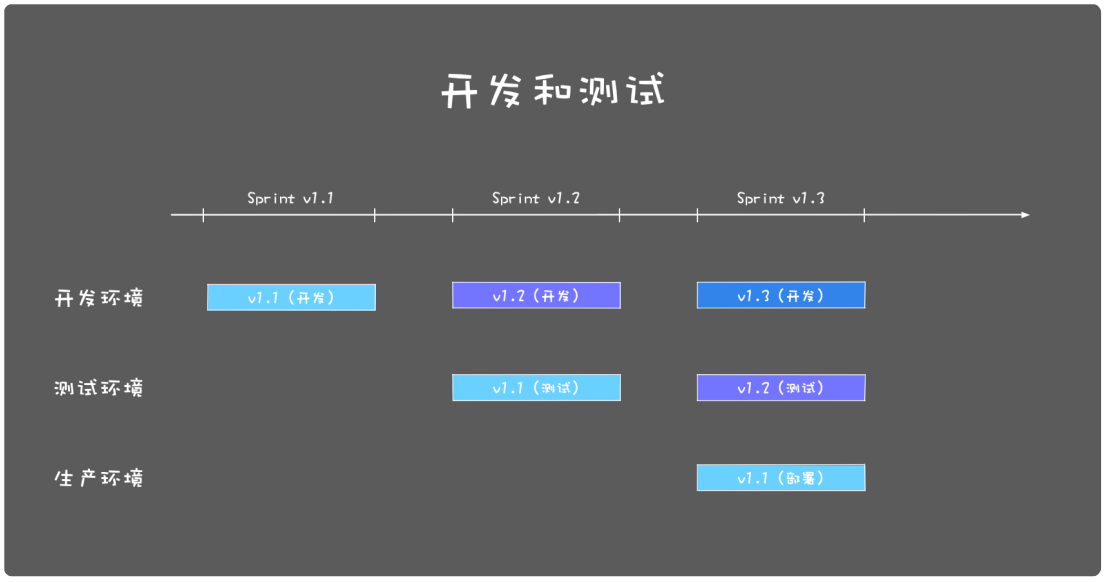
实际实践下来，发现 1 周一个 Sprint 确实可行，而且质量也可以有保障，这里面有几个因素：

（a） 有足够比例的自动化测试代码，可以很好地保证质量。当用户的主要功能都通过自动化测试覆盖时，基本可以保证主要功能流程不会出问题。

（b） 一个 Sprint 开发完成后，并不马上部署生产环境，而是先部署到测试环境，会有 1 周时间测试。

（c） 有专业的测试人员进行测试，并非完全依赖自动化测试。有时候一些大的功能更新，甚至会组织全组成员一起测试，以弥补测试人员不足的情况。

在一个 Sprint 开发结束后，并不马上部署生产环境，而是先部署测试环境测试。



也就是说，虽然是 1 周的 Sprint，但是其实还有 1 周的时间进行测试。每个 Sprint 不仅开发新功能，同步还要修复以前版本的 Bug。

这样基本上可以保证有好的质量。而且这种 1 周的迭代，可以保持每周都有内容更新，还有个好处就是每周更新的内容不多，出现问题的话，很容易就定位到是什么地方导致的问题。

3. 基于敏捷开发如何做计划?

大厂里面通常会在上一年底确定第二年整年的大的开发计划，并确定上线的时间范围，每个季度再根据情况做一些调整。

这些大的计划最终会变成具体的开发任务，一个大的开发任务，会分拆到各个部门，各部门再将任务分拆到各个项目组。基于敏捷开发的话，主要就是看把这些开发任务放到哪几个 Sprint 去做，并且确保在规定的时间范围内完成。

至于工期的估算，在迭代规划会上会对每个 Ticket 进行打分，根据分数可以预估有多少工作量，要花多少时间。

4. 如何沟通协作?

组和组之间的沟通协作，主要通过邮件、会议、内部沟通工具，最终任务会以 Ticket 的形式体现。

团队内部的话，因为都在一起，所以沟通起来很方便，每天站立会议都是很好的沟通方式。

在敏捷开发中，有一种实践叫结对编程，就是两个程序员在一台电脑上一起工作。这个一直争议比较大，但是如果用来两人一起排查一些问题，或者是资深程序员带新手程序员，则是一种非常好的协作方式。

5. 上面介绍的实践案例和标准 Scrum 有什么不同?

