加餐三-聊一聊Google是如何做CodeReview的

100篇的正文已经全部结束了，估计你学得也有点累了吧？时隔这么久，正文终于结束了，从今天起，我们继续加餐内容。

跟正文内容相比，加餐内容我希望尽量轻松有趣，帮你拓展知识面，主要是课后的一些小分享，有的会以讲故事为主，但我也希望它能给你带来收获。如果能够引发你的思考和共鸣那就更好了。所以，我也希望你在留言区，多说说自己的感受和看法，多多与我互动。

话不多说，让我们正式开始今天加餐的内容吧！

为什么国内企业不重视Code Review？

在专栏第80讲中，我列举了Code Review的重要性，在项目中执行Code Review会带来哪些好处，以及如何克服一些常见的难题，在项目中启动Code Review等等。今天，我们想再继续这个话题，和你聊一下Code Review。不过，我刚才也说了，今天的内容会相对轻松一些，我会主要给你讲讲我在Google做Code Review的一些经验和心得。

我们都知道，Google在Code Review方面做得非常好，可以说是很多公司学习的榜样。从我个人的经历来说，我的技术成长相当大的一部分得益于当年在Google的Code Review。所以，我也希望更多的同行能意识到Code Review的重要性，能够在项目中推行Code Review，受益于Code Review。

但据我了解，国内的大部分公司都不怎么接受Code Review，在开发中，根本没有Code Review的流程。所以，我一直思考，到底是什么原因，导致这么优秀的一种开发模式，在国内的技术圈内没有被发扬光大。很多人会认为，主要原因是，项目工期紧，没时间做Code Review。我觉得这只是表面的原因，最根本的原因还是缺少技术文化传承。

我们知道，普遍而言，越是大公司里的工程师，技术能力会越强，技术影响力会越大。这些公司的工程师，即便跳槽去其他公司，一般都会担任核心成员或者Leader的角色。但是，在国内，即便像BAT这些输出有影响力工程师最多的一线公司，也没有很好地实践Code Review，相对应的，这些公司的工程师也就没有一手的Code Review的经验和感受，更无法了解到Code Review的好处，也更不会在团队、公司，甚至技术圈中去推行Code Review了。

打个不恰当的比方，这些一线互联网公司的工程师一直接受着“996”狼性文化价值观的熏陶，即便跳槽去其他公司，作为资深员工或者技术Leader，他们也会带领新的团队开始996，最终导致整个IT行业的加班氛围都很浓，不加班反倒会显得不正常。

用996作类比，如果BAT这些比较有技术影响力的公司，内部对Code Review很认可，执行得非常好，从这些公司往外输出的工程师，就会像我一样，大力传播Code Review。星星之火可以燎原，慢慢地，整个技术圈就会接受并且推行Code Review了。

实际上，据我所知，不只是我，只要是从Google跳槽出来的工程师，到了其他公司之后，都特别热衷于传播Code Review。而且，只要是被Google工程师带领过的团队，在开发流程中严格执行过Code Review的团队，对Code Review都无比认可。所以，我个人觉得，很多人不认可、不推行Code Review，最直接的原因还是没有经历过Code Review，没有有经验的人来带。

实际上，才开始接触Code Review的时候，我也比较反感。我刚毕业就进入了Google，在此之前，上学的

时候，尽管也写了很多代码，也参与过一些垂直课题的研发，但是，那时候的开发只是为了完成功能，从来没有考虑过代码质量问题、代码设计问题，更别提Code Review了。现在想想，自己当时对Code Review的认知水平，跟现在很多国内工程师的认知其实是差不多的。

所以，在一开始进入Google的时候，对于Code Review我也是不怎么接受的。我第一次提交的代码不足百行，就被Leader Review出了n多问题，而且大部分问题都非常细节，比如变量的命名不够达意、注释不够规范、多了一个空行、少了一个空格之类的。对于这些琐碎的细节，我当时心里挺排斥的，心想：我是来“造火箭”的，为什么成天纠结于这些“拧螺丝”的事情呢？

现在回去想想，当时的想法真的挺幼稚的。

如果站在团队协作的角度来看，对于一个长期维护、多人参与、代码比较多的项目来说，代码的可读性、可维护性等与质量相关的问题，是非常重要的。所以，Code Review作为保证代码质量的最有效手段之一，也就非常有必要了。如此吹毛求疵地执行Code Review，看似非常极端，但也表明了公司强硬的态度、坚定的立场，就是要把Code Review执行彻底。这也是Code Review没有在Google流于形式的一个很大的原因。

在入职一段时间后，来来回回经过多次Code Review之后，我的代码质量整体提高了很多，被Review出的问题也越来越少了，我也切身地体会到Code Review的好处。因此，慢慢地，对这件事，我从排斥变得认可。与此同时，我也慢慢地开始Review别人的代码了。

Google是如何进行Code Review的？

在Google，我们把每次提交的代码片段叫做一个CL，全称是Change List。它就相当于GitHub中的PR（Pull Request）。每个CL都要至少一个Owner和一个具有Readability的同事Approve，才能提交到代码仓库中。其中，Owner一般都是技术Leader或者项目负责人，而Readability是一个证书，表示你具有了写出可读代码、符合编码规范代码的能力。Readability会细化到每种编程语言，比如Java Readability、C++ Readability等。

如果你想申请某种语言的Readability，你就要提交一段至少包含100行代码、并且稍微有点复杂的CL给Readablity评审委员会。评审委员会会指派一个资深工程师Review你的代码，给你一些修改建议，然后，你需要根据修改建议对代码进行修改，再提交Review。这样来来回回几次之后，他觉得没问题了，就会给你颁发Readability。有了Readability之后，你的Review才真的能起到Approve的作用。当然，即便没有Readability，你对同事代码的Review本身也是有价值的。所以，并非只有Readability的人才能Review别人的代码。

