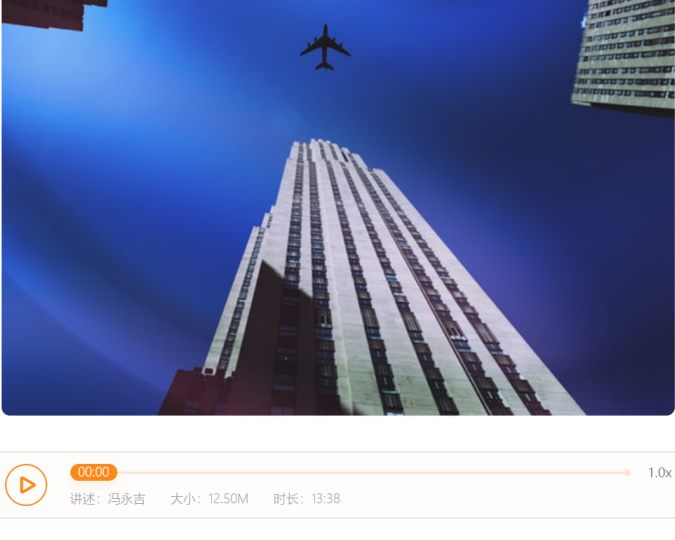


80 | 开源实战二（下）：从Unix开源开发学习应对大型复杂项目开发

王争 2020-05-06



00:00

讲述：冯永吉 大小：12.50M 时长：13:38

1.0x <

上两节课，我们分别从代码编写、研发管理的角度，学习了如何应对大型复杂软件开发。在研发管理这一部分，我们又讲到比较重要的几点，它们分别是编码规范、单元测试、持续重构和 Code Review。其中，前三点在专栏的理论部分都有比较详细的讲解，而唯独 Code Review 我们还没有讲过，所以，今天我就借机会和你补充一下这一部分的内容。

很多年前，我跟一个有十几年研发经验的某一线大厂的技术专家聊天，聊天中我提起了 Code Review，他便对 Code Review 一顿否定。他说，Code Review 比较浪费时间，往往会虎头蛇尾，不可能在企业中很好地落地执行。当我又提起，Code Review 在 Google 执行得很好，并且是已经习以为常的开发流程的时候，他竟然说这绝对不可能。

一个技术不错，可以玩转各种架构、框架、中间件的资深 IT 从业者，居然对 Code Review 有如此的偏见，这到底是哪里出了问题呢？我觉得问题主要还是出自认知上。

所以，今天，我并不打算讲关于如何做 Code Review 的方法论，我更希望充当一个 Code Review 布道师的角色，讲一讲为什么要进行 Code Review，Code Review 的价值在哪里，让你重视、认可 Code Review。因为我觉得，只要从认知上接受了 Code Review，对于高智商的 IT 人群来说，搞清楚如何做 Code Review 并不是件难事。而且，Google 也开源了它自己的 Code Review 最佳实践，网上很容易搜到，你完全可以对照着来做。

话不多说，让我们正式开始今天的内容吧！

我为什么如此强调 Code Review？

Code Review 中文叫代码审查。据我了解，在国内绝大部分的互联网企业里面，很少有将 Code Review 执行得很好的，这其中包括 BAT 这些大厂。特别是在一些需求变动大、项目工期紧的业务开发部门，就更不可能有 Code Review 流程了。代码写完之后就随手就提交上去，然后丢给测试狠命测，发现 bug 再修改。

相反，一些外企非常重视 Code Review，特别是 FLAG 这些大厂，Code Review 落地执行得非常好。在 Google 工作的几年里，我也切实体会到了 Code Review 的好处。这里我就结合我自身的真实感受讲一讲 Code Review 的价值，试着“说服”一下你。

1.Code Review 践行“三人行必有我师”

有时候你可能会觉得，团队中的资深员工或者技术 leader 的技术比较牛，写的代码很好，他们的代码就不需要 Review 了，我们重点 Review 资历浅的员工的代码就可以了。实际上，这种认识是不对的。

我们都知道，Google 工程师的平均研发水平都很高，但即便如此，我们发现，不管谁提交的代码，包括 Jeff Dean 的，只要需要 Review，都会收到很多 comments（修改意见）。中国有句老话，“三人行必有我师”，我觉得用在这里非常合适。即便自己觉得写得已经很好的代码，只要经过不停地推敲，都有持续改进的空间。

所以，永远不要觉得自己很厉害，写的代码就不需要别人 Review 了；永远不要觉得自己水平很一般，就没有资格给别人 Review 了；更不要觉得技术大牛让你 Review 代码只是缺少你的一个“approve”，随便看看就可以。