我上面介绍的内容，确实和标准的 Scrum 有不少不一样的地方。

首先是角色名称不一样，在 Scrum 里面是分 Product Owner、Scrum Master 和 Team 三种角色，而在这个案例中是产品经理、项目经理和团队成员，但其实只是名字叫法不一样。

还有要注意一点，就是传统的项目经理，会是偏控制型角色，Scrum Master 则更多是一种服务型的角色，主要职责是保障敏捷流程的执行，以及提供必要的帮助，很多团队的决策就是采用集体决策的方式。

另外，Scrum 有四种会议，除了前面介绍的三种：每日站会（Daily Scrum）、Sprint 计划会（Sprint Planning）和 Sprint 回顾会议（Sprint Retrospective），其实还有一种会议是 Sprint 评审会（Sprint Review）。

Sprint 评审会主要是客户审查 Sprint 的完成结果。因为上面这个小组并没有直接的客户，都是完成产品经理提交的需求，而且沟通紧密，所以没有安排专门会议。

这个小组的站立会议并不是“标准”的站立会议，Scrum 的站立会议通常只有 15 分钟，并且只有轮流发言环节。

这里增加的每天审查 Ticket 环节，主要是为了将优先级高的 Bug 修复之类的 Ticket 放到当前 Sprint，及时响应，及时处理。有的项目组没有这个环节，是由测试人员或者 Scrum Master 直接将 Ticket 放到看板。

这个小组并没有使用用户故事来开发需求，而是由产品经理事先写好需求文档。在上一篇文章里面，提到了 Scrum 采用用户故事的方式，分拆需求，减少繁重的需求文档，在实现的过程中再沟通确认需求。

这是 Scrum 推荐的一种方式，也是一种高效的方式，但并不代表这是唯一的方式。如果有产品经理，可以提前几个 Sprint 就将需求文档写详细，一样可以达到高效的理解需求的效果。

那么这样还算敏捷开发么？

其实在《05 | 敏捷开发到底是想解决什么问题？》就有讲过，是不是敏捷开发，核心并不是应用了哪个方法，而是应用的时候，是否遵循了敏捷开发的价值观和原则。

比如说非标准的站立会议效率更优，那么就应该采用非标准的站立会议；如果有专业产品经理事先做好需求分析，可以达到解释清楚需求的效果，就没必要一定要用用户故事来理解需求。

总结

上一篇文章我们讲了大厂里和敏捷相关的一些流程规范，这一篇又讲了一个小组是怎么应用敏捷开发来开发项目的。

现在看上一篇文章中我留的思考题：如果每周一个 Sprint，怎么保证每周都有交付，还能保证产品质量？想必你已经有了答案。

要保障质量，还是离不开充分的测试，不仅要有自动化测试，还要辅助一定量的人工测试。敏捷开发虽然求快，但是不代表应该牺牲质量。

其实，大厂的敏捷实践并不神秘，关键是分而治之，最终团队小，项目小，所以才可以敏捷起来。大厂会注重流程和工具的应用，通过 Ticket 的方式来管理和跟踪开发任务，通过自动化的方式来部署。

大厂的敏捷实践，一般是基于 Scrum、极限编程和看板，针对各自项目组的特点，会有所侧重有所调整，在遵循敏捷的价值观和原则的前提下，做到高效使用。

希望上面介绍的敏捷应用，能对你理解敏捷开发有所启发，帮助你优化改进日常项目流程。还有要注意的一点就是，没有万能的开发模式，只有适合项目的开发模式，最重要的还是要摸索出一套适合你自己项目特色的开发模式。

限于篇幅，对于 Scrum、极限编程和看板，我并没有展开细讲，还需要大家自己辅助看看书，我在《学习攻略 | 怎样学好软件工程？》和《05 | 敏捷开发到底是想解决什么问题？》文章中也列了一些参考书籍。

留言区有同学推荐的文章《天下武功，唯快不破—新时代敏捷项目管理之道》对敏捷开发也有很不错的讲解，推荐阅读。

课后思考

看完本篇内容，你可以将上面介绍的开发模式和你现在的项目开发模式对比，你觉得有哪些好的地方可以借鉴？你觉得有哪些做的不够好，可以改进的地方？

另外，你也思考一下，为什么文章中，这个项目没有在一个 Sprint 里面同时完成开发和测试，而是把测试放在下一个 Sprint，这样做有什么优缺点？欢迎在留言区与我分享讨论。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Ctrl + Enter 发表　　0/2000字　　提交留言

