从校园到职场 - 再谈切忌照本宣科

原创

caoz 于 2015-12-10 20:43:06 发布

从校园到职场 - 切勿陷入照本宣科

从CNZZ历史讲创业与打工的区别 这一篇建议也重温一下

又从知乎看到一个好玩的文章，有个实习生进入职场，说怎么巨头的技术这么这么烂啊，指出了七八个问题来。然后不多久，果然见人家技术负责人出来回复，你只看到了烂，而背后的逻辑和原因思考过没？曾经有一个年轻人也是这样以为，兴致冲冲的要去重构系统，然后造成的那段东西，还留在那里，也是你觉得很烂的一部分。

正好结合我写的关于CNZZ的历史，以及我在知乎也回答过一个类似的问题，从校园的研发到职场研发有什么区别？ 也提到了一些典型案例，类似的问题还真不小，好吧，惭愧的说，我刚毕业进入职场那时候，也会有很多不正确的认识。

今天有个全球Top5的北美名校美女童鞋问了一个问题，她说她们学一个竞价逻辑叫做GSP，教授说google的竞价策略就是用的这个东西，问我百度是不是，我一看，这个词还不太懂，看了看说明，好像是那么回事，我就说，不能说不是，但也不能说是，你看你们上课看到了这个逻辑，好像解释起来也不复杂，可是谷歌的商业广告系统至少有几十名非常资深的算法工程师为策略服务，你觉得可能说你上了一堂大学的课出来就能写出谷歌的广告计费逻辑和排名算法么？ 百度最初刚开始创业的时候，两个实习生就开始做广告系统的产品设计，最开始也是简单粗暴，可以说就是这个逻辑，但07-08升级凤巢的时候已经参照谷歌的系统，大为改进，再说就是某个教科书的典型策略，肯定是不对的，但你要说基本还是从这个策略出发的，我也不能反驳。

说这个什么意思呢？你大学里学到的所有典型的策略，算法，逻辑，在职场里，只是最基本的一些东西，甚至可能是已经被颠覆的东西，比如数据结构里强调的第三范式在高并发场景中早就被摒弃了，也就是你即便在课程里得到了满分，你来到职场，依然是菜鸟一名，因为你所面临的场景，业务环境，需求变更的频次，和你学校里完全不同。

那么是不是说你就不能发现职场公司的问题呢？

我们先说职场上所谓技术问题的由来是哪些原因造成的。

1、年代久远，菜鸟的产品

巨头也是从初创公司起步的，在起步之初，可能技术实力也不是很好，而且我们知道信息技术成长性很快，很多现在我们司空见惯的一些东西在十年前还属于无人知晓的黑科技，在这种情况下，一个持续运营的公司，多少会有一些历史上很粗糙和菜鸟的代码，并且可能部分仍在运营，这是正常的。

那么为什么会仍在运营呢？说明这个东西虽然不够好，不够正确，但是一直没有出大的问题，没有给系统添太多麻烦，所以，虽然这个东西不好，但是基于以业务为核心的诉求，企业技术部门并不是特别有解决的意愿和动力。

2、需求迭代的产物

和教科书不同，职场上的需求是瞬息万变的，一些初入职场的人会觉得这公司需求经常改，要求经常变，无法接受，觉得是不规范，不健康的表现；坦白说一点吧，15年前，乃至20年前，我们说，印度的软件业最为发达，其开发流程最为规范，当时很多中国企业去学印度，学开发规范；觉得印度在IT方面领先中国很远，但是发展到今天，在互联网大潮中，请问，哪个互联网公司还会去印度取经学习开发流程和规范？请问谁还会说印度比中国在互联网研发领域领先？ 问题在哪里呢，中国互联网产业之所以发展快就在于没有包袱，敢想敢干，随需应变，作为定制开发，强调需求确认，强调定稿，但是这种模式在互联网时代是自寻死路，你必须能随时跟着需求走，跟着时代的潮流前进，你如果按照所谓的瀑布流去做项目，你做互联网，你必然没有机会，我刚毕业的时候，也遇到这个问题，领导说，好好学软件工程，今天我可以明确的告诉大家，在互联网时代，软件工程里的所有开发思想，除了敏捷开发值得学习外，其他的都可以不用再考虑！ 如果你还认为你的开发需要明确的边界条件，明确的需求确认，明确的发展路线，请离开这个行业。

所以，问题就出来了，一个产品，最初设计的目标是A，但是开发过程中突然发现B才是真正的目标，而好不容易把开发一半的目标A的系统转为目标B，上线之后又发现需要兼容C和D的目标，所以，新人过来一看，不了解这个背景，这个迭代的历史，就会觉得，这系统谁他妈的设计的，这逻辑怎么都是拧着的？没办法，不断试错，不断调整，就是这样过来的。

3、所谓的正确的架构，存在着你所不知道的坑

就好比上面说的第三范式，新人来一看，你数据结构各种冗余，你怎么不会第三范式啊，大学课程啊，因为他不知道涉及分布，涉及负载均衡的时候，这个第三范式无法满足快速扩展的需求。

所以很多时候，教科书上的一些范例，方法，并不适应新的业务需求和应用场景，而此时，就会被只读过教科书的孩子们认为是，代码太烂，技术太逊。

4、技术演进中的印迹

一个多年运营的系统，在演进的过程中，会存在大量不同的参与者，每个参与者都会有自己的逻辑，想法，以及处理的方式，那么在代码迭代中，技术的迭代往往是优先考虑最紧迫的任务，优化最耗费资源的模块，在不断迭代中，代码的一致性和结构的一致性可能就无法维持，导致一些本来架构优美的结构可能只剩下两三个模块，新的迭代代码替换了其他模块，但是新人来一看就觉得，这模块做的好复杂，好啰嗦，很多没意义的东西塞在里面。请相信我，在优美的技术结构，经过几茬这样的迭代，都会变得面目全非，杂乱无章。然后等待下一个神来做整体的重构。