在Google，每种编程语言都有对应的编码规范。但是，Code Review本身并没有统一的Check list。在Code Review的时候，除了编码规范可以参考之外，大部分都是靠工程师自身的经验来Review。不过，Review考虑的也无外乎这样几个常见的方面：代码结构是否合理、代码是否容易理解、业务是否正确、异常考虑是否全面、是否有隐藏的bug、线程是否安全、性能是否满足业务需求、是否符合编码规范等等。

Code Review听起来很复杂，要考虑的点很多，但实际上，等到你做熟练了之后，并不会花费太长的时间。一个CL从提交Review到最终合并到代码仓库，一般也就需要一天的时间。当然，对于一些比较大的CL、比较复杂的CL、有比较多争议的CL，以及一些新手的CL，可能会花费比较多的时间。

但是，大部分情况下，我们都不提倡太大的CL。太大的CL对代码审查者来说是很大的负担，Review过程会很慢，会导致代码迟迟提交不上去。

对于比较复杂的CL，我们一般建议要写好文档，或者通过类似Jira这样的项目工具，详细描述CL的前因后果、上下文背景。这样，代码审查者就能一眼看懂代码包含的设计意图。对于争议比较多的CL，我们建议直接当面沟通，这样也更加有效率。对于一些新手的CL，因为他们对编码规范等不熟练，可能来来回回要改好几次，才能满足要求，但这个过程是每个新人都要经历的，多改几次就好了。

实际上，Code Review并不神秘，如果你想了解更多关于Code Review的事情，可以去读一读Google官方公布的Code Review最佳实践。当然，如果有什么疑问，你也可以在留言区问我。

让国内大部分IT从业人士认识到Code Review的重要性，形成Code Review的技术文化，可能还需要一个漫长的时间。不过，我特别希望，你在学完专栏之后，能够意识到Code Review的重要性。有朝一日，当你成了领导，有了话语权、影响力之后，能够推动在团队、公司内进行Code Review，甚至为Code Review在整个国内技术圈中发扬光大贡献一份力量。

课堂讨论

你觉得为什么国内的大部分公司都不重视Code Review，在开发中都没有Code Review流程呢？你觉得如何把Code Review在国内技术圈中发扬光大呢？有什么好的建议吗？

欢迎留言和我分享你的想法，如果有收获，也欢迎你把这篇文章分享给你的朋友。

精选留言：

- wkq2786130 2020-06-24 00:28:28
  我刚开始也不理解code review，直到有一天发现自己写的代码自己读不懂，然后开始优化，开始写注释，理清主要逻辑，开始分层，开始使用通俗易懂的命名，后来逐渐意识到code review的好处 [6赞]

- tingye 2020-06-24 07:14:09
  国内code review难推广的一个原因可能也和文化有关，老外习惯直来直往评价和就事论事，中国人为人处世讲究委婉，要面子，特别同级别同事间往往不好意思直接指出别人的问题 [4赞]

- Jackey 2020-06-24 01:25:25
  能做的就是在自己的团队中大力推广code review，不符合规范的代码一定不能进入repo [2赞]

- Jxin 2020-06-24 01:13:48
  原因：
  1.缺少追求卓越的氛围。先run的理念退化成了能run就行。
  2.招聘要求上基本都会有代码设计能力，编码规范，甚至代码洁癖的项。但实际上基本不提，顶多背几个设计模式。那么编码能力就变得很鸡肋，因为它与薪资几乎无关。叫好不叫坐大概就是这个意思。
  3.重构本是小步快跑，但我看到的大部分都是重写，而非重构。这就导致认知中的重构成本很高，进而就会排斥。而只写代码不重构代码，在编码能力的提升上是很缓慢的。如果把识别坏代码的能力看作是一把尺子。经常重构的人，这把尺子的精度是一毫米，只写功能的人精度只有一分米。那么在识别坏味道评估改动点时就会很模糊，模糊就更不敢下手，恶性循环。

  办法：
  1.氛围，国内的开源项目先开始讲究，带个氛围。
  2.将编码能力和算法放在同等位置看待。其实编码能力强的人，往往意味着思路清晰，讲究。这种人工作能力一般差不了。
  3.普及重构理应小步快跑的理念。把事情拆小，把小事情做好，都很重要。重构需要会拆解工作，然后也别看重构手法简单，刻意训练后也会有质变。（重构是提高普遍认知的有效手段，只有认知上去了coder

eview才能被 重视） [2赞]

- progyoung 2020-06-24 09:57:03
  就个人的工作经历来看，很多互联网公司程序员的职责就是完成业务，没有功能bug是第一位的需求，至于代码质量，在leader的眼中根本就不重要。而且一个需求提出之后，恨不得马上就能看到结果，发布上线。在这样的大环境之下，996盛行，code review就被认为是浪费时间，怎么能推行的开呢？ [1赞]

- 虢國技醬 2020-06-24 00:58:45
  我觉得code review好处之一就是帮助部分同学提高编码技能；毕竟工作不像在学校，写的不好的同学老师会手把手教你，code review让大家看到优秀的，也看到槽糕的 [1赞]

- liu_liu 2020-06-24 09:37:12
  1. 没有切身体验到 code review 的好处，大多是流于形式的过一遍，或者根本没有。
  2. 觉得是种负担，浪费时间。没有把它作为一种可以提高代码质量，发现隐藏问题的方式。
  3. 目前我们的 code review 也只是检查些较明显出错的地方，并不细致。
  4. 需要对 code review 有丰富经验的人从小范围开始推广，并见到实效，进而逐步铺开。

- 迷羊 2020-06-24 09:17:54
  感觉还是没有会 Code Review 的人带