精选留言(19)

williamcai

有个疑问v1.1 v1.2 开发分支是不是从master同一个版本拉下来的吗，因为到1.2的时候，v1.1处于待测，不可能合到了master

3 2019-03-09

> 作者回复: 好问题，这个通常有两种方案：
>
> 方案一：v1.1可以不合并回master，如果有bug修复，现在v1.1上修复，然后把修复的代码同时cherry-pick到master，就是要提交两个PR，内容一样。
>
> 方案二：v1.1在部署后合并到master，这样就可以把v1.1上的修改合并到master，但是可能会有不少代码冲突
>
> 各有优缺点，具体怎么做还是看项目组的决策。

Felix

读了老师这篇文章，给我最大的启发就是扑克估算；我之前的做法是让具体开发自己先估算，我再基于他的估算结果根据自己的认识进行微调，虽然这估算经过两个人，但这种形式还是有估算不准的情况下周周会我就要确定新的流程；后续我会让小组内成员加上我一起针对Ticket进行估算，充分讨论后达成一致
还有一个问题问一下宝玉老师，我对于结对编程的概念不是很明晰；一个冲刺，分派给两位同学开发，他们相对来讲都很资深，分工明确，互相配合，但是分别在自己的电脑上开发……这种是否是结对编程呢

2 2019-03-09

> 作者回复: 赞，可以先实验，看看扑克估算是不是适合，如果好的话就可以进一步固定下来。
>
> 你说的那种不算结对编程。
>
> 结对编程（英语：Pair programming）是一种敏捷软件开发的方法，两个程序员在一个计算机上共同工作。 一个人输入代码，而另一个人审查他输入的每一行代码。 输入代码的人称作驾驶员，审查代码的人称作观察员（或导航员）。 两个程序员经常互换角色。

一路向北

这篇学习完后，对敏捷的实际操作有了更深的理解。
对于小公司小团队的项目，因为项目经理，产品经理都是身兼数职，是否有更好的实施方式呢?
目前认为的难处：1. 作为项目成员的程序员可能还需要去做项目经理或者产品经理的工作，2. 项目交织在一起，有新的项目在做，也有原先项目的维护工作或者新的需求。这样的情况在实施敏捷开发的时候应该如何最大化的发挥敏捷的优势?

1 2019-03-11

> 作者回复: 项目经理、产品经理兼多个项目是正常的，也没大问题。
>
> 但是让程序员同时兼做开发和项目经理工作就很不好，因为项目经理需要更多全局掌控，而一旦要花精力在开发上，很难跳出具体的开发工作，会极大影响项目管理工作；项目管理工作也会频繁打断开发，造成进度延迟。

> 所以我建议应该有专职的项目经理，不应该让程序员兼职项目管理。
>
> 新旧项目交织并不是问题，可以放在一个项目一个Sprint里面一起管理，也就是同一个Sprint里面有维护的Ticket，也有新需求的Ticket，只要保证开发人员同一时间只是做一件事，而不要几件事并行，就可以最大化发挥敏捷优势。

十斗簸箕

请问老师对于C++开发有什么好用的单元测试、继承测试框架推荐?

1 2019-03-11

> 作者回复: 这个问题我真帮不上你，因为我C++不懂，你得自己去网上问问别人。
> 帮你发了条微博问问，希望有用：https://weibo.com/1727858283/Hko9VyVbI
>
> 更新：微博上大家都推荐gtest，应该不错。

编程国学

我们组三个人，每个人负责一个项目，这些项目都有交叉，一直采用领导提需求(界面设计、功能需求)，分工编码，自己写，自己测，几乎无deadline，全靠自觉，如果说模型，增量与迭代组合，项目交付之后，用户使用很少(政绩项目)，快5年了，寻求突破之道，如果留下，怎么改变?

1 2019-03-09

> 作者回复: 这种情况，从内部改变是有点难的了。
>
> 估计平时也不忙的，如果可能的话，可以写点开源程序，写点自己的项目，在外部寻求突破吧。

SOneDiGo

关于课后提问:
我觉得可能是第一个sprint一时间还不能写完完整的代码可以去跑测试，所以只能放到下周...如果执念于第一个sprint也要做测试，可能项目代码没能好好完成，测试出一堆bug，那么这个强求的测试可能没什么意义了，反而还要回来再改bug,违背了敏捷的理念

缺点:如果说在安排的时候过于专注在开发上，有些bug不能及时在第二个sprint安排前发现，导致sprint2的进展出现困难，也违背了敏捷的理念

1 2019-03-09

> 作者回复: 👍
> 是的，测试需要时间的，如果功能开发星期五才完成，那么下周一部署就没时间测试。
>
> 线上Bug的修复优先级是最高的，其次是预部署环境的Bug，高于新功能开发的优先级。

cljloves

才看完了BOB大叔的《代码整洁之道-程序员的职业素养》，里面也有立会，比手指预估，Sprint回顾会议等。
但是我现在的公司开发人员组成是，Android与iOS各1名，后端3名，Web前端1名，分任务的话，总觉

得难以实施，也就后端比较能实现比手指预估这种敏捷方式。三前端都只有1个人，预估就一个人说了算的感觉。苦恼...

