【双语】程序员——请学会交流

作者：Polly
原文链接：https://ld246.com/article/1517742835371
来源网站：链滴

英文原文链接：https://henrikwarne.com/2017/05/28/developers-talk-to-people

Many software developers have a tendency to avoid talking to people. They would rather just ely on written communication in chats, email or issue tracker tickets. However, talking to peop e more can make them more effective as software developers. Here are some examples:

程序员大多都有同样的毛病就是不愿意与人交流，他们宁愿用通讯工具比如email,提交工单也不远与直接交流，但是在工作中主动与人交流会很大程度的提高我们的工作效率，下面举几个栗子来给大家释一下：

Example 1. Suppose you are implementing a new feature. The ticket in the issue tracker descr bes how the new functionality should work. As you start working on it, you discover that impl menting it exactly the way it is described will be difficult and take quite a bit of time. However if the new feature it changed slightly, it fits right in with the existing code, so it would be poss ble to implement it much faster. At this point, many developers think to themselves: "they a ked for the more complicated version, so that' s what I' ll implement".

第一个栗子::假设你正在实现一个新的功能，需求文档里描述了该功能的内容，当你开始着手开发这功能的时候，你发现这个功能非常的复杂而且会耗费你很长的时间。但是如果稍作修改，就可以重用前的代码从而很快的完成这个功能，不过即使是如此，大多数程序员还是会认为： "他们可能是想让开发一个更复杂的版本"。

Instead, talk to the product manager (or whoever requested the feature) and explain the trad -off you discovered (slightly changed feature would be much faster to implement). Ask if they still want it as originally described. The point isn' t that you should persuade them that the f ature should be changed – the point is to let them know about the trade-off. Often they are ot aware of it. Whether they change their mind or not, the discussion leads to better understa ding for both of you.

相反，如果你去和你的项目经理（或者其他提出需求的人）讨论这个问题，让他们知道做这个简单的改所带来的好处（可以更快的完成这个功能），问是否还是采用原来的设计，注意这里的重点不是让们接受或者采纳你的想法，而是让他们知道这么做的好处，他们不会感到奇怪或者有什么不好的反应不管最终他们是否采纳你的意见，这对你们双方进一步理解这个功能有很大的帮助。

Example 2. Recently, a tester was verifying some small changes I had made. He wrote down i the ticket the cases that worked, and some cases that didn' t work. The reason some cases d dn' t work had to do with how I had deployed the new version of the software. I could have ritten down an explanation of that in the ticket. However, it takes time to explain something c early in writing. So instead I walked over to him and explained what had happened. This way, e had the opportunity to ask more questions, and I had the opportunity to add more clarificat on without it ping-ponging back and forth as comments in the ticket.

第二个栗子: 最近一个测试人员给我提交了一些小的需要修改的问题，他描述了哪些功能可以正常执，哪些功能不能。那些不能运行的功能是因为他不会去部署新的版本，我需要去告诉他怎么来部署，完全可以写一个说明文档给他，但是我没有这么做，我直接跑去他面前跟他解释该如何去做，通过这方式他有更多的机会问我更多的问题，我也不需要噼里啪啦的敲半天键盘来解释这件事了。

Even better, as we were talking about the ticket, I showed how I would check if the problem as due to the deployment or not. As I did that, he said "wait, what was that" – he had never used the command I used. So not only was the problem cleared up quickly, the tester learned a command he didn' t know about.

更赞的是，当我们在讨论这个问题时，我向他展示了我是怎么去检查并解决这个部署的问题的，当我示时，他很好奇这个操作而且他从来没有接触过，所以除了解决了当前的问题，他还学会了以后再遇

这个问题该怎么去解决。

Example 3. The other day, I was going to add a feature that required knowing if another feat re had been activated or not. The activation check had been a bit problematic before, so I wa ted to clean it up a little in the process. So I thought about where to put the changed activati n check. But before going ahead and implementing the solution, I mentioned to a colleague hat I intended to do.

第三个栗子: 又有一天，我准备开发一个监控器去监控一个功能是否运行正常，之前的监控器有点问，我准备在他的基础上去修改一下，替换一些新的代码进去，但在做这一切之前，我还是打算先去找事讨论一下我的想法。

He immediately said “why don’ t you put it there instead?”. Sure enough, his proposal was much better than mine – a better place to put it, and less impact overall on the code. So by ju t discussing for a few minutes, we found a better solution. I think of this as an informal design eview, similar to a code review, but done before implementation, not after. I frequently try to iscuss a change with a colleague before going ahead and doing it. Time and again I am grate ul that I did. Many times we discover flaws, or better solutions, that I would never have thoug t of, no matter how hard I tried. So with a little bit of discussing, I end up with much better so utions.

我的同事立刻就回答我：“你为什么要替换这段代码？”果然，他有比我更好的思路，影响更少的代。我跟他讨论了一会，得到了更好的解决方案。我想这就是_结对编程_（敏捷开发中的一种方式，我得这样翻译比较好接受，虽然可能不是很准确）的好处，就像一次代码回顾，但是是在代码实现之前行的，不是之后。从那之后我每次要实现什么功能都要跟同事去进行讨论，慢慢的发现了其中巨大的处，经常会发现一些缺陷，一些我根本想不到的更好的解决方案。不管我有多努力，一点点的讨论都给我的工作带来很大的好处。

CONCLUSION

in a lot of cases, talking to developers, testers, product managers and other stake-holders is b neficial to both parties. It is usually also faster than written communication. Despite this, I see many developers that are a bit reluctant. If you are one of those developers, make an effort to talk more to people for a few weeks, and see if it makes a difference.

结论:

通过这些例子，不管是和程序员，测试人员，项目经理还是其他有关的人员进行交流和讨论，对双方是有好处的，通常这种方式要比通讯软件效果要好的多。尽管如此还是有很多程序员不愿意去交流，果你是这样的人，可以尝试几周，你会看到它给你的工作带来的变化。(注意这是在你没有影响其他人作的前提之下哦，别人家正忙着呢去打扰别人，会引起反感)。