不做代码审查又怎样？ - 文章 - 伯乐在线

从一次回顾会议开始

“要不……我们不做……代码审查了……试试？”还记得当有人抛出这个建议时周围同学的表情，那种表情用两个字加两个标点符号就可以形容：“什么？！”

对了，先介绍一下背景，这是项目一次普通的回顾会议，我们正在讨论的是如何让代码审查更有效率和效果。我们做代码审查的方式比较简单直接，就是每日站会后，大家围在一台开发机周围，逐一轮换讲解昨天所有提交的内容，就像下图中的那样。还有，这是一个已经超过了7年的比较大型的项目，代码审查是我们从项目开始就坚持的一个实践，所以当有人提议废除它的时候，这在很多同学心里是想都没想过的事情。

代码审查是一个很好的实践，可以帮助团队里的同学了解其他同学在做什么，可以分享项目的上下文，可以分享技术上的一些小魔法，可以发现很多潜在的代码缺陷，可以提高代码质量，还可以有很多很多好处……

但是，在真正的实施过程中，很多情况下并不像想象的那般美好，经常出现例如有些同学由于跟不上其他人讲解的速度（毕竟不是自己写的）或是没有相关的上下文（例如刚加入项目的新成员），或是由于提交没有被很好的切分和组织，导致整个过程都处于游离状态（就像下图中的我……毫无摆拍痕迹），而代码审查的效果也打了折扣，渐渐的变成了一个流程，一个过场， 一个习惯。

于是团队里就有人站了出来，引导大家去发现背后的问题，也就引来了这样一场激烈的讨论。在讨论中，有些同学坚持在说代码审查还是很有用的，有这样那样的好处，需要保持下去；有些同学则非常实际地指出了执行上的各种困难和问题。讨论异常激烈，直到有人小心翼翼地提出了文章开头的那个建议，一片哗然后大家都陷入沉寂：是啊，不做代码审查了，我们会失去或是得到什么呢？

沟通的收益和成本

在我看来，代码审查所暴露出的问题本质上就是如何权衡沟通的成本和收益的问题。

在软件开发过程的发展中，无论是从一开始的瀑布，后来的敏捷，还是当下的精益。都有很大一部分篇幅是在强调沟通或者说是强调沟通方式的演进，都是为了解决已有沟通方式所带来的各种限制和问题。

例如在我们自己的项目中就采用以下的实践来促进团队内外的沟通，包括：迭代计划会（Iteration Planning Meeting），每日站会（Daily Standup），代码评审会（Code Review），回顾会议（Retrospective Meeting）；还有XP中的结对编程(Pair programming)，现场客户 （ On-site Customer ）等。

而沟通的好处是可以得到快速的反馈，无论是80年之前就出现的PDCA环还是前两年大热的精益创业，都

是在强调快速反馈和基于反馈的快速迭代，通过这种方式来消除生产和创业过程中造成的各种浪费。

但是（对，又是但是……），沟通也是有成本的，而且成本一般都还不低，相信这点也不需要多解释，大家肯定都经历过各种各样效率极低令人抓狂的讨论或是会议。

在沟通的成本和收益同时摆在我们面前，如何做出选择？如何才能设计一套刚刚好的沟通体系，平衡收益与成本，来满足团队和项目的需要呢？

沟通金字塔

在读《重构》的时候，我深深地体会到：这个世界上没有什么是不变的，包括变化本身。所以要想得到解脱，我们就需要从简单的对与错，好与坏的漩涡中跳脱出来，用变化的眼光来看待周围的事物，并做好一直变化的准备。

而面对沟通成本与收益的选择困境时，我第一个想到的就是测试金字塔。了解测试金字塔的同学肯定都知道，测试金字塔是一个很好的工具，它帮助我们从单一的测试选择困境中跳脱出来，将各种不同类型的测试建立起关联、纳入一个统一的体系，从而让我们可以在一个更高的维度来系统的思考和审视每一种测试的策略，关注点也从简单的“该不该”变为了“如何变化”。

至于“如何变化”？我们可以通过在金字塔上添加一层新的测试种类来弥补整体测试策略中粒度太粗的问题；也可以根据项目情况通过移除一层测试种类，用其上层或是下层的测试来覆盖其测试用例，来减少成本；也可以通过将某个测试用例在金字塔中向上层或是向下层移动来寻找收益与成本的平衡。

举个实际点儿的例子，例如我可以将一些基于UI的集成测试用例下移，用成本更低的单元测试来覆盖从而减少成本，加快反馈速度；也可以将一个单元测试用例上移，用基于UI的测试来增加其稳定性的和体现的业务价值。

看，我们讨论的内容从简单的要不要写UI测试，需要写多少单元测试测试已经被转换到对于整体策略的变化和调整上来了。那对于代码审查的问题能否也通过金字塔这个工具转换到更大的空间上寻求突破呢？这就是沟通金字塔。

