38 | 思维：科学与系统——两类问题的两种思维解法

2018-10-29 胡峰

写了多年代码，做了好多的工程，不停地完成项目，但如果你一直仅仅停留在重复这个过程，那么就不会得到真正的成长与提高。你得从这些重复做工程的过程中，抽象提炼出真正解决问题的工程思维，用来指导未来的工程实践。

什么是**工程思维**？我从自己过往经验中提炼出的理解是：**一种具备科学理论支撑，并成体系的系统化思维**。做了多年的软件开发工程，碰到和解决了数不清的问题，最终这些问题，我发现稍微抽象一下，可以归为以下两类：

1. 可以简单归因的问题：属于直接简单的因果关系；
2. 难以简单归因的问题：属于间接复杂的因果关系。

上面的描述可能有点抽象，那具体该怎么理解呢？这里我分别举两个例子：线上有个 Bug，找到了有问题代码片段，需要一个优化实现方案来解决，这就是第一类问题，原因和结果非常明确清晰；线上老是出故障，而且反复总出意外故障，对于这个结果，它的原因是什么，这就很难简单归因了，就属于第二类问题。

对于这两类问题，我想讲讲两种不同的思维框架提供的解法。

科学与理论

第一类问题，现象清晰，归因明确，那么它唯一的难处就是为这个问题找到最优的解决方案。求解最优化问题，就需要科学与理论的支持，也即：**科学思维**。

先讲一个其他行业的故事：造船工程。很早以前，关于应该造多大的船，人们都是靠感觉摸索的。后来（十九世纪中期）有个英国工程师布鲁内尔（Brunel）意识到船应该尽可能造得大些，于是他设计了当时世界上最大的船。这是一艘挑战当时工业极限的船，该设计甚至还引发了当时社会激烈的辩论。

布鲁内尔的目标是建造一艘足够大的船，大到无需中途停留，直接能从英国开到印度，那么如此远的航程就需要有足够的货物与燃料（那时的燃料主要就是煤）的装载能力。而支撑他设计背后的理论却很简单，船的装载能力是体积决定的，跟船尺寸的立方成正比，而船航行受到的阻力则是和船底的面积成正比。所以，船越大，装载能力越大，但单位载重量的动力消耗却下降了，这就是为什么布鲁内尔要尽可能地造大船。

这就是科学理论给予造船工程的方向指引。吴军老师也曾在一篇文章《计算机科学与工程的区别》里指出：

> 科学常常指出正确的方向，而工程则是沿着科学指出的方向建设道路；在工程中必须首先使用在科学上最好的方法，然后再作细节的改进。

我做在线客服系统时碰到一个问题和滴滴打车的匹配问题非常类似，打车是人和车的匹配，而咨询客服是人和客服的匹配。抽象来看，这个匹配的算法并不复杂，但因为涉及到非常具体且繁琐的业务规则，实现起来就有特别多业务逻辑，导致性能有问题。这就是软件工程现实中的第一类问题，需要找到优化方案。

对于这类问题的解法，就是先用计算机科学理论来分析其性能的复杂度边界与极限，而咨询分配就是在 N 个客服里进行挑选匹配，每次只匹配一个人，所以理论复杂度极限是 O(N)。只要 N 有限大，那么匹配的性能最坏情况就是清晰的。

理论分析给出了边界，工程实现则是建设道路，这就需要在边界里找到最短路径。在客服匹配问题的工程实现总考虑的方式是：最坏的情况是每次匹配都要遍历 N 次，最好的情况是 1 次，那么实现方案评估就是尽可能让最好的情况发生的概率最大化。假如你的实现方案 90% 的场景概率都发生在最好情况下，10% 的场景发生在最坏情况，那么整体性能表现可能就比最坏情况高至少一到数个量级。实际提高多少，这取决于 N 的大小。

而另一个工程实现考虑的维度是，如果每次匹配中有 M 个高消耗操作，那么进一步的优化方式就是如何减少 M 的个数或降低每次操作的消耗。

这就是用科学思维来指导工程实践，科学理论指出方向，探明边界，工程实践在边界的约束范围内修通道路，达成目标。正如前面故事中，造船理论往大的方向走也有其极限，因为除了能源利用率的经济性外，越大的船对其他建造、施工和运营方面也会带来边际成本的提高，所以也就没法一直往大里造，这就是工程现实的约束。