2.Code Review 能摒弃“个人英雄主义”

在一个成熟的公司里，所有的架构设计、实现，都应该是一个团队的产出。尽管这个过程可能会由某个人来主导，但应该是整个团队共同智慧的结晶。

如果一个人默默地写代码提交，不经过团队的 Review，这样的代码蕴含的是一个人的智慧。代码的质量完全依赖于这个人的技术水平。这就会导致代码质量参差不齐。如果经过团队多人 Review、打磨，代码蕴含的是整个团队的智慧，可以保证代码按照团队中的最高水准输出。

3.Code Review 能有效提高代码可读性

前面我们反复强调，在大部分情况下，代码的可读性比任何其他方面（比如扩展性等）都重要。可读性好，代表后期维护成本低，线上 bug 容易排查，新人容易熟悉代码，老人离职时代码容易接手。而且，可读性好，也说明代码足够简单，出错可能性小、bug 少。

不过，自己看自己写的代码，总是会觉得很易读，但换另外一个人来读你的代码，他可能就不这么认为了。毕竟自己写的代码，其中涉及的业务、技术自己很熟悉，别人不一定会熟悉。既然自己对可读性的判断很容易出现错觉，那 Code Review 就是一种考察代码可读性的很好手段。如果代码审查者很费劲才能看懂你写的代码，那就说明代码的可读性有待提高了。

还有，不知道你有没有这样的感受，写代码的时候，时间一长，改动的文件一多，就感觉晕乎乎的，脑子不清醒，逻辑不清晰？有句话讲，“旁观者清，当局者迷”，说的就是这个意思。Code Review 能有效地解决“当局者迷”的问题。在正式开始 Code Review 之前，我们将代码提交到 Review Board（Code Review 的工具界面）之后，所有的代码改动都放到了这一块，看起来一目了然、清晰可见。这个时候，还没有等其他同事 Review，我们自己就能发现很多问题。

4.Code Review 是技术传帮带的有效途径

良好的团队需要技术和业务的“传帮带”，那如何来做“传帮带”呢？当然，业务上面，我们可能通过文档或口头相传的方式，那技术呢？如何培养初级工程师的技术能力呢？Code Review 就是一种很好的途径。每次 Code Review 都是一次真实案例的讲解。通过 Code Review，在实践中将技术传递给初级工程师，比让他们自己学习、自己摸索来得更高效！

5.Code Review 保证代码不止一个人熟悉

如果一段代码只有一个人熟悉，如果这个同事休假了或离职了，代码交接起来就比较费劲。有时候，我们单纯只看代码还看不大懂，又要跟 PM、业务团队、或者其他技术团队，再重复来一轮沟通，搞的其他团队的人都很烦。而 Code Review 能保证任何代码同时至少有两个同事熟悉，互为备份，有备无患，除非两个同事同时都离职……

6.Code Review 能打造良好的技术氛围

提交代码 Review 的人，希望自己写的代码足够优秀，毕竟被同事 Review 出很多问题，是件很丢人的事情。而做 Code review 的人，也希望自己尽可能地提出有建设性意见，展示自己的能能力。所以，Code Review 还能增进技术交流，活跃技术氛围，培养大家的极客精神，以及对代码质量的追求。

一个好的技术氛围，能让团队有很强的自驱力。不用技术 leader 反复强调代码质量有多重要，团队中的成员就会自己主动去关注代码质量的问题。这比制定各种规章制度、天天督促执行要更加有效。实际上，我多说一句，好的技术氛围也能降低团队的离职率。

7.Code Review 是一种技术沟通方式

Talk is cheap, show me the code. 怎么“show”，通过 Code Review 工具来“show”，这样也方便别人反馈意见。特别是对于跨不同办公室、跨时区的沟通，Code Review 是一种很好的沟通方式。我今天白天写的代码，明天来上班的时候，跨时区的同事已经帮我 Review 好了，我就可以改提交，继续写新的代码了。这样的协作效率会很高。

8.Code Review 能提高团队的自律性

在开发过程中，难免会有人不自律，存在侥幸心理：反正我写的代码也没人看，随便写写就提交了。Code Review 相当于一次代码直播，曝光 dirty code，有一定的威慑力。这样大家就不敢随便应付一下就是交代码了。

如何在团队中落地执行 Code Review？

刚刚讲了这么多 Code Review 的好处，我觉得大部分你应该都能认可，但我猜你可能会说，Google 之所以能很好地执行 Code Review，一方面是因为有经验的传承，起步阶段已经过去了；另一方面是本身员工技术素质、水平就很高，那在一个技术水平没那么强的团队，在起步阶段或项目工期很紧的情况下，如何落地执行 Code Review 呢？

