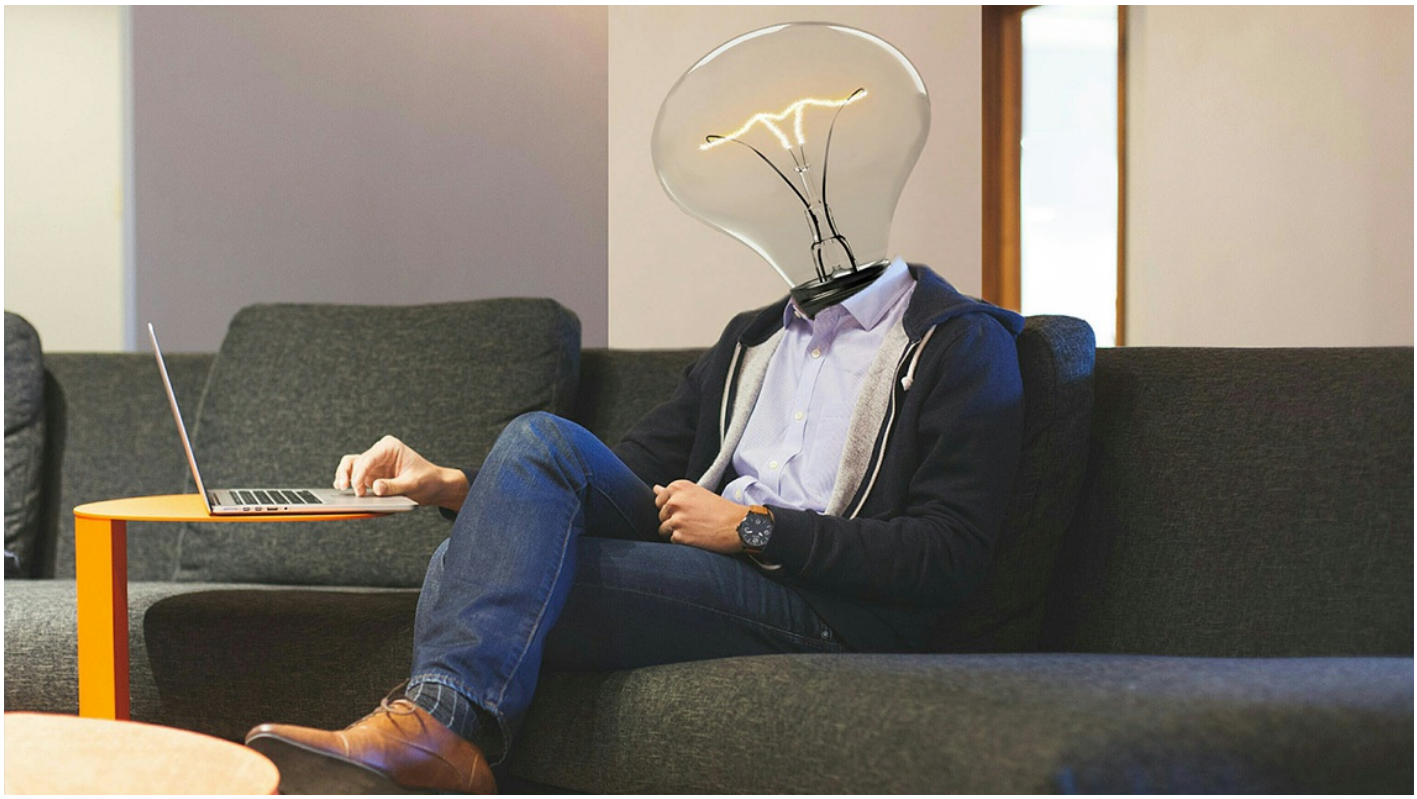
27讲尽早暴露问题：为什么被指责的总是你



今天我准备讨论一个经常会让很多程序员郁闷的事情，为什么你已经工作得很辛苦了，但依然会被指责。在讨论这个问题之前，我们先来讲一个小故事。

程序员小李这天接到了一个新的任务。系统要做性能提升，原先所有的订单都要下到数据库里，由于后来有很多订单都撤了，反复操作数据库，对真正成交过程的性能造成了影响。所以，技术负责人老赵决定把订单先放到缓存里。

