好文精选|浅谈对技术的几点感悟

更新于2019-09-24 18:36:02

41	0	1	1
阅读	回复	评分	收藏

1 版主
笔落惊风雨 楼主

1	0	18
粉丝	关注	帖子

（转自《HR刊物》（2019年第1期）夯实基础，砥砺前行）

明旭 | 互联网业务中心

我在技术研发的岗位上工作13年，来萤石也有5年多了。工作年限虽然不算短，但按照《刻意练习》一书中阐述的： “1万小时的刻意练习”才有可能成为专家的话，我连时间这个必要条件都达不到。因此，这里我只能谈自己对于技术本身的理解，以及技术研发这个职业的几点感悟，一些是我自己曾经走过的弯路，远算不上金科玉律，主观感受偏多，对与错、是否适合大家很难说，只能说做个参考。

感悟1：技术+实践=认知

我们看多少本书，读多少文章，甚至深入探究某技术的实现原理，这只能是技术的范畴，这个层面的掌握对于工程师来说是不够的，还要有将所掌握的内容应用于**实践**，应用到产品当中，解决了实际问题，才有可能产生我们对某项技术的认知。 这一点我想很多工程师都有体会，不过我在工作当中遇到下面几种情况：

为了应用而应用： 有时候看到某项技术很好，就有一种冲动，一定要在项目中应用它，有时候忽略了产品的实际问题、痛点。再新再炫酷的技术，如果不能真正解决实际问题，就不值得我们引入它。 我在评审这样的方案的时候通常都是“**5个为什么”原则**， 为什么要引入这项技术？ 到底要解决什么问题？ 仔细探究下来会发现要解决的问题和引入的方案并不是完全契合的。我对此有个经验性的判断标准，就是我们是否知晓这门技术的弊端是什么，如何缓解、权衡这些弊端？??如果说不出来，通常代表这我们对这门技术的认知还比较有限。

技术藏起来，不应用： 在解决某个问题的时候，不去研究是否有等价的、甚至相同的问题已经有人解决了。而是从if else开始。或者用比较“挫”的方法解决一个比较困难的问题，而不是去思考是否有现成的模型、方法、工具来辅助自己解决这个问题。 如果一个问题看起来应该是别人也可能遇到过的，那么首选要做的是，看是否有现成的东西可以直接引入，而不是“造一个轮子”出来。吴军在《谷歌方法论》中讲到，一个好的工程师需要**善于寻找等价问题**，把精力放在那些还没有被解决的问题上，否则我们能够解决的问题是比较有限的。《技术的本质》一书中对技术的定义是： “技术是实践元器件（components）的集成（assemblage）。” 其中的内涵也是技术的本质体现在对已有东西的组合上，即**组合能力**才是作为技术人员的核心竞争力。

感悟2： 技术人员要理解产品、理解用户

作为研发人员，我们当然希望自己的设计、代码重用性很好，这样比较有成就感。为此，我们掌握了很多这方面的技术，诸如《代码整洁之道》、GOF《设计模式》、马丁福勒的《重构：改善既有代码的设计》等等。 毋庸置疑，这些都是很好的技术和方法。但还有一点容易忽略的是： 它们都是建立在对业务有深刻理解的基础上。如果没有这个前提，我们是很难做出好的设计和代码的。从需求中提取业务模型，以业务模型为基础转化为实现模型，这样的设计和代码不能说完美，但通常差不到哪里去。反之，通常好不到哪里去。好的设计和代码，它们通常与业务保持很好的一体性，这个特性很重要。这是我结合《领域驱动设计》和实际项目经验中体会比较深刻的一点。 研发人员有时候会觉得自己做的事情没有技术含量，对自己的技术能力提高没有什么帮助，每天在做重复的事情。 这里我不能说所有的项目、业务都是有高技术含量的，但可以肯定是，**理解业务**本身就是一件比较有技术含量的工作，尤其是对技术人员来说。 因为理解业务就意味着我们起码要用两个视角看待同一个系统：**业务视角的What +技术视角的How**。 而使用多个视角去看待同一个系统的技能并不是那么容易掌握。因此，理解产品和理解业务虽然不是一名技术人员进阶的充分条件，但通常是必要条件。产品和技术一样都是有生命力的，并非一成不变的，我们应该两者兼顾。