接下来，我就很多人关于 Code Review 的一些疑惑，谈谈我自己的看法。

有人认为，Code Review 流程太长，太浪费时间，特别是工期紧的时候，今天改的代码，明天就要上，如果要等同事 Review，同事有可能没时间，这样就来不及。这个时候该怎么办呢？

我所经历的项目还没有一个因为工期紧，导致没有时间 Code Review 的。工期都是人排的，稍微排松点就行了啊。我觉得关键还是在于整个公司对 Code Review 的接受程度。而且，Code Review 熟练之后，并不需要花费太长的时间。尽管开始做 Code Review 的时候，你可能因为不熟练，需要有一个 checklist 对照着来做。起步阶段可能会比较耗时。但当你熟练之后，Code Review 就像键盘盲打一样，你已经忘记了哪个手指按的是哪个键了，扫一遍代码就能揪出绝大部分问题。

有人认为，业务一直在变，今天写的代码明天可能就要再改，代码可能不会长期维护，写得再好也没用。这种情况下是不是就不需要 Code Review 了呢？

这种现象在游戏开发、一些早期的创业公司或者项目验证阶段比较常见。项目讲求短平快，先验证产品，再优化技术。如果确实面对的还只是生存问题，代码质量确实不是首要的，特殊情况下，不做 Code Review 是支持的！

有人说，团队成员技术水平不高，过往也没有 Code Review 的经验，不知道 Review 什么，也 Review 不出什么。自己代码都没写明白，不知道什么样的代码是好的，什么样的代码是差的，更不要说 Review 别人的代码了。在 Code Review 的时候，团队成员大眼瞪小眼，只能 Review 点语法，形式大于效果。这种情况该怎么办？

这种情况也挺常见。不过没关系，团队的技术水平都是可以培养的。我们可以先让资深同事、技术好的同事或技术 leader，来 Review 其他所有人的代码。Review 的过程本身就是一种“传帮带”的过程。慢慢地，整个团队就知道该如何 Review 了。虽然这可能会有一个相当长的过程，但如果真的想在团队中执行 Code Review，这不失为一种“曲线救国”的方法。

还有人说，刚开始 Code Review 的时候，大家都还挺认真，但时间长了，大家觉得这事跟 KPI 无关，而且我还要看别人的代码，理解别人写的代码的业务，多浪费时间啊。慢慢地，Code Review 就变得流于形式了。有人提交了代码，随便抓个人 Review，Review 的人也不认真，随便扫一眼就点“approve”。这种情况该如何应对？

我的对策是这样的。首先，要明确的告诉 Code Review 的重要性，要严格执行，让大家不要懈怠，适当的时候可以“杀鸡儆猴”。其次，可以像 Google 一样，将 Code Review 间接地跟 KPI、升职等联系在一块，高级工程师有义务做 Code Review，就像有义务做技术面试一样。再次，想办法活跃团队的技术氛围，把 Code Review 作为一种展示自己技术的机会，带动起大家对 Code Review 的积极性，提高大家对 Code Review 的认同感。

最后，我再多说几句。Google 的 Code Review 是做得很好的，可以说是谷歌保持代码高质量最有效的手段之一了。Google 的 Code Review 非常严格，多一个空行，多一个空格，注释有拼错的单词，变量命名得不够好，都会被指出来要求修改。之所以如此吹毛求疵，并非矫枉过正，而是要给大家传递一个信息：代码质量非常重要，一点都不能马虎。

重点回顾

好了，今天的内容到此就讲完了。我们一块来总结回顾一下，你需要重点掌握的内容。

今天，我们主要讲了为什么要做 Code Review，Code Review 的价值在哪里。我的总结如下：Code Review 践行“三人行必有我师”、能摒弃“个人英雄主义”、能有效提高代码可读性、是技术传帮带的有效途径、能保证代码不止一个人熟悉、能打造良好的技术氛围、是一种技术沟通方式、能提高团队的自律性。

除此之外，我还对 Code Review 在落地执行过程中的一些问题，做了简单的答疑。我这里就不再重复罗列了。如果你在 Code Review 过程中遇到同样的问题，希望我的建议对你有帮助。

课堂讨论

对是否应该做 Code Review，你有什么看法呢？你所在的公司是否有严格的 Code Review 呢？在 Code Review 的过程中，你又遇到了哪些问题？

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。

课程预告

5 月- 6 月课表抢先看

充 ¥500 得 ¥580

赠「¥ 99 运动水杯+ ¥129 防紫外线伞」

【点击】图片, 立即查看 >>>

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究法律责任。

志恒Z

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 (18)