这就会牵扯到一个技术选型的问题，于是，老赵找了几个可以用作缓存的中间件。为了给大家一个交代，老赵决定让小李给这几个中间件做一个测试，给出测试结果，让大家一起评估。小李高兴了，做这种技术任务最开心，可以玩新东西了。

老赵问他：“多长时间可以搞定？”

小李说：“一个星期吧！”

老赵也很爽快，“一个星期就一个星期。不过，我得提个要求，不能是纯测中间件，得带着业务跑。”

“没问题。”小李一口答应下来。老赵怕小李做多了，还特意嘱咐他，只测最简单的下单撤单环节就好。

等真的开始动手做了，小李发现，带着业务跑没那么容易，因为原来的代码耦合度太高，想把新的中间件加进去，要先把下单和撤单环节隔离开来。而这两个操作遍布在很多地方，需要先做一些调整。

于是，小李只好开始不分白天黑夜地干起来。随着工作的深入，小李越发觉得这个活是个无底洞，因为时间已经过半了，他的代码还没调整完。

这时，老赵来问他工作进展，他满面愁容地说，估计干不完了。老赵很震惊，“不就是测试几个中间件吗？”

小李也一脸委屈，“我们为啥要带着业务跑啊？我这几天的时间都在调整代码，以便能够把中间件的调用加进去。”

老赵也很疑惑，“你为啥要这么做？”

“你不是说要带着业务跑吗？”

“我是说要带着业务跑啊！但你可以自己写一个下单撤单的程序，主要过程保持一致就好了。”小李很无奈，心里暗骂，你咋不早说呢?

是啊！你咋不早说呢？不过，我想说不是老赵，而是小李。

谁知道有问题?

我们来分析一下问题出在哪。在这个故事里，小李和老赵也算有“以终为始”的思维，在一开始就确定了一个目标，做一个新中间件测试，要带着业务跑。

小李可以说是很清楚目标的，但在做的过程中，小李发现了问题，原有代码很复杂，改造的工作量很大，工作可能没法按时完成。

到此为止，所有的做法都没有错。**但接下来，发现问题的小李选择了继续埋头苦干，直到老赵来询问，无奈的小李才把问题暴露出来。**

在老赵看来，这并不是大事，调整一下方案就好了。但是小李心生怨气，在他看来，老赵明明有简单方案，为啥不早说，害得自己浪费了这么多时间。

但反过来，站在老赵的角度，他是怎么想的呢？“我的要求是带着业务跑，最理想的方案当然是和系统在一起，你要是能搞定，这肯定是最好的；既然你搞不定，退而求其次，自己写一个隔离出来的方案，我也能接受。”

你看出来问题在哪了吗？老赵的选择没有任何问题，问题就出在，**小李发现自己可能搞不定任务的时候，他的选择是继续闷头做，而不是把问题暴露出来，寻求帮助。**

作为一个程序员，克服技术难题是我们工作的一个重要组成部分，所以，一旦有困难我们会下意识地把自己投入进去。但这真的是最好的做法吗？并不是，**不是所有的问题，都是值得解决的技术难题。**

在工作中遇到问题，这简直是一件正常得不能再正常的事儿了，即便我们讲了各种各样的工作原则，也不可避免会在工作中遇到问题。

既然是你遇到的问题，你肯定是第一个知道问题发生的人，如果你不把问题暴露出来，别人想帮你也是有心无力的。

如果老赵不过问，结果会怎么样？必然是小李一条路跑到黑。然后，时间到了，任务没完成。

更关键的是，通常项目计划是一环套一环的，小李这边的失败，项目的后续部分都会受到影响，项目整体延期几乎是必然的。这种让人措手不及的情况，是很多项目负责人最害怕见到的。

