之前公司（UMF）存在的问题

2022-03-11

2021.08.31 原文

我之前任职的公司是传统类型的电信服务类企业，话费支付起家。区块链部门是创新业务部门，内部孵化项目和产品，属于成本部门，公司给钱，但也有一些收入预算。部门收入的来源主要是：

- 政府补贴
- 招投标项目
- 熟人介绍项目

都是交付形式的项目，招投标项目是按照指定要求开发软件然后全权交付出去，熟人介绍项目是把内部开发的区块链产品交付出去，都要提供开发联调和售后支持。如果没有项目中标、没有熟人介绍，就没有收入了，存在一定收入压力。

这种营利方式属于相对好理解的模式。很早以前我还没有了解部门收入来源的时候，就无意间和同事说，“现在已经不是靠卖软件挣钱的时代了”，暗指部门领导的挣钱思路有问题。同事问，那靠什么？我随口说，“卖服务啊”。我当时并没有关注相关问题，只是闲聊而已。

想到这个话题是因为，今天问到 HR 公司盈利模式的问题，“我们又不是像卖煎饼一样，卖一个挣一个的钱。为什么互联网公司的工资高？靠的不是直接营利的收入，是投资人砸钱，花投资人的钱。互联网公司的钱都不是真实的钱，很多互联网公司，为什么财报亏损但还是很多投资人愿意投资？他们都是先占有市场和用户，然后上市，最后由股民掏钱了。我们也是有 IPO 的计划……”

当然，对这些内容的真实性和准确性存疑。但能肯定的，和之前公司的销售模式完全不同。

HR 和我提到说，有什么问题可以随时问，我们的企业文化就是 “坦诚沟通”。我说，可能直接问只会得到一些「道貌岸然」的回答，有些事需要不断「试探」才能知道真相。我想到，我这样的行为方式，可能和之前的经历有关。

如果你提出一个现实的具体的问题，一个人在回答的时候说了很多，扯得很远，某银行的金融方式什么，某大学的教授说了什么话，某体制内机构的高官是什么样的经历，公司某高层怎么要求，公司某员工有什么样的行为，名人名言、历史典故，等等。说了一圈，你发现他并没有正面回答你的问题，好像回答了，又好像没有。偶尔一两次你会觉得，这个人水平很高，接触的都是一些高级的人，说出来的都是大道理方法论。

时间长了以后，你发现，每一次具体的实际的问题，他都会用方法论来回答，顾左而言他，就是不正面回答你。甚至很多次说出来的都是相同的事例不同角度的道理，从具体问题总结出方法论是一种高级的技能，但如果一味讲究方法论，也会是一种灾难。所有问题一说出来啥都懂啥都明白，怎么回事别人是怎么做的我们应该怎么做，但就是解决不了现实的问题。后来你也许会觉得，可能他不是不想回答，是他也不知道问题的答案。有人说，其实他也很迷茫。

在具体的工作内容上，他唯一的要求就是 “客户认可”，也就是能赚钱的、最好有直接收益的。你可以把产品相关的技术方向全部列出来，但是发现在所有的技术方向上，没有一个是所谓 “客户认可” 的，甚至全国有吗？是什么？客户怎么会因为你使用了某一种技术而买账呢。你明知道项目存在各种各样的问题，知道该怎么改善那些问题，但是在 “客户认可” 的判断依据下，这些问题没有一个是需要解决的，没有一个产品的技术方向是可以深入的。

“客户认可” 绝对是再正确不过的判断标准，公司做什么事情一定是为了营利，付出的成本要体现在收入上无可厚非。但是，但就是感觉这玩意儿很难啊。公司本身是那样的营利模式，但是在那种营利模式下，国内挣钱的都没有几家。

尤其是看到国内某开源项目如火如荼势如破竹，部门的产品寻找突破口更加困难。在近几年都没

有显著创新的情况下，没有技术储备、没有发展路线、没有商业资源，甚至工程质量都一言难尽，当你说产品的竞争力不足，又会听到说“其实我们公司不是专门做这个的”，公司的高层几度放弃部门和产品，这些都预示着部门在走向灭亡。

2022.03.11 更新

最近注意到服务集成这个软件开发的领域，以及在这个方向上有所成就的创业公司，同时看到有人来公司面试，偶然想起一些以前的事情，回想起我在上家公司的经历。之前文中没有具体说的是，部门的市场定位不清晰。当然造成那样的结果，有很多历史原因，公司一开始是有靠山的子公司，作为甲方立场很足，后来转变为乙方的形式，没有活跃在市场上的基因。这里只说结果。

如果部门的定位是拥有技术基础去自研产品，那么问题就很严重。

我在正式接手核心产品的开发后，发现项目的代码在工程上有凌乱不堪，不是代码乱，是没有顶层设计，没有明确的模块划分，没有清晰的目录结构，没有靠谱的软件设计。项目支持四五种数据库，然而数据库的配置方式竟然不统一，分散在不同配置文件的不同位置，哪儿生效哪儿不生效，哪个功能好使哪个功能不好使，要么靠经验，要么靠猜。API 的设计也很糟糕，我在《随想》中提到过关于 URL 参数的问题，另外 URL 的配置和参数的校验是写在单个的配置文件中，意味着如果在智能合约中想新增 URL，就需要改配置文件然后重启节点。至于配置规则的热加载，好像没有人关心。