感悟3： 接受别人的不完美和认识自己的不完美

我们在日常工作中会和上下游的很多人打交道，在信息传递和工作成果的传递过程中，不可避免地会出现上游传递的信息、工作成果有问题，下游对自己的信息理解有误等问题。对此，有人从自身的角度解决问题，有人从别人身上去解决问题。 前者思考的问题是： 我自己怎么做才能让工作更顺畅；而后者思考的问题是：别人怎么做才能让我更舒服。

我的体会是，**要求自己比要求别人更有效**，没有必要把精力花在那些难于产出结果的事情上面。 比如，我经常遇到在项目总结的时候看到研发抱怨，为什么需求总是变来变去，下次能不能请产品经理不要变需求了，或者在某一个阶段不要变需求了。结果通常是： 下一个项目仍然是这个问题。 实际上我们更应该关注的事情是： 变更本身是否合理、以及我们的应对变更的方式是否合理，带来了哪些问题等。产品经理不是上帝，不太可能一下子做出来的东西就是符合用户需求的， 而我们自己也不是上帝，并非项目周期中表现得完美无缺，多想一想自己能够为改进项目、产品的质量、效率做些什么，才是比较正向的思考方式。 我对团队成员的要求是： 遇到问题不要先想着别人哪里没做好，而是先想想自己哪里没做好。 给上下游同事提要求不是不可以，但一定要具体，不要用诸如： 质量太低了、问题太多了、覆盖度不够等，越具体越好，否则我们的要求很难收到效果，对没有效果的事情做努力其实是在浪费自己的时间。

感悟4： 按照高一个层级的视角要求自己

这是我在极客时间中的《朱赟的技术管理课》中看到的，觉得总结的很好。 一名员工被提拔了，通常是因为他已经在按照高一个层级的视角来要求自己了，而不是因为他首先被提拔了，才按照高一个层级来要求自

己。每个人的岗位是有边界的，但这是由于分工协作的需要，但并不是说我们就把眼界放在自己的岗位边界内就能够做好工作了。 我自己的总结是：**“边界内做事，边界外看事。”**如果我们能够从自己主管的视角来看问题，通常能够把事情做得更好。 我们也许经常遇到难于权衡的问题，比如做一个功能能够让响应速度快10%，但改造工期需要2个月，到底应不应该做呢？此时大家通常都会上报领导做决策，当然还有同事直接以工作量太大为由，觉得不太可能做这件事情，连上报都懒得上报了。我在萤石工作了5年，接触到工作努力的同事很多，这一点上我自愧不如。但在敢于做决策、理性分析和判断的同事并不多见。一个挺小的问题也要抛给大领导做决策。比如前面的问题， 如果我们原封不动的抛给领导，我认为就是给领导出难题了。如果我们自己站在领导的位置，如何决策呢？ 如果只有这些信息，还是无法决策的。快10%对于用户来说重要么？ 自己手头有没有比这更重要的工作？竞品的响应速度如何？ 这些都是支撑我们决策的要素。我们让领导决策之前，需要把这些支撑决策的信息收集好，如果信息量足够，通常我们自己也知道该如何选择了。

感悟5： 技术改造类项目，目标驱动的技巧： 指标、目标与权衡

我们在评价所做的工作、产品等，都需要设定明确的目标，以目标为导向行事。但如何定义目标、甚至是目标是什么经常会困扰我们。比如一个系统我们觉得太难维护了，要重写一遍-即重构。但重构的目标究竟是什么呢？ 显然重构本身不能成为目标，它只是手段，可维护性又如何来衡量呢？我遇到的一些项目，在文档上就写“系统越来越难于维护，因此重构势在必行。” 下面就开始出设计方案了。但这样太草率了， 草率到我们可能都无法判断最后是否达到了既定的目标。?我通常会问下面的问题：

不做这件事情到底会怎么样？--**帮助我们正确的认识到底要解决什么问题**

重构的好与坏到底与那些因素有关---**指标**

这些指标之间的关系是什么，如何取舍?---**权衡**

这些指标，我们应该做到什么程度---**目标。**

在《SRE：Google运维解密》一书中，针对一个系统的可用性拆分成为两部分：SLI（可用性指标）和SLO（可用性目标）。而在定义可用性目标的时候通常需要做权衡，比如下文将会提到的分布式架构中的CAP理论。这样的做法在诸多比赛场上运用得很成熟。如NBA球员好与坏，有得分（直接产生得分）、篮板（为本组贡献一次得分的机会）、助攻（辅助队友得分）、失误（给对方一次得分的机会）这样的主指标（当然还有在场时间内的胜负值等指标）， 然后用这些指标来衡量一名球员的素质。其中也会有权衡的问题，诸如： 可以自己得分，也可以传球给队友，此时应该怎么做等。