5、侧重点不同的考量问题

比如说，新人发现一个报表系统效率很低，耗时很长，于是嘲笑，连索引都不会么，这么烂的结构怎么设计的；但是他不知道的是，这个数据结构首先要满足一个非常非常巨量的每日数据新增，而后才要满足每周一次的报表生成，那么，如果按照他认为生成报表的优美结构来设计，所增加的索引和数据字段的设计，就会在日常带来大量的i/o开销，数据新增的逻辑就会导致额外几倍十几倍的开销（不夸张），也就是说需要很多台额外的服务器来支撑，而收效确是，可以每周生成报表的时候从几十分钟缩短到几秒钟，那么，如果你是老板，你会同意这样的方案么？

没有全局观，只从局部去看设计，就会自以为是的认为别人做的很烂，很垃圾。

6、资源紧缺的作品

大公司也会资源紧缺么？至少人力资源一直是紧缺的，比如有个紧急的活动，有个紧急的任务，时间特别急，程序员只好急急忙忙从第三方开源软件抓了一段代码，改吧改吧塞进去了，反正业务满足了，至于不好那也实在没时间了，结果这个临时活动效果不错，于是慢慢成为常态活动，而这个工程师又去干别的去了，这个代码反正也没出错，那就一直跑着呗，新人来一看，这啥玩意啊，代码东一榔头西一棒槌，这程序员不会设计系统么？拜托，站着说话不腰疼，真没时间设计。

那么，下面要说，新人就不能提出现有系统的问题么？就不能提出改进的方案和建议么？当然可以，你进入一个企业就是干这个的，公司花钱养你就是让你来改进系统提升业务支撑能力的。但是，这需要有一个正确的认识和思路在前面。

第一，尽可能更完整的了解系统，查看一套代码的时候，了解所谓的烂和不好的起因是什么，历史演进的过程是什么，了解其在业务系统中的地位和价值是什么，有了一些认识和理解后，再来思考所谓好和不好的问题。

第二，从一个最有把握，结构最简单的地方入手，用你认为正确的方式修改一个最小的结构，然后通过测试，发布的流程，然后通过运维和其他主管人员获得反馈，了解改进是否具备对旧版本的比较优势，以及比较优势究竟有多大，多重要。不重要没关系，但是你要了解是否这个改进是正确的；完成第一个后，努力去完成第二个，当你能够顺利发布完成十个或更多的这样的改进和优化，并确认你的代码比较优势确实明显的时候，再来谈别人的代码烂不烂的问题，如果你这些都做到了，不需要你来谈，你的主管会看得到的。

第三，轻易不要谈重构，轻易不要谈重构，轻易不要谈重构

必须说三遍

我讲过一个架构悖论，某些巨头的技术架构就出现过，为了满足所谓高扩展性的原因，开发人员做了一个特别复杂的逻辑关系特别庞杂的系统，他们说这个系统多花点时间是值得的，因为以后你有新的需求可以灵活扩展了！于是产品人员和运营耐心等待。

新系统上线后，市场发生了一些变化，新的热点出现，运营人员说，我们能不能增加一些你看那个那个，那个那个功能。技术人员说，对不起，这个需求在我们设计的初期完全没有考虑到，因为现在架构太复杂了，所以这个版本肯定不能支持，你放心，下次重构的时候我们会考虑的。

我对架构的原则是，适度灵活，简单至上，你会发现，越是简单的架构，遇到新的业务需求，改动和升级越容易，很简单的一个道理，新的工程师进来看代码理解逻辑的成本低。所谓照顾更多的需求可能性，这么说吧，人家设计了开发语言，就是照顾更多需求的可能性，你难道也要设计一款开发语言？

再说一个我一直特别想吐槽的地方，php之所以被发明并且被快速传播，成为流传最广的语言，是因为脚本语言是满足需求最快最简单的方法，门槛极低，开发测试极为简单方便，但是现在一堆人搞各种php框架，好像不做框架就不是php开发一样，我就不明白了，你想搞框架你用php干嘛啊，人家好不容易把事情弄简单了，你又咔嚓咔嚓弄回去了，我现在看着那些所谓可以灵活扩展的框架就头大，简直是莫名其妙，最后就是学习成本极大增加，调试成本极大增加，优化成本极大，极大，极大增加，面对高并发场景，优化难度比原生态开发高了不止一个数量级。 当然，这个是题外话，和今天主题无关。

回到重构话题

彻底理解业务逻辑，尽可能做到360度无死角理解各种业务逻辑之间的关系，（如果你处于巨头公司，能做到这一点几乎可以封神！）

彻底理解当前架构的演进过程（中间踩了多少坑，有多少发展中的问题都是怎么解决的，如果不理解，请相信我，你绝对会再踩一遍！）

彻底理解各个模块的调用关系和彼此的依存关系。

彻底理解当前架构的瓶颈和问题。（前提是必须知道这个架构为什么能work，否则这个理解等于不理解！）

然后再去谈重构的话题。

最后一点，强调一下，问题即机会

如果你发现一个巨头依然存在很多技术上连你都觉得不对的问题，你应该高兴才对，这不是你的机会么？但是你要明白，哪些是真的问题，哪些是你涉世不深的误会，你一步步解决这些问题，在解决过程中一步步理解系统，到一定程度，你会明白更多，理解更深；只是站出来喊，这些垃圾那些垃圾，只能暴露你的浅薄和无知。又要重复那句可能掉友善度的话，别做教科书般的SB。