相比于测试金字塔中的“UI——Service——UT，“Iteration Planning Meeting（见上图）—— Code Review（见上上上图）—— Pair programming（见下图）”就可以类比成沟通金字塔，和测试金字塔一样更靠近金字塔顶端的（例如迭代计划会）沟通频率越低，成本越高，但越接近业务；越靠近金字塔底端的（例如结对编程）沟通频率越高，成本越低，越接近实现。这几种不同的沟通方式所沟通的内容肯定也会有所重叠，通过将各个层次的沟通方式进行组合来保证我们团队的整体沟通质量，就像通过金字塔中的各种测试层次的组合来保证产品质量的一样。

如果可以这么类比的话，那我们对于沟通质量的管理也应该是动态的、系统的、从整体上出发的。例如就可以通过在沟通金字塔中添加一层新的沟通机制来弥补沟通粒度过粗的问题，例如QA Team现在在做的每周例会就是一个好的例子；也可以将一些沟通内容通过层次（上下）的调整，甚至是通过直接减少一层沟通方式来优化我们的整体沟通效率，例如我们可以通过增加结对编程的Switch频度来替换掉成本更高的代码审查。而目标就是通过不断地动态调整和优化沟通结构，试图寻求一个沟通成本，沟通收益，沟通效率平衡的沟通环境。

回到问题上来

如果沟通金字塔的理论说的通，那代码评审就不再是一个：“必须要做的敏捷实践”，而只是沟通金字塔上的一层而已。那它的存在必然是为了弥补上下层沟通之间的空隙，那这个空隙到底是什么呢？是什么样的沟通是结对编程所不能覆盖，而用类似于迭代计划会这种更高层的沟通机制覆盖又不太经济的呢？为了让团队重新找回这个答案，我们最终决定试一试：停止代码审查一个月，在这一个月的时间我们去体会没有代码审查的得与失，在一个月之后重新举行回顾会议再来讨论是否要继续做代码审查。

在一个月后如期进行的回顾会议上，团队又重新讨论了这个议题，最终觉得通过这一个月的尝试，在还无法做到更频繁地Switch Pair的情况下，代码审查还是很有必要的。例如在这个月中，大家对于其他人在做的工作了解变少，集成出现了很多冲突；缺陷的数量也有所增加，其中有些是很明显的错误，很容易通过代码审查的方式发现并在前期消除；代码质量也有明显下降，出现了测试的缺失和很多代码坏味道。

而另一方面为了让代码审查能够真正的发挥其作用和价值，经过讨论我们也优化了代码审查的方式，让大家更有参与感，更有效率，也更有乐趣（见下图抓拍）。

交付价值 Over 遵循实践

日本剑道有个心诀，叫守 破 离:
1.“守”：最初阶段须遵从老师教诲，认真练习基础，达到熟练的境界。
2.“破”：基础熟练后，试着突破原有规范让自己得到更高层次的进化。
3.“离”：在更高层次得到新的认识并总结，自创新招数另辟出新境界。

守固然重要，但如果不能在守得基础上寻求突破，领会其中的奥秘和背后的道理，则始终无法达到离的新境界。在中国的武术中也有“无招胜有招”的说法，这里的无招就是指在将招数融会贯通之后，能够运用招式背后的原理，打破招数的限制，随机应变，自由应对。

而反观我们自己，是不是已经慢慢的不知不觉的被困在“守”的围城之内，变成了猴子定律中最后的那群猴子，只知道去拿香蕉会被打，也会跟着其他猴子去打那些试图拿香蕉的新猴子，但是为什么要这么做？我们已经忘了，或从来都没有知道过。

所以，不要以为遵循了敏捷提倡的一些实践我们就是敏捷的，不要以为遵循了精益的实践我们就是精益的。在我们没有理解并追求其背后真正价值的时候，只不过是平添了另外一份成本而已，不如不做。

加入伯乐在线专栏作者。扩大知名度，还能得赞赏！详见《招募专栏作者》
1 赞 1 收藏 评论

关于作者：ThoughtWorks

ThoughtWorks是一家全球IT咨询公司，追求卓越软件质量，为客户提供注重实效的咨询服务（IT组织优化、技术咨询、测试策略、客户体验）。同时我们也擅长构建定制化系统和产品，帮助客户快速将概念转化为价值。 我们满怀信心和激情，不断推动敏捷在中国IT业的推广和应用，提升中国软件开发的生产力。

合作联系

Email：bd@Jobbole.com

QQ： 2302462408 （加好友请注明来意）

更多频道

小组 - 好的话题、有启发的回复、值得信赖的圈子
头条 - 分享和发现有价值的内容与观点
相亲 - 为IT单身男女服务的征婚传播平台
资源 - 优秀的工具资源导航
翻译 - 翻译传播优秀的外文文章
文章 - 国内外的精选文章
设计 - UI,网页，交互和用户体验
iOS - 专注iOS技术分享
安卓 - 专注Android技术分享
前端 - JavaScript, HTML5, CSS
Java - 专注Java技术分享
Python - 专注Python技术分享