@程序员，敏捷开发防坑指南请查收！

原创：弯月 CSDN 3月24日

作者 | 弯月

责编 | 郭芮

随着市场的瞬息万变和软件行业的迅猛发展，传统的瀑布式软件开发模型因其漫长的开发与反馈周期，在抢占市场先机和快速满足用户需求方面日渐失去竞争优势。与此同时，敏捷开发以其快速迭代，持续满足不断变化的用户需求而受到越来越多人的重视。

虽说敏捷开发更加适合现代快速变化的软件开发行业，但是想要真正贯彻敏捷思想却非一件易事，在敏捷的实践过程中有很多常见的误区需要引起大家的注意。

01 CSDN

协作就意味着更多的会议

开会，开会，开会！程序员最烦的就是开会，而敏捷开发的会议实在太多了：每日的站立会议，每个迭代开始前的需求讨论会议，迭代开始时的计划会议，迭代结束时的验收会议，之后还有回顾会议，以及各种数不清的小范围会议。通常一个迭代为期两周，十个工作日，80个小时，这些会议就要占到20个小时以上，程序员哪还有时间写代码。更加讨厌的是，代码写到一半被叫去开会，一下午都没有效率可言了。

大家有这样的感受，说到底还是没有从思想上转变过来。敏捷思想是一种深刻的文化变革，真正的敏捷需要团队成员以及客户之间及时的沟通与紧密协作。一味低着头写代码，回头才发现做出来的功能不是客户想要的；又或者写了半天前端代码，才发现后台的API已经变了，原来的参数都不能用了；又或者调查了半天的bug，第二天才知道整个功能都被删除了；你不禁在心里怒吼："为什么不早点告诉我？"而敏捷就是要通过及时频繁的沟通，快速地对变化做出反应，将损失和风险降到最低。

02 CSDN

完整的跨职能团队就意味着很多人

理论上来讲，一个完整的敏捷跨职能团队应该具备完成业务目标的所有专业角色，包括：产品经理、前端开发人员、后台开发人员、设计师、测试人员等等。各队人马加到一起必然形成一个庞大的团队，规模如此庞大的团队一起开会，分分钟都是烧钱的感觉。

然而，敏捷开发虽然要求角色齐备，但在规模上应该有严格的控制，Scrum推荐的团队规模在5-9人之间。这样做目的在于：需要做出任何决策的时候，团队可直接做决定，不需要请示领导，也不需要正式的会议，在工作桌旁找一个空地，或大家围在白板前，一边讨论一边做笔记，不会超过30分钟就可做出决策，简洁高效，而且团队中的每个人都可以参与进来。

亚马逊著名的两个披萨原则，就是两个披萨能够吃饱的一个团队规模。简单来说，大概就是十来个人的团队。只有在这样的小团队里面，成员最具有活力，摩擦也最小，沟通成本又最低。而这样的一个团队有独立的自主能力，所以最能产生出创新。

03 CSDN

项目经理应该发挥领导的作用

正如第2条所述，一个完整的敏捷跨职能团队要求角色齐备，但各个专业角色上的人数却有严格的控制，所谓一个萝卜一个坑，有时甚至一个人负责多个角色（比如全栈开发人员），每个人都要负责好自己的专业领域。这是横向的组织结构，并没有层级。程序员不仅要写好代码，更重要的是与团队人员沟通——分析和理解需求，讨论解决方案，做好前后端以及其他接口，与测试人员分析和解决bug，向客户演示并说明产品的使用方法等等。

而项目经理在团队内的作用应该是协调和辅助，而不是上级对下级的领导。一个好的项目经理应该鼓励程序员多多沟通，而不是做他们的“代言人”，更不能下达指示。哪怕对方是客户或设计师，也应该由程序员与他们面对面的直接沟通，项目经理需要做的是帮他们联系相关的人员或安排会议（更像助理的角色）；或者是在他们遇到困难时帮助他们获得更多关注。

04 CSDN

每日的站立会议是汇报工作

敏捷开发最简单也最容易实施的莫过于每日的站立会议，但是人们往往把这个会议当成了工作汇报会议，这其实是一个严重的误区。

一般，每日的站立会议包含三个问题：

- 昨天的工作内容；
- 今天的工作计划；

- 遇到的困难。

实践过程中常见的形式：

- 昨天的工作成果汇报（昨天我忙了一整天）；
- 今天的工作计划（今天我也很忙）；
- 没有困难（我只负责写代码）；或困难很多（抱怨……）。

然而，这个会议真正的目的是促进团队之间的沟通：

- 昨天的工作内容：这个API我做完了（前端可以用了）；这个功能我做完了（测试可以开始了）；这个问题解决了（谢谢各位的帮助）；等等。
- 今天的工作计划：今天我要做这个画面（如果API有问题我可能会找后端开发）；今天我要做这个API（前端很快就能拿到了）；今天我要做这个测试（有问题可能会给你们bug票）；今天我要和客户开会（可能需求或计划会有变化）；等等。
- 困难：我的这个画面在等API（后端你可能需要调整工作优先顺序或加快速度）；我的设计在等客户反馈（你们可以先看看设计，但可能还会有变化）；我今天会联系客户（帮你问问那个问题）；等等。

