**Code Review中的几个提示**



Code Review应该是软件工程最最有价值的一个活动，之前，本站发表过《[简单实用的Code Review工具](http://coolshell.cn/articles/1218.html)》，那些工具主要是用来帮助更有效地进行这个活动，这里的这篇文章，我们主要想和大家分享一下Code Review代码审查的一些心得。

首先，我们先来看看Code Reivew的用处：

1. Code reviews 中，可以通过大家的建议增进代码的质量。
2. Code reviews  是一个传递知识的手段，可以让其它并不熟悉代码的人知道作者的意图和想法，从而可以在以后轻松维护代码。
3. Code reviews 也鼓励程序员们相互学习对方的长处和优点。
4. Code reviews 也可以被用来确认自己的设计和实现是一个清楚和简单的。

你也许注意到了在上面的Code Reivew中的诸多用处中，我们没有提到可以帮助找到程序的bug和保证代码风格和编码标准。这是因为我们认为：

1. **Code reviews 不应该承担发现代码错误的职责**。Code Review主要是审核代码的质量，如可读性，可维护性，以及程序的逻辑和对需求和设计的实现。代码中的bug和错误应该由单元测试，功能测试，性能测试，回归测试来保证的（其中主要是单元测试，因为那是最接近Bug，也是Bug没有扩散的地方）
2. **Code reviews 不应该成为保证代码风格和编码标准的手段**。编码风格和代码规范都属于死的东西，每个程序员在把自己的代码提交团队Review的时候，代码就应该是符合规范的，这是默认值，属于每个人自己的事情，不应该交由团队来完成，否则只会浪费大家本来就不够的时间。我个人认为“meeting”是奢侈的，因为那需要大家在同一时刻都挤出时间，所以应该用在最需要的地方。代码规范比起程序的逻辑和对需求设计的实现来说，太不值得让大家都来了。

10年前，上面这两件事会是理所当然的（10年前的中国的软件开发还没有Code Reivew呢），今天，在中国的很多公司上面这两件事依然被认为是Code Reivew最重要的事，所以，我能够看到很多开发Team抱怨Code Review就是一个形式，费时费力不说，发现的问题还不如测试，而评审者们除了在代码风格上有些见术，别的也就没什么用了，长而久之，大家都会开始厌烦这个事了。

所以，在今天，请不要把上面的那两件事分散了Code Review的注意力，取而代之的是，对于Bug，程序的作者要在Review前提交自己的单元测试报告（如：XUnit的测试结果），对于代码规范，这是程序作者自己需要保证的，而且，有一些工具是可以帮你来检查代码规范的。

**当然，上述这些言论并不是说，你不能在Code Review中报告一个程序的bug或是一个代码规范的问题。我只是说，那并不是Code Review的意图。**

下面是我们认为的几个小提示可以让你更好进行Code Review这项最有价值的活动。

**1.- 经常进行Code Review**

以前经历过几个相当痛苦的Code Review，那几次Code Review都是在程序完成的时候进行的，当你面对那近万行的代码，以前N多掺和在一起的功能，你会发现，整个Code Review变得非常地艰难，用不了一会儿，你就会发现大家都在拼命地打着哈欠，但还是要坚持，有时候，这样的Review会持续3个小时以上，相当的夸张。而且，会议上会出现相当多的问题和争论，因为，这就好像，人家都把整个房子盖好了，大家Review时这挑一点那挑一点，有时候触动地基或是承重墙体，需要大动手术，让人返工，这当然会让盖房的人一下就跳起来极力地维护自己的代码，最后还伤了团队成员的感情。

所以，千万不要等大厦都盖好了再去Reivew，而且当有了地基，有了框架，有了房顶，有了门窗，有了装修，的各个时候循序渐进地进行Review，这样反而会更有效率，也更有帮助。

下面是一些观点，千万要铭记：

* **要Review的代码越多，那么要重构，重写的代码就会越多。而越不被程序作者接受的建议也会越多，唾沫口水战也会越多。**
* **程序员代码写得时候越长，程序员就会在代码中加入越来越多的个人的东西。** 程序员最大的问题就是“自负”，无论什么时候，什么情况下，有太多的机会会让这种“自负”澎涨开来，并开始影响团队影响整个项目，以至于听不见别人的建议，从而让Code Review变成了口水战。
* **越接近软件发布的最终期限，代码也就不能改得太多。**

