bobby · 更新于 2022-11-16

上一节 3. 配置jwt 5. jenkin的安装… 下一节

要理解敏捷开发，最好的切入点是知道：什么不是敏捷开发？

早些年，我参与过一个不小的项目，客户是一著名电视台。系统是用来支持当时大火的选秀节目，具体模块有场外观众短信支持，候选人得票情况的可视化显示，现场嘉宾的打分等子模块。总之，是一个功能蛮多，交互蛮复杂的系统。
因为在当年，选秀是一个新形态，所以作为开发人员，当然也不知道客户（电视台）想要的是什么东西。但最糟糕的是，连电视台自己也不知道想要的东西具体长啥样？而且因为是搞艺术的，想象力自然丰富，因此提的需求大概就是：
“要酷，要震撼，要能调动观众的积极性…”等等。总之就是需求很模糊，说不清楚。
但当时还不流行所谓的“敏捷开发”，常规的开发流程如下：

1. 需求分析
2. 设计
3. 编码
4. 测试
5. 交付

这也就是我们常说的“**瀑布模型**”。
可以现象，在该流程下，居于核心的就是需求分析。一旦需求分析出现大的偏差，之后的设计、编码、测试，即使做的再好，也是徒劳。因为你最后交付的根本不是用户想要的。
但是软件是这种东西，因为看不见摸不着。因而在具体的东西出来之前，大概率，用户根本就不知道自己想要的是什么。就好像电视台的工作人员，只知道 “酷、炸、炫”是他的需求，但如果再往具体处问，就不知道该说啥了。
但没办法，公司要挣钱，项目就一定要推下去，因此需求调研人员只能硬着头皮上。
虽然和电视台人员没少聊，但总体来说，也是言辞不详。最后的结果是，参考电视台的说辞，加上自己的猜测，再参考国外同类节目，晕晕沉沉弄了一个《需求分析》出来。
然后把《需求分析》传给了电视台，也不知道对方到底认真看了没有，反正最后的回复是：“没错，就是它，尽快出东西！”。
于是接下来，大家加班加点2个月，第一版终于出来了。欢天喜地给电视台看，本来大家的期望是掌声和赞美。但没想到，对方看了之后非常失望，说根本就不是他想要的东西。我们和对方争辩说，《需求分析》你也认可了，怎么可能不是你想要的东西呢?
结果对方也很委屈，答道：“《需求分析》我是看了，但这肯定不是想要的东西，你看这点…，这点…。”
原来，就这个《需求分析》来说，没想到同样的文字，不同人真会有不同的理解，甚至是完全相反的理解。
但这就是软件开发，因为看不见摸不着，高度定制，没有工业化标准。因此对用户来说，直到第一次看到实物（可运行的软件），才会逐渐在脑子中清晰他想要的是什么的东西。这就是软件开发的本质所在，没有任何人真的有错。但当时并不明白这个道理。
虽然很生气，奈何项目不小，对方又居于绝对强势。不爽归不爽，但回过头来，只有继续加班加点，吭哧吭哧的重新设计，重新编码。
因此，一个最初预期30个人月的项目，最终竟投入了100个人月还不止。
好在无论如何，最后，客户还是认可了我们的产品。但其中付出的辛苦，只有在身在其中，才会有真正明白。当然，人吃一个亏，总归会有些进步的。

，我们就会

尽量的在《需求分析》中增加图示。如果项目不大，甚至会先做出一个原型出来。
当然，这些做法效果肯定是有的。但也并没有从根本上解决问题。之后的项目还是会有返工的情况出现，区别只是严重程度的大小而已。
而我后来明白，这种情况（返工），只要是在“瀑布模型”下，就无法根本避免，因为来自于以下事实：

1. 只要有沟通，就会有信息的变形，因此《需求分析》是不可靠的。
2. 即使《需求分析》是可靠的，随着项目的进展，竞争对手的出现，需求也可能过时。
3. 瀑布模型，这种自上而下的开发方式，无法响应这个快速变化的时代。

而要解决这个问题，就必须引入更轻便的开发方式，这就是“敏捷开发”。
但是普通人可能没想到，敏捷开发并没有定义具体的开发过程，而是起源于一个简单的理念（确实够敏捷的），这就是《敏捷宣言》：
我们一直在实践中探寻更好的软件开发方法，身体力行的同时也帮助他人。由此我们建立了如下价值观：
>
个体和互动 高于 流程和工具；工作的软件 高于 详尽的文档；客户合作 高于 合同谈判；响应变化 高于 遵循计划；>
也就是说，尽管右项有价值，我们更重视左项的价值。
而之后的各种方法论，其实都是为了践行这个原则。