智能合约的机制也有问题。交易在提交到合约后，交易的检查和执行分为两个步骤，检查函数的入参和出参都是 bitMap，必须要严格保证入参和出参的长度一致，否则节点就会 panic，因为外面是用循环处理的，会 out of index。问题在于，那可是智能合约，怎么会用那么生硬的写法。后来前面的人告诉我，写智能合约的原则就是，“绝对不能出错”，因为说是智能合约，其实是和项目耦合很深的功能模块，美其名曰系统合约，是底层链的开发人员去开发的，而不是交给用户使用。开发合约，就需要对底层链有足够的了解。当时的人似乎还对这种事情有一点自豪感，感觉像是，“我们能写，因为我们比较熟悉”，丝毫不认为那是一种功能的不健全，而认为是有门槛的 feature。听说本来是没有打算支持智能合约的，由于需要的不断扩张，就硬生生加上了。当时的某人还拿 leader 的某篇文章奉为真理，说，其实区块链也不一定需要智能合约，对那种和广义智能合约理念背道而驰的设计大加赞赏。

多写几个合约后，就会注意到每个合约的检查函数上，都会有一个判断交易是否为空指针的语句。本以为系统内部的函数调用，怎么会凭空出现空交易呢，经过复现和排查，发现在并发情况下，队列偶尔会出现异常，push 一批交易进去，pop 出来就有空交易了，这纯粹是数据结构的问题。然后虽然定位到问题，但没有去解决，因为大不了每次都在合约上写个判断，算是我偷懒。也不知道前面的人，还埋了多少隐性 bug 在里面。

当用合约处理业务的时候，就会发现数据库的读写性能会成为交易性能的瓶颈，比如 MySQL，而且存在一个我一直没想通的问题：合约里的检查函数，怎么判断交易是成功还是失败？因为合约是要针对业务去开发的。合约对交易的检查和执行，都是是在 BFT 共识的 commit 阶段，这个时候已经完成共识了，检查函数并不能去预执行数据库的写操作（如果合约依赖数据库的特性比如事务，区块链就没有意义了），难道要把所有有可能失败的场景全排除一遍？是语义层面的排除，还是执行层面的排除？即使能够枚举出异常，又会损耗多少性能？那可能会产生疑问，为什么不在共识前检查？共识前的检查也是有的，但不管在共识前还是共识后，对数据库的操作总量不会变，对性能的损耗不会变。（延伸思考：为什么公有链不存在这样的问题，联盟链存在。）

版本管理的混乱也是在工程方面的问题之一，甚至都没有人能说清楚，当前的版本号到底是多少，是 2.0 吗？配置文件里可还写的是 1.4，是 2.0.1 吗？仓库里可还有 2.0.3，但不知道是谁改的，有哪些变动。甚至主干代码中会出现用于测试的 case，有些需要异常场景测试的情况，比如在 BFT 共识过程中恶意投票，只能通过改代码的方式观察效果，结果那部分代码就保留在了项目中，可以通过配置启用。其他原因的代码冗余也存在，比如智能合约的公用接口，为了兼容 UTXO，就不得不增加对应的接口，但其他合约完全用不到那些，又不得不实现。

项目在技术方面是存在各种各样问题的，包括我之前解决的由于使用 VRF 导致共识在提案阶段黑名单失效的问题，都说明系统尤其是在比较核心的地方都不够完善，而重构项目的成本又非常高。项目存在的最大价值和意义，就是参加一个行业内知名的测试活动，测试通过后某机构会给企业颁发证书，以证明这个软件是合格的、有资质的、符合标准的。然后企业拿着这个证书就可

以宣传、投标、卖钱。至于软件本身好不好，并不重要。我当时光看测试项还感觉没啥，都是一些对区块链的基本要求，能通过测试没什么大不了的。直到亲手操作后发现，测试过程中存在大量的困难，都是人为困难，由于之前开发人员的不专业、团队管理的松散，以及项目本身很多不合理的设计，加上文档和人员的流失，都大大增加了测试过程的准备难度。也许大家没有意识到，什么是有效的困难，才造成了一种项目很好的假象。

如果部门的定位是服务集成，提供解决方案和技术支持，也是不合格的。

除了区块链底层，部门还有 Bass、中间件、SDK、浏览器之类的项目，会涉及到 Kafka、Zoopeeper、Redis、普罗米修斯等组件，但用的很浅，本身技术含量低，整体上做的也不到位，没有产品、没有 UI、没有用户思维、没有 owner 意识，全是在有业务需求的情况下临时改进，结果每次都手忙脚乱，总是以优先应付客户为要务。服务集成的事情，可以简单也可以难，可以做好也容易做不好。然而部门另一方面又不太想做服务集成，比如在用 Hyperledger Fabric 做一些什么事情的时候，leader 就说：客户会问，你们底层用的 Fabric，只是拿来用，也没干什么呀，为什么要收钱？至少能体现出部门的定位有多混乱了。