这个短暂的会议应该更像一个小广播，每个人都可以接收到与自己的工作相关的最新消息。同时在你遇到困难时，也可以通过这个小广播引起所有人的注意。

05 CSDN

完美的工作成果

每个人都希望将自己的工作做到尽善尽美，通过展示完美的工作成果赢得领导的赞赏和同事们的钦佩。然而，在敏捷开发流程中，由于种种因素限制，完美的工作成果几乎不存在：

- 时间限制：通常每个迭代只有两周的时间，这其中包括设计、开发和测试等所有工作。

- 需求的不完整：敏捷是在迭代循环中不断改善和扩展的过程。项目初始，我们只需要构建最小可行性产品，然后在它的基础上通过迭代不断改善和扩展。
- 框架的不完备：在迅速开发的过程中，我们无法考虑周全每一种极端情况或边缘用例，也无法一次性将所有可能用到的技术和框架都包含进来，只能在必要的时候一点点添加和完善。
- 无时无刻不在的变化：客户的需求会变，技术也在不断更新，敏捷旨在迅速对这些变化做出响应，我们必须以开放的心态，随时迎接即将到来的变化。

除了受种种的因素制约之外，提前向别人展示未能尽善尽美的工作也会有意想不到的收获。比如在你展示的过程中，有人注意到了某些问题并及时提供了反馈，这样你就可以及时修正，从而减少返工。在与同事探讨的过程中，也许你会想到更方便更省事的解决办法。所以，团队成员之间应该展开亲密的合作，也许你走到同事的座位旁看一眼就能帮他发现一个bug呢。

另外，还需要注意一点：在种种因素的制约下，我们需要按照重要程度与紧急程度来划分工作优先级。相信大家都熟悉时间“四象限”，我们要利用有限的时间，为客户提供最重要最紧急的功能。我知道这一点很难，但是我们都要学会放手不重要不紧急的工作，容忍“不完美”。

06 CSDN

实施敏捷方法论就是向敏捷转型

很多公司举行每日的站立会议，以两周的Sprint为迭代周期，再加上Jira等管理工具，就堂而皇之地说已成功转型成了敏捷开发。然而仔细一看却发现：在分任务的时候，有的用户故事（user story）只有一个故事点（story point），而有的却有十几个（两周的时间只有十个工作日！）；在定义需求的时候，一个页面上有几十个字段，也不管这些字段的重要性以及将来会不会使用，就与客户挨个进行讨论；在讨论解决方案的时候，所有边缘案例都要讨论到，比如移动开发中考虑所有的设备类型；等等。种种迹象表明：他们本质上依然在遵循瀑布式开发流程！

这种形式主义根本无法贯彻敏捷的基本思想，结果只会适得其反，团队成员在两种模式的夹缝中束手束脚。

07 CSDN

一个迭代周期内的工作不均衡

在软件开发项目中，开发需要等设计，测试需要等开发，前端需要等后端，所以在迭代的头几天里，设计和后端忙的不可开交，前端和测试无所事事；而迭代的最后几天，测试加班加点，设计无聊得发慌；所以大家常常抱怨工作不均衡。

其实，这只是因为大家还没有完全摆脱瀑布式的阶段开发流程。在敏捷开发中，设计必须以开发人员提出的解决方案为基础，同时测试人员也必须明白客户的原始需求（而不是根据设计“推测”）；API等接口定义应该是由前后端共同商议决定，而且在接口确定下来（必要的时候甚至可以由测试人员提供一份模拟数据）之后，前后端可以尝试并行开发；测试人员写完测试用例也应该分享给所有人，一则帮助开发人员思考用户用例，也可以向设计师确认需求；在迭代快结束的时候，我想测试的bug票也够开发人员（代码bug）和设计师（设计bug）忙了。

08 CSDN

总结

敏捷开发是企业的一次深刻的文化变革，我们要以客户为中心，以满足客户的需求为最高原则，促进团队成员的沟通与协作，增强团队的自主性和灵活性，高效地应对一切变化。同时，我们也要合理地安排工作优先级，按照轻重缓急，持续交付最重要最紧急的功能。

- ☞ 李笑来登顶 GitHub TOP 榜！币圈大佬要教程序员如何自学编程
- ☞ 马云：蚂蚁金服这样做区块链！
- ☞ 云漫圈 | 女生适合做程序员吗？
- ☞ Google首页玩起小游戏，AI作曲让你变身巴赫
- ☞ 曝光！月薪 5 万的程序员面试题：73% 人都做错，你敢试吗？

```
System.out.println("点个在看吧！");
console.log("点个在看吧！");
print("点个在看吧！");
printf("点个在看吧！\n");
cout << "点个在看吧！" << endl;
Console.WriteLine("点个在看吧！");
Response.Write("点个在看吧！");
alert("点个在看吧！")
echo "点个在看吧！"
```

点击阅读原文，输入关键词，即可搜索您想要的 CSDN 文章。

喜欢就点击“在看”吧！

阅读原文