2019-03-11

> 作者回复: 其实这样做，即使一个人一样有意义的：
> 1. 可以帮助一起理解清楚需求
> 2. 会议上公开的估算，本质上也是一种口头许诺，结果会更严谨准确
>
> 所以没什么好苦恼的，可以先试试看。

我来也

同学们的总结和疑问都好专业啊，标出了重点，也找出来值得深挖的地方。
老师的回复很好的对文章内容做了补充。
这篇文章收获颇多。

2019-03-10

> 作者回复: 是呀，说明大家都有很多自己的思考。
>
> 文章毕竟篇幅有限内容有限，现实中的项目远比想象的要复杂，关键还是看如何灵活运用好这些知识到项目中。
>
> 希望你也可以多多留言分享🤝

K战神

谢谢玉哥，我现在找到一点感觉，
首先要把握原则，在自己规划的迭代周期内，
可以进行调整但是不能打乱节奏，
而不是像瀑布模型那样根据任务划分周期，
敏捷是周期内能应对多少需求和测试等工作。。找到一个节奏也是很重要的一个方面。

2019-03-09

> 作者回复: 是的，敏捷的迭代里面时间是固定的！下一篇还会讲一点关于时间和范围的内容。

K战神

玉哥好，我们在现在的瀑布模型开发中，
很担心的就是需求变更和需求理解不足。
如果在开发过程中需求变更了，按照我现在的开发模式就是项目经理预估组没评审，测试再评审，修改工期，但是一般小的改动可能加班就干完了。懒得再走流程。
敏捷开发中遇到这种需求变更着急上线的，影响了原来的开发节奏该怎么进行取舍？

2019-03-09

> 作者回复: 谢谢，总算没叫叔叔😄
>
> 在敏捷开发中，通常迭代比较小，所以大多数时候可以放在下个Sprint。

> 如果确实需要放到当前Sprint，也没问题，到时候影响到其他Ticket的开发进度，其他Ticket会顺延到下个Sprint。记住一个原则：敏捷开发里面，范围是可以变的，时间不能变，一个Sprint必须按时结束。

alan

老师好！关于人工测试，我想向您请教一个困惑。
我所在的团队正在尝试敏捷开发，遇到的问题是“人工测试”不知该摆在什么位置。
我们当前在Jira上设置的工作流是这样：todo→进行中→已完成→测试中→已测试。这种工作流是可行的吗？还是说把当前sprint的测试工作，都放到下个sprint会比较好？
由于我们有专职测试的同事，但是很多issue是测试同事很难进行测试的，导致工作流走不顺畅。
但如果不设置“测试中”这列，又觉得质量没有足够的保障（我们的自动化测试还很不完善）。
谢谢老师！不知您后续是否会就敏捷开发中测试的话题单独讲述。

2019-03-09

> 作者回复: 我觉得当前Sprint的测试，还是应该和当前Sprint一起走比较好，因为这个Sprint的内容是不是上线，还是要看测试是不是已经通过。另外两个Sprint或多个Sprint是可以同时存在的。
>
> 针对文章中的流程，以下工作流可以参考：
> ToDo->开发中->代码审查中->合并master->部署测试环境->测试通过->已部署
>
> 设置“测试中”取决于测试人员是不是很多，任务可能有冲突，要不要知道测试人员当前正在干嘛，不需要的话其实不需要，只需要知道测试结果就好了：测试通过移动到“测试通过”栏，测试没通过，回到“To Do”栏。
>
> 有些测试只有开发能测试的，这种应该在Ticket中标明，例如在标题上加上“[开发]”标签，然后由组内其他开发人员辅助测试，或者根本不需要测试。
>
> 自动化测试要放作为日常开发任务的一部分，比如一个任务，你可以创建两条Ticket，一条是开发的ticket，一条是自动化测试代码的Ticket，分别进行打分。
>
> 后面还会有测试章节。

