43 | 无法实现：困扰与反思

2018-11-09 胡峰

程序员有句口头禅叫：“技术上无法实现！”这句话，在我过去多年的程序员职业生涯中经常听见，甚至我自己就曾说过很多次。如今，当我再次听到有人说出这句话时，不禁开始反思起来，为什么程序员爱说这句话呢？为什么曾经我也时不时说这句话呢？

一仔细思考，就惊讶地发现一个事实：这句口头禅背后隐藏着一个阻碍我们成长的陷阱。

一、困扰

当接到一个需求或碰到一个问题，我们回上一句：“技术上无法实现！”这是真的无法实现吗？还是隐藏着其他的困扰？

1. 不知

当我刚开始工作的第一年，我在一家银行客户现场工作。当时要给银行的出纳管理部做一个系统，这个系统有个功能就是上传各个国家的高清真假币鉴别对比图片，然后银行的出纳和柜员就可以在系统上学习各个国家纸币的鉴别方式了。

针对这些高清纸币图片，客户因为怕别人盗取乱用，就要求必须对图片做加背景水印的功能。当我们在召开需求讨论会时，我听到这个需求就懵了，因为完全不知道要怎么做。毕竟当年我才刚刚开始学习如何做 Web 化的管理系统，从来没有用程序处理过图片。

彼时，当我想起程序化的图片处理时，我就只能想起像 PhotoShop 那样高度专业化的图片处理工具软件，觉得这肯定是一个很复杂的事情。所以，当我们讨论起加背景水印的功能时，我自然

脱口而出：“这在技术上无法实现！”

然后我们进一步谈起，当前客户他们是怎么做的。客户确实是找了专门的外包设计公司用 PhotoShop 来给图片一张张手工加水印。这听起来就是一个比较繁琐的过程，所以，当我回答“在技术上无法实现”时，客户都是业务人员，也不太懂程序技术上的事，听到我的答案也就略显失望。

好吧，如今回想起来，我说“技术上无法实现”时，仅仅是因为当时的我并不知道如何去实现。而且想当然地感觉要进行图片处理，必须要具备有 PhotoShop 一样的专业背景知识，而这对当时的我而言是完全不能想象的。

因此，当时我说出的那句“技术上无法实现”，仅仅是因为不知和不解而心怀畏惧。因为畏惧，所以我用了这句口头禅来回避了这个问题，甚至没有去调研一下技术可行性，就由此固步自封，在这片知识领地留下了一片空白，也不能为客户创造更进一步的价值。

“技术上无法实现” 的口头禅，此时成为了遮挡我们因不知而畏惧的面具。

2. 不愿

有一年，我出差在客户现场赶项目，连续加班了四个周末，也就是大概一个月在连续上班。终于我们的项目快要如期上线了，每个通宵的早晨，看着东方慢慢变得红润透亮的天空，感觉已经快要看到胜利的曙光了。

就在这样一个曙光照耀的早晨，项目经理跑来对我们说：“原有的一块业务逻辑今天在和客户聊起时，他们说也只是试试这个流程，可能要改变。但我们的实现方式太僵硬，都是硬编码赶出来的。要不我们改成更灵活的、可以通过配置的方式，一旦上线后再改起来就更麻烦了。我可以先去和客户再沟通下，给我们再争取点时间。”

一下子，我们都被打击得不行，改成配置怎么改？逻辑那么复杂，又不是那种简单的开关式配置。当时，项目经理早已脱离技术一线时间颇久，也一时半会儿没啥方案。在沉默地思考了一阵后，我又说出了那句话：“逻辑太复杂，变动太大，这短期在技术上无法实现的。”

其实，那时我心里是有一个方案的，如今看来虽不是什么优秀的方案，但也是当时我唯一知道且可行的方案。就是通过 Java 的动态类加载机制，把业务逻辑外移，流程内置的方式以便可以动态热加载新的业务逻辑类。但这意味着可能要面临一次重大的重构，又是两周的持续加班，而我当时只是想赶快离开这沉默的讨论会，去美美地睡上一觉。

后来，这个故事在我睡醒后依然以我妥协结束。我建议了这个方案，最后当然也是我去实施了这个方案，庆幸的是并没有如“预料”那般加上两周班，只用了一周，项目就上线了。再之后的后续维护中，我又学习了新的东西，流程引擎，动态脚本，继续下一版本的重构，我们升级成了一个更好、更通用的方案。

当时我说出的那句“技术上无法实现”，只是因为觉得很麻烦，不愿意而已。后来睡醒后，回了一些血，有了能量，觉得应该接受这个挑战。因为客户的需求变化就是一个客观事实，也不会因为我的主观意愿而改变。