所以，理论的意义不在于充当蓝图，而在于为工程设计实践提供有约束力的原理；而工程设计则

依循一切有约束力的理论，为实践作切实可行的筹划。

简言之，科学理论确定了上限，工程实践画出了路线。

系统与反馈

第二类问题，结果明确，但归因很难，那么找到真正的原因就是第一个需要解决的难点。这时，我们就需要用另一种思维方式：**系统思维**。

回到前面举的例子，线上老是出故障，而且反复出意外故障。如果简单归因，查出故障直接原因，发现是代码写得不严谨，实现有不少漏洞和问题，仔细看就能分析出来，但触发条件罕见不容易测出来，于是提出解决方案是增加代码评审（Code Review）流程来保障上线代码的质量。

关于代码评审就是我从业多年来遇到的一个非常有意思的问题，大家都觉得它有用，也都说好，但很多时候就是执行不下去。因为它不是一个简单问题，而是一个系统问题。万维钢在《线性思维与系统思维》这篇文章里，给出了一些系统问题的典型特征，其中有两条是这样说的：

- 多次试图解决一个问题，却总是无效；
- 新人来了就发现问题，老人一笑了之。

我呆过的很多公司和团队，都想推行代码评审，最后都无果而终。反而是一些开源项目，还搞得有声有色。还是万维钢的那篇文章里，其对系统的定义：“所谓系统，就是一个由很多部分组成的整体，各个部分互相之间有联系，作为整体又有一个共同的目的。”简单想想就会发现公司项目所在的“系统”和开源项目所在的“系统”其构成就完全不同，而且目的也不同。

> 一个系统中可以有若干个正反馈和若干个负反馈回路，正反馈回路让系统或者增长、或者崩溃，是要偏离平衡，负反馈回路则尽力保持系统的平衡。
>
> 对你想要解决的这个问题而言，可能就有一个回路，正在起主导的作用！如果你能发现在系统里起主导作用的回路是什么，你就抓住了系统的主要矛盾，你就找到了问题的关键所在。

曾有行业大牛在前公司有很好的代码评审传统和流程规范要求，自己也坚决支持代码评审。后来去了另一个同行差不多规模的公司，进入到团队后想推行代码评审时，就遭遇了巨大的阻力，不止是“老人呵呵，一笑了之”了，还甚至被公开地反对了。显然，对于代码评审这个问题，他的前后两家公司拥有完全不同的正、负反馈回路，以其个体之力，想要去改变已有的反馈回路，其实相当艰难。

我自己也曾在团队做过一些尝试，但一直找不到合适地建立正反馈回路的好方法。引入严格的代码评审流程，其负反馈回路立刻发生作用：更多的工作量，更多的加班等。负反馈，团队立刻就能感知到，而其正反馈回路发生作用带来好处却需要一定的时间。而且另一方面，建立新的回路或者摆脱当前的循环回路，还需要额外的能量来源，也即激励。

在解决系统问题，建设正反馈回路上也有过成功的样本。比如，在公司层面要求工程师产出专

利，这对个体来说就是额外的负担，而负担就是负反馈。为了降低负反馈回路的作用，可以让专利和晋升到一定级别关联上，并增加专门培训来降低写作门槛；专利局每通过一份专利，就奖励一笔奖金（几千到上万），甚至没通过都能奖励几百块，这些就是建立正反馈循环回路的激励能量。

另外一个例子是，为了让程序工程师们更有分享的意愿和提升表达能力，就出一个规则，把分享和每年的晋升提报关联起来，本质就是提供了潜在可能的经济激励。经济学原理说：人会对激励做出反应。是的，经济学原理很有效。

软件工程，是研究和应用如何以系统性的、规范化的、可度量的过程化方法去开发和维护软件；而实际上软件开发本身就是一个系统工程，里面存在很多没法简单归因的第二类问题，它们没有通用的解法，只有通用的思维。

一个优秀的工程师应该同时具备科学思维和系统思维，它们是工程思维的两种不同表现形态：**系统思维洞察问题本质，科学思维发现最优解法**。

学完本章，你也可以去细心观察你周围的环境，看看都有哪些问题，属于哪类问题，以及你能发现它们的本质吗？欢迎你留言分享一二。