敏捷冲刺
因为世界是易变的，因此敏捷开发就不提倡像“瀑布模型”那样，一开始就制定整个产品的详细开发计划。而是提倡“小步快跑”的方式来实行整个产品功能。通过一个小计划接着一个小计划，通过反复迭代，最终实现产品的自我“进化”，自我完善。
在敏捷开发中，我们把每个小开发计划，称为一个“冲刺”。而实践中，常见的冲刺时间是2周，最多不超过1个月。
每个“冲刺”前，会有冲刺启动会，之后有总结会，一个冲刺的结束也是下一个冲刺的开始。同时，要在下一次冲刺开始之前对上一次冲刺进行总结。然后根据总结情况，制定下一次冲刺的具体内容。
能够看出，这种短节奏、快调整的开发过程，相对于“瀑布模型”，最大的好处是灵活多变，反应敏捷。任何时候，只要市场有变化，就马上调整下一步的开发计划。甚至是彻底放弃，及时止损。
精简文档
从事过开发的朋友都知道，文档是不可靠的。因为只要有文字，就会有曲解。更糟糕的是，面对快速的开发节奏，文档往往不能及时跟进。而错误的文档还不如没有。
因此，在敏捷开发中，始终强调“可运行的软件胜过一切文档”，也提倡在客户、产品、开发之间，通过软件本身来沟通。
因为是迭代式（冲刺）开发，因此每前进一步，都会有可运行的软件呈现。而软件是不会说谎的，因此就从根本上避免了沟通误差的问题。
每日站会
因为敏捷开发强调沟通，因此团队就必须提供时间和场景供大家沟通，每日站会因此而生。
在站会上，任何人（产品经理，测试、工程师）都可以分享昨天他干了什么？有什么困难？有什么需要帮助的？以及...
总之，就是通过及时的交流，把任何问题解决于无形之中。
敏捷教练
敏捷开发极度强调过程，强调沟通，强调团队合作。因此就需要有一个人站出来，作为敏捷过程的组织者、督促者、协调者。这个人就是“敏捷教练”。
很重要的一点：敏捷教练不是官。而是接头人，是组织者。需要的不是权威，是责任心。
结对编程
敏捷开发极度强调沟通的价值，也相信在软件开发中，1+1 会大于2，尤其是在解决复杂问题的时候。因此，在敏捷开发中，如果队员出现某些个人无法解决的难题，他可以申请结对编程。
具体的做法是一个编程，一个人在坐在旁边“指导”，然后过一段时间再交换。
这样，就会极大的避免面临复杂问题是，一个人思路彻底受阻，从陷入思维“死胡同”的情况出现。
.最后
当然，敏捷开发也绝不是解决一切问题的银弹，它也有自己的适用场景，具体来说，敏捷开发特别适合互联网等市场竞争激烈，需要快速响应的场景。

因此，对这

些项目来说，“瀑布模型”（或者变种）依然是首选。其中的关键是在动手之前，对《需求分析》反复论证，直到万无一失。
但总的来说，软件首先是“软”的，要解决根本问题，还是要靠其中的人（产品、测试、开发）。因为软件是“软”的，则务必要求软件的从业人员也是“软的”，从这个角度出发，软实力是每个软件从业人员走向优秀的必备之道。

最后总结一下：

敏捷开发：我们也不知道到底要开发啥，走一步看一步吧
用户故事：老板说明天要上这个功能，怎么实现我不管
快速迭代：上次做的功能点击率太低，把广告改成全屏
用户痛点：昨天用户投诉了，把广告调整下
拥抱变化：老板天天都有新想法，大家要适应（不要怪我）
持续交付：每个版本都有问题，总是持续交给测试，交付间隔10分钟
结对开发：bug太多了，直接去测试妹子的工位边测边改
代码评审：这个代码是你审的，将来出了问题你是要负责任的
弹性工作：不限定下班时间，修完bug才能走
四个润会：（Scrum Meeting）每天早上9:00开始，想上班迟到没门

3. 配置jwt 上一节 下一节 5. jenkin的安装和配置

我要提出意见反馈

企业服务 网站地图 网站首页 关于我们 联系我们 讲师招募 帮助中心 意见反馈 代码托管

意见反馈 收藏教程 标记书签