再见陛下

赶紧查一下啥叫scrum啥叫sprint😂

2019-03-09

> 作者回复: 抱歉，敏捷概念太多，没展开细讲

阿杜

优点：新开发的功能有足够的时间测试
缺点：合并分支有点麻烦，开发和改bug同时进行，如果前一个sprint开发的代码bug比较多，可能会影响这个sprint的开发

关于分支部署那里，我们采用的办法是拉个新分支做开发，在预发测试好了再合并但master部署。

另外阅读本文的收获有两个：扑克牌估算工作量，这个确实之前是非常头疼的环节，准备尝试一下新方法；不同的会议的作用和介绍，有些可以借鉴一下用

2019-03-09

> 作者回复: 分析的很到位
>
> 分支不一定要合并，也可以考虑一个Commit放到两个Branch，git用cherry-pick可以支持
>
> 新分支开发也是个不错的办法，有一个小问题就是如果线上版本要打补丁，而这时候开发版本和master合并了，就稍微麻烦一点。

天之大舒

模式差不多。但我们测试和开发同步，我们有自动测试和专门的测试人员，测试人员测试前一天开发提交的代码，开发人员优先解决测试发现的问题（会导致开发加班）。如果不同步，会影响版本发布

2019-03-09

> 作者回复: 其实早期我试过在一个Sprint里面开发和测试，后来发现测试时间不充分，上线后老有小问题，所以调整为一周开发，一周测试，错开的这种方式，就特别稳定了，每周也有发布。

天之大舒

看了上下两篇文章，自己当前紧急的问题是成员的责任心和能力问题，就是怎样培养团队成员？老师有没有好办法

2019-03-09

> 作者回复: 这个呀，有一些建议参考：
> 1. 招人和开人都很重要，招优秀的，开掉没有责任心，没能力的。这两点都不容易做到，不过得坚持做；
> 2. 设置合理的流程，配合一定的奖惩制度；你奖励什么，团队就会往哪方面发展；
> 3. 团队要有梯队，不能都是资历浅的也不能都是资深的，保持一个合适的比例是比较健康的；
> 4. 实战中锻炼，实战中磨合；给他们有挑战的任务，给予合适的指导（这就是有梯队的原因，需要高一级别的待低一级别的）
>
> 仅供参考

Lrwin

1.如果项目做完后，运维问题有没有什么好的方案?
2.使用敏捷开发后，因为sprint大多都是对于功能而言的，我发现技术债会变多，如何解决?
3.同时多个功能需求来了，只开启一个分支，不对不同功能开启分支，因为敏捷开发是用于快速响应业务的，如果在一个分支的情况下，这个期间某个功能不要了，怎么办?
4.每天的ticket 需要监控吧？否则sprint有可能delay的

2019-03-09

> 作者回复: 1. 运维的话得看你是基于什么样的服务器。运维要注意两个问题：一个是要实现自动化；二要实现监控和报警；

> 2. 要定期重构，并且把重构作为Sprint的任务之一；
> 3. 那你不如还是开分支呀，我在上一篇文章中介绍了基于git的开发流程，GitHub-Flow可以很好解决你这种问题。Git开个分支没啥成本的；
> 4. 看板就是干这事的，每天看一下看板的各个栏下面有多少Ticket，就知道进展如何了。

池强泽

大公司分小团队，跨团队合作，依赖必然很多。感觉会延误迭代周。

2019-03-09

> 作者回复: 迭代的话，通常都会按时发布。
>
> 偶尔有依赖，例如某个功能依赖另一个组的实现，而他们没能按时完成，要么这个功能放到下个Sprint继续，要么这个Sprint就不发布了，合并到下一个Sprint一起发布。

williamcai

优点是开发测试并行进行，提高工作吞吐量，缺点是上一次的问题要拖到本次解决，增加代码合并的成本

2019-03-09

> 作者回复: 👍
> 是的，不仅提高吞吐量，最重要是有充足的测试时间。
>
> 两个分支同时工作，合并是个麻烦事😥

西西弗与卡夫卡

开发和测试分成两个Sprint，好处是开发和测试可以相对独立计划开展工作（因为计划开始时间都是周一）。坏处是，某种意义上破坏了每个Sprint要交付完整价值的初衷。不过，逻辑上可以将两次Sprint看成一次完整交付来衡量结果

2019-03-09

> 作者回复: 👍
>
> 是的，好处是可以有更多时间测试。其实从整体来看，还是每周都有一个新版本部署都有产出，所以还是可以算每周一个Sprint。
>
> 这只是其中一种每周一Sprint方案，并不是唯一方案。