“技术上无法实现” 的口头禅，此时成为了我们推脱的借口。

二、反思

不论是 “不知” 还是 “不愿”，“技术上无法实现” 的口头禅看来都不会给我们带来什么帮助，它反而阻碍了我们进一步做出更好的产品，从而给客户留下遗憾。

随着工作经验的增多，技能的积累，我便越来越少说这句话了。事实上，我发现大部分的用户需求，技术上总是可以实现的。这些需求的真正限制，要么是时间，要么是资源。

所以，面对一个紧迫的或不合理的客户需求，甚至诉求时，不应该再以如此苍白的一句话去应对了。这个需求背后涉及的技术实现，要么可能你现在未知，要么你至少知道一个方案，只是觉得过于复杂，而且会带来很多“副作用”，所以不愿意这样去做罢了。

但总之，你需要一个办法去应对一个让你觉得 “技术上无法实现” 的需求。我建议不要立刻像我当年那样做出如此简单的判断就推脱过去，其实我们完全可以把这样的问题放在下面这样的框架中去思考下。

1. 全局背景

这一步的目的并不是要找到并确定实现方案，只是对这一问题涉及主题的相关内容有一个全局性的了解。

近年我都在做京东咚咚，一个 IM 系统，所以就以此举个例子吧。不时我们会收到用户反馈在安卓客户端应用切到后台就会收不到消息，这里用户只是提供了一个说法，甚至都不算现象。但这是一个问题，而且是一个我觉得在技术上无法百分百根除的问题，换言之就是我可能想不出一个方案能让我的所有用户都再也不会碰到类似的问题。

而用户碰到这样的问题可能的原因有：

- 移动弱网络，消息投递失败率高；
- 应用切后台就被系统杀掉，所以收不到；
- 第三方推送渠道，比如：某一类用户完全没有这种渠道可达；
- 应用本身的问题，比如：Bug，版本碎片导致的兼容性问题。

以上简单的问题分类，背后都隐藏着一个解决或优化问题所需的巨大且复杂的实现方案。针对每一类问题的方案，可以先去大概有个了解，但这里还不需要很深入。

2. 聚焦范围

对上面列出的全局背景问题分析分类后，会发现没有一个是轻松容易就能解决的。而且这时还必然面临资源和时间的问题，也就是在特定的资源和时间下，我应该优先解决哪类？所以，这一步的本质就是从上面的全局分类中，聚焦一个范围，然后集中深入调研评估。

3. 定义标准

前面说了用户仅仅反馈了一个说法，站在用户的角度，他们总是希望没有任何问题的。但站在我的角度，我知道我只聚焦了一部分问题，所以我需要清晰定义这部分问题解决的成功标准。

比如，针对应用切后台就被系统杀掉，对用户无感知，所以认为收不到消息是有问题的。针对这个问题的聚焦范围，我可以提供第三方推送渠道在十分钟内的推送通知补偿，重新唤醒用户重回应用，避免消息的遗漏。通过估算每日活跃用户和可能投递给第三方渠道消息通知量以及第三方渠道自己标榜的投递成功率和业界一些经验数据，就能估算出该解决方案的标准：通知唤醒到底能补偿多少用户的指标。

4. 深度评估

有了范围和标准，剩下的就是深度评估方案路径问题。大体上任何一个方案，其中有些是你已经轻车熟路的实现路径，另一些则是你可能从未走过的陌生之路。

轻车熟路的部分可能更容易评估，但很多程序员还是容易高估自己；而从未走过的陌生之路，就评估得更离谱了。关于评估，可以保守一些，因为一般来说现实总是比理想的路径曲折一些的。

经过了上面四层思考框架的过滤，这时想必你已成竹在胸了，并能很好地衡量该技术实现方案的成本与收益。除此之外，进一步还需斟酌考虑的是方案是否足够优化，毕竟我们做工程就是要找到一条最优化的实现路径。

当面对任何一个需求，除非能一下从理论上发现其实现的物理限制，我们恐怕不能够再轻易说出“技术上无法实现”了。即使真的是无法实现的需求，也有可能通过变通需求的方式来找到一条可实现的路径。“技术上无法实现”的口头禅仅仅是我们阻挡需求的快捷方式，但这样的思维也阻碍了我们进一步去找到真正的实现路径和优化方案。

- “你看这个需求能实现么？”
- “哦...”

改掉了这句口头禅后，有些问题也挺难简单地回答了。

那你是否爱说这句口头禅呢？又是在怎样的情境下说的呢？

极客时间

程序员进阶攻略

每个程序员都应该知道的成长法则

胡峰 京东成都研究院 技术专家