所以，虽然单从小李的角度看，这只是个人工作习惯的事，但实际上，处于关键节点的人可能会带来项目的风险。而小李的问题被提前发现，调整的空间则会大很多。

遇到问题，最好的解决方案是尽早把问题暴露出来。其实，这个道理你并不陌生，因为你在写程序的时候，可能已经用到了。

Fail Fast

写程序有一个重要的原则叫 Fail Fast，这是什么意思呢？就是如果遇到问题，尽早报错。

举个例子，我做了一个查询服务，可以让你根据月份查询一些信息，一年有12个月，查询参数就是从1到12。

问题来了，参数校验应该在哪做呢？如果什么都不做，这个查询参数就会穿透系统，传到你的数据库上。

如果传入的参数是合法的，当然没有任何问题，这个查询会返回一个正常的结果。但如果这个参数是无意义的，比如，传一个“13”，那这个查询依然会传到数据库上。

事实上，很多不经心的系统就是这么做的，一旦系统出了什么状况，你很难判断问题的根源。

在这个极度简化的例子里，你可以一眼看出问题出在输入参数上，一旦系统稍具规模，请求来自不同的地方，这些请求最终都汇集到数据库上，识别来源的难度就会大幅度增加。尤其是系统并发起来，很难从日志中找出这个请求的来源。

你可能会说，“为了方便服务对不同数据来源进行识别，可以给每个请求加上一个唯一的请求ID吧？”

看，系统就是这么变复杂的，我经常调侃这种解决方案，就是没有困难创造困难也要上。当然，即便以后真的加上请求ID，理由也不是现在这个。

其实，要解决这个问题，做法很简单。稍微有经验的人都知道，参数校验应该放在入口的位置上，不合法的请求就不让它往后走了。这种把可能预见的失败拦在外面的做法就是 Fail Fast，有问题不可怕，让失败尽早到来。

上面这个例子很简单，我再给你举一个例子。如果配置文件缺少了一个重要参数，比如，缺少了数据库最大连接数，你打算怎么处理？很多人会选择给一个缺省值，这就不是 Fail Fast 的做法。既然是重要参数，少了就报错，这才叫 Fail Fast。

其实，Fail Fast 也有一些反直觉的味道，很多人以构建健壮系统为由，兼容了很多奇怪的问题，而不是把它暴露出来。反而会把系统中的 Bug 隐藏起来。

我们都知道，靠 debug 来定位问题是最为费时费力的一种做法。所以，别怕系统有问题，有问题就早点报出来。

顺便说一下，在前面这个例子里，透传参数还有几个额外的问题。一是会给数据库带来额外的压力，如果有人用无意义查询作为一种攻击手段，它会压垮你的数据库。再有一点，也是安全问题，一些SQL攻击，利用的就是这种无脑透传。

克服心理障碍

对我们来说，在程序中尽早暴露问题是很容易接受的。但在工作中暴露自己的问题，却是很大的挑战，因为这里还面临着一个心理问题：会不会让别人觉得自己不行。

说实话，这种担心是多余的。因为每个人的能力是强是弱，大家看得清清楚楚。只有你能把问题解决了大家才会高看你，而把问题遮盖住，并不能改善你在别人心目中的形象。

既然是问题，藏是藏不住的，就像最开始那个故事里的小李，即便他试图隐藏问题，但最后他还是不可能完成的，问题还是会出来，到那时，别人对他的评价，只会更加糟糕。

比起尽早暴露问题，还有更进一步的工作方式，那就是把自己的工作透明化，让别人尽可能多地了解自己的工作进展，了解自己的想法。

如果能做到这一点，其他人在遇到与你工作相关的事情，都会给你提供信息，帮助你把工作做得更好。当然，这种做法对人的心理挑战，比尽早暴露问题更大。

从专栏开始到现在，我们讲了这么多原则和实践，其实，大多数都是在告诉你，有事先做。

一方面，这是从软件变更成本的角度在考虑；另一方面，也是在从与人打交道的角度在考虑。

越往前做，给人留下的空间和余地越大，调整的机会也就越充足。而在最后一刻出现问题的成本实在太高，大到让人无法负担。

总结时刻

我们今天讨论了一个重要的工作原则，把事情往前做，尽早暴露问题。我们前面讲的很多内容说的都是这个原则，比如，要先确定结果，要在事前做推演等等。越早发现问题，解决的成本就越低，不仅仅是解决问题本身的成本，更多的是对团队整体计划的影响。