举一个和同事做过的短信网关的例子，短信网关的目的是给指定的手机号，通过调用短信供应商的接口，发送指定内容的短信。我们系统对短信网关做重构：

不做重构会怎么样： 供应商故障会导致短信发不出去，从而导致用户注册等关键操作无法进行，从而导致用户流失、退货。

如何评价重构的好与坏：

成功率：发送成功的短信/总短信，成功率越高越好；

延迟：发送到用户接收到的时间，延迟越低越好；

成本：即用了多少钱发送了短信，成本越低越好

体验约束：最好不要给用户发送重复短信

权衡：高成功率、低成本、低延迟、不重复肯定是最好的。但供应商A可能成功率高、但延时高，供应商B延时低，但成功率不高等。可能还有C供应商成功率高、延迟低、但成本也高等等。我们到底选谁呢，这就需要权衡了。 比如延迟3秒和5秒其实区别不大。又比如我们给用户发送广告，98%成功率和99%成功率差别也不是很大，但用户注册环节差别则是比较大的。

此时我们可以根据特定的场景做不同的组合策略：

注册场景：高成功率、限定范围内的延迟，可以接受较高的成本（单个用户的价值远高于这个成本，没有必要因为省一分钱导致一个用户注册不成功）， 那就可以做供货商互备，质量高的优先，但质量其次的供货商作为备份，防止A供货商出问题短信发送失败，在未知结果的情况下，可以接受发送重复的短信，因为用户来注册了， 发送一个短信验证码，发送两遍通常比发送不到要好；

广告场景：延迟不那么重要，成本和成功率是优先考虑的，供货商互备的意义不大。

除此以外： 还需要建立供应商量化的评价体系，针对供应商延迟、成功率、成本进行综合评价，以便于我们精确筛选供应商。

经过这些分析之后，我们就可以定义相对量化的目标：

如：高优先级（如注册场景）成功率99.99%+， 延迟小于30S，成本XXX； 低优先级（如广告场景）的成本成功率99.9%，延迟小于24H，成本XXX。 这样就可以有的放矢地开展详尽的设计方案，以及针对实际结果是否达到预期目标的判定。

感悟6：保持终生学习的习惯

我所从事的工作同时受到物联网和互联网技术浪潮的冲击，每天的新知识、新方法都觉得学不过来。一段时间不跟进就觉得自己落后了，如果几年不跟进，就不太好意思说自己是做技术的了。我们可以将要掌握的技术拆分为两类：

经久耐用型：这些理念几年、甚至几十年都不过时，但要深刻理解它并非易事，这样的技术需要用匠心反复思考和实践才能掌握。 比如C（一致性）A（可用性）P（分区容忍性）理论，我最开始虽然也理解了是什么，但并没有觉得有什么用。但后来发现这理论太有用了，几乎我涉及到的分布式架构都会有应用， 无论是把服务放在多个服务器上的微观问题，还是更大规模的数据中心的设计考虑，都离不开CAP的影子，经常在C、A、P的几个保障之间思考如何权衡的问题，哪个要素需要强一些，哪个要素应该弱一些；

持续更新型：?技术更新非常频繁的，可以说我们生活在最好的时代，因为我们能够看到如此活跃的技术氛围；也生活在最差的时代，因为做一个技术的跟随者都会觉得很辛苦。11月份去上海参加了CNCF（云原生）的技术会议，其中的工具、项目在一页PPT上密密麻麻的铺满了，把每个项目是做什么用的弄清楚都需要花上好几天了。我不禁感慨，自己确实需要多多努力了。对于这样纷繁复杂的技术，我认为还是需要理解背后的本质，其背后的本质通常都是经久耐用型的。**从本质的视角出发去理解、应用**，会让我们事半功倍。

作为技术人员，这两方面的技术都应该做跟进和应用。我每年大概会看几十本技术方面的书（当然，也会看些其他方面的），仍然觉得自己有些Out。作为技术人员，需要对技术保持敬畏之心，不断给自己学习的动力，才有希望把技术做好。

收藏

楼主本帖1条评分记录，共获得 威望：1分

+1	太给力了~~~	回帖	2019-09-24 / 双脚落地

评分	回帖	收藏	收录

快速回帖 使用高级回帖(可批量传图、插入视频等)

- 回帖时多说几句，会让你的发言得到更多关注

表情

用户名： 密码： 新用户注册

Ctrl + Enter 快速发布