我个人的习惯，并且也是对团队成员的要求是——先Review设计实现思路，然后Review设计模式，接着Review成形的骨干代码，最后Review完成的代码，如果程序复杂的话，需要拆成几个单元或模块分别Review。当然，最佳的practice是，每次Review的代码应该在1000行以内，时间不能超过一部电影的时间——1.5小时（因为据说那个一个正常人的膀胱可以容纳尿液的最长限度）

当然，在敏捷开发中，他们不需要Code Reivew，其实，敏捷开发中使用更为极端的“结对编程”（Pair-Programming）的方法 —— 一种时时刻刻都在进行Code Review的方法，个人感觉在实际过程中，这种方法有点过了。另外，大家可以看看本站的另一篇文章《[结对编程的利与弊](http://coolshell.cn/articles/16.html)》来了解一下这种方法的问题。

**2.- Code Review不要太正式，而且要短**

忘了那个代码评审的Checklist吧，走到你的同事座位跟前，像请师父一样请他坐到你的电脑面前，然后，花5分钟给他讲讲你的代码，给他另外一个5分钟让他给你的代码提提意见，这比什么都好。而如果你用了一个Checklist，让这个事情表现得很正式的话，下面两件事中必有一件事会发生：

1. 只有在Checklist上存在的东西才会被Review。
2. Code Reviews 变成了一种礼节性的东西，你的同事会装做很关心你的代码，但其实他心里想着尽快地离开你。

只有不正式的Code Review才会让你和评审者放轻松，人只有放松了，才会表现得很真实，很真诚。记住Review只不过是一种形式，而只有在相互信任中通过相互的讨论得到了有意义和有建设性的建议和意见，那才是最实在的。不然，作者和评审者的关系就会变成小偷和警察的关系。

**3.- 尽可能的让不同的人Reivew你的代码**

这是一个好主意，如果可能的话，不要总是只找一个人来Review你的代码，不同的人有不同的思考方式，有不同的见解，所以，不同的人可以全面的从各个方面评论你的代码，有的从实现的角度，有的从需求的角度，有的从用户使用的角度，有的从算法的角度，有的从性能效率的角度，有的从易读的角度，有的从扩展性的角度……，啊，好多啊，还让不让人活了。不管怎么说，多找一些不同的人会对你很有好处。当然，不要太多了，人多嘴杂反而适得其反，基本上来说，不要超过3个人，这是因为，这是一个可以围在一起讨论的最大人员尺寸。

下面是几个优点：

1. 从不同的方向评审代码总是好的。
2. 会有更多的人帮你在日后维护你的代码。
3. 这也是一个增加团队凝聚力的方法。

**4.- 保持积极的正面的态度**

再说一次，程序最大的问题就是“自负”，尤其当我们Reivew别人的代码的时候，我已经见过无数的场面，程序员在Code Review的时候，开始抨击别人的代码，质疑别人的能力。太可笑了，我分析了一下，这类的程序员其实并没有什么本事，因为他们指责对方的目的是想告诉大家自己有多么的牛，靠这种手段来表现自己的程序员，其实是就是传说中所说的“半瓶水”。

所以，无论是代码作者，还是评审者，都需要一种积极向上的正面的态度，作者需要能够虚心接受别人的建议，因为别人的建议是为了让你做得更好；评审者也需要以一种积极的正面的态度向作者提意见，因为那是和你在一个战壕里的战友。记住，你不是一段代码，你是一个人！

**5.- 学会享受Code Reivew**

这可能是最重要的一个提示了，如果你到了一个人人都喜欢Code Reivew的团阿，那么，你会进入到一个生机勃勃的地方，在那里，每个人都能写出质量非常好的代码，在那里，你不需要经理的管理，团队会自适应一切变化，他们相互学习，相互帮助，不仅仅是写出好的代码，而且团队和其中的每个人都会自动进化，最关键的是，这个是一个团队。