一方面，事前我们要通过“以终为始”和“任务分解”早点发现问题；另一方面，在做事过程中，一旦在有限时间内搞不定，尽早让其他人知道。

这个原则在写程序中的体现就是 Fail Fast，很多程序员因为没有坚持这个原则，不断妥协，造成了程序越来越复杂，团队就陷入了无尽的泥潭。

原则很简单，真正的挑战在于克服自己的心理障碍。很多人都会下意识地隐瞒问题，但请相信你的队友，大家都是聪明人，问题是藏不住的。

如果今天的内容你只记住一件事，那请记住：**事情往前做，有问题尽早暴露。**

最后，我想请你回想一下，如果遵循了这样的工作原则，你之前犯过的哪些错误是可以规避掉的呢？欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言

西西弗与卡夫卡

及早暴露问题其实是反人性的，因为意味着要向领导或众人展示自己的无知。回顾过往某个项目的时候，发现自己其实在过程中已经意识到存在问题，但因此要暂停项目、产生更多成本，就会下意识安慰自己说，其实问题没那么严重，就这么来吧，问题会解决的。现在想想，这种心态真是要不得。及早承认问题，袒露自己的不足，并积极寻找解决方案，才是成熟的标志吧

2019-03-06 00:40

作者回复

反人性，有点说重了，不理性更恰当。理性一点的话，我们就会发现，在事前做其实成本反而是低的。

2019-03-06 11:48

David Mao

在工作中经常会遇到这样的人，尤其是新人。按照自己的思路做了好久，也没把事情搞定。研发里的敏捷开发，每日展会，快速反馈，快速交付和这个有异曲同工之妙。

2019-03-06 21:31

作者回复

对，是这样的。

2019-03-06 22:17

One day

突然想起之前做过一个项目，经理把模块功能都分清楚了，把有联系功能的人分租2-5人一组。每个组都有一个或两个大牛，因为项目比较紧，项目经理说的最多的句话就是及时沟通，及时反馈，及时解决。在时间维度就有很好的把握，从项目立项之初，也是按老师递推演进的方式，计划SIT,UAT,灰度发布等时间结点按照倒推的方式进行，也给项目重要结点预留时间，最后结果是大家基本很少加班，验收没什么问题，还给我们多放几天假等等。那个时候我就觉得在这个项目经理下做事最有效率，事情安排妥当，把可能会遇到事情，做一部分预知，对于无法预知的也能让人尽早提出来，及时解决

2019-03-06 11:09

作者回复

好的项目都是类似的，糟糕的项目各有各的不幸。

2019-03-06 22:59

捞鱼的搬砖奇

做需求流程没搞清就动手做，写到出问题的地方卡住了。一直卡在谁也不告诉，知道被问起进展才知道厌恶了进度。

2019-03-06 01:32

作者回复

我隔着屏幕都能听见你的一声叹息。

2019-03-06 23:02

毅

我的做法是让程序员做之前先自述流程和思路，然后分解任务尽可能细，我可以估摸着后面可能会出问题的点，他不说我可以提前问，但有人就是细不了，那我会要求他每天提交代码，这样我的每天的事会多。还有一种是在任务分解和预估时间有的人总是过于乐观，以致我都不太相信，但又想看看这次是不是真有惊喜，有点矛盾　我想问问老师我的做法可以有啥调整改进，或者有别的建议，毕竟现状我也有些无奈，自我感觉也并不轻松～

2019-03-08 07:15

AlanP

我在带团队（研究生团队）做项目的时候经常遇到这种问题，大家总是要等到每周例会的时候才说遇到了什么什么问题，老师也反复强调了几次遇到问题要及时说，但是听的人不多，不过我已经养成了一旦发现自己暂时解决不了的问题就及时找老师讨论。但团队还是要做工作的，我想到的解决方法是用Scrum的每日站会，这样能及时了解大家的状态。

2019-03-06 09:09

作者回复

先解决自己，再用过程解决团队。

2019-03-06 23:00