## 基础软件，未来只有开源一条路？

### 代码不是核心竞争力，那什么才是？

我们看看一个闭源的软件的一生：立项的动机可能是某个公司或者个人对于一个市场机会的洞见找到了一个高价值的场景，通过开发一个软件能够很好的提高效率或创造价值，甚至可能就是一张来自甲方的合同，总之这个公司招募了一伙程序员，设计师，产品经理，开始项目的开发。一切顺利的情况，顺利的满足了甲方的需求，甲方也很开心的付钱了，然后这个公司发现，好像这个软件改一改也就能够在同行业另一个客户那边卖出去，这就太好了，感觉找到了一条致富路。可是好境不长，客户这边的场景和需求在变化，原来的软件可能不一定能够满足新的需求了，但是开发团队就这几杆枪，稍有不慎一个方向判断错误，可能时间和机会窗口就错过了。这就意味着，对于项目领头人的要求就很高，要求持续能够引领行业的方向。对于有些有着研发能力的甲方，因为受限于没有源码，就算知道如何改进，也只能干瞪眼。

其实这个问题的本质在于：闭源软件开发商虽然可能是技术的专家，但是并不一定是业务或者场景的专家，软件进化的速度受限于开发团队和产品经理自己的认知和见识的进化速度，除非开发商强大到能够持续引领整个行业的进化方向，否则无解。

其实这个问题，教员早就给出了答案：「**…凡属正确的领导，必须是从群众中来，到群众中去。这就是说，将群众的意见（分散的无系统的意见）集中起来（经过研究，化为集中的系统的意见），又到群众中去作宣传解释，化为群众的意见，使群众坚持下去，见之于行动，并在群众行动中考验这些意见是否正确。然后再从群众中集中起来，再到群众中坚持下去，如此无限循环，一次比一次地更正确、更生动、更丰富…**」

— 《关于领导方法的若干问题 32》, 1943

第一点，**开源软件的生命力来自于场景的垄断，而背后更本质的垄断是人才垄断。**

为什么强调从群众中来？**软件的初始动机虽然来自于少数人的洞见，但是持续保持洞见并不是一件容易的事情，这就是为什么很多技术团队或者产品团队容易「自嗨」，一旦脱离用户，极易出现这样的问题**。闭源软件厂商触及用户的手段不外乎于传统的商业宣传和销售，用户从感兴趣到使用起来的门槛很高，实施的周期也很长，另外通常销售会站在产品团队和客户中间，通过一些信息不对称来获取超额的利润，其中最大的信息不对称就是封闭的源代码本身或者定制化。这导致的问题是，相比流行的开源软件，闭源软件没有办法高效的获取，吸收和理解更多的场景，这对于软件来说通常是一个致命的问题，如果见过的场景不够多，更没有办法判断产品那些需求该做是普遍需求，哪些是伪需求坚决不做，我认为这就是做产品的「触感」。

对于一个流行的开源软件，本身不会有上面提到的问题：因为有足够多的用户，那么一定能看到足够多的场景，也能看到足够多的稀奇古怪的用法，这一个个用户的反馈，修过的一个个 bug，提出的一个个建议，会持续的产生类似「复利」的效果，**你的软件越强壮，见过的场景越广，会进一步让你接触到更大的用户群，帮助软件变得更强大，如此循环。实际上开源软件本质上是通过放弃一部分通过信息不对称产生的潜在利润，换取了极其高效的传播和场景触及效率**，但是有意思的是，实际上牺牲掉的这些潜在利润大概率也不一定真的牺牲掉，一来可能本身付费能力有限，二来可能实际上这些用户通过宣传站台二次传播或者代码贡献等方式回馈了项目本身。

在上面那个过程中还会产生一个更加厉害的效应：人才的垄断。正所谓「事在人为」，上面提到的场景垄断中种种的技术决策和实践都是人来操作的。一个流行的开源软件在变成事实标准的过程中，一定会培养出大量熟悉这个产品的工程师，用户，摇旗呐喊的粉丝，代码贡献者，甚至挑刺吐槽的人。传统意义上，大家理解的开源社区只是狭义上的开发者社区，只有贡献代码才算参与，但是我认为只要和这个产品发生关联的人，都算是社区的一部分，「人尽其材」才是构建开源社区的终极目标。这个优势是会随着时间的流逝不断累积。

第二点，**迭代，迭代，迭代，只有高速迭代才能立于不败之地。**

上面教员的话里面有个关键的点，关于正向循环，也就是迭代。这个道理同样也适用于软件开发，软件从来都不是静止的，随着市场和竞争环境的变化，你今天的竞争优势，很可能明天就不是了。很多人都喜欢用静态的眼光看待问题，热衷于各种方案的横向对比，而忽略了进化速度，在这点上，我可能更看重的是同一个产品的纵向对比，举个例子：目前有 A, B, C三个方案，可能当下看这三个方案差距不大，也许在百分之五十之内。但是如果其中一个开源方案每次和自己半年前比都是在翻倍的提升（背后开源社区推动），但是闭源的方案的进步受限于团队规模和资源。这时候的选择除非是那种火烧眉毛的情况，否则一定应该选择一个迭代速度更快，增长率更好，更代表未来的方案，这个也很好理解。这是人的思维的一个惯性，人总是倾向用线性思维去看待问题，于是对非线性增长的事物往往会习惯性的低估。

### 如何挣钱？未来在云端

刚才我们聊了很多产品哲学上的东西，我们接下来聊聊商业，以及在云时代开源软件的位置。让我们回到开篇提到的那个话题：担心用户拿到代码，就不给钱了。这个观点背后的一个暗示是，用户付费买的是代码，如果有代码，用户就没有其他理由付钱。其实这个结论是靠不住的，客户付费买的是解决问题和创造价值，而不是代码，如果拿到你的代码自己折腾付出的成本大于给你的钱（如果你能如实交付价值的话），用户没有任何理由不付钱。而且这里的成本包括，比较明显的成本，例如人力成本，机器成本。也包括一些经常被人忽略的成本，例如错失市场机会的沉没成本，业务改造迁移成本，学习成本，线上出问题没人懂修带来的风险成本，这些隐性的成本往往是比显性的成本高得多的。

上面我的解释中暗示了一点：软件的价值取决于它解决了什么问题，创造了什么价值，而不是开源与否。举个例子：一个分布式关系性数据库，一定比一个分布式缓存更加有商业价值，这是由前者的应用场景，存储的数据以及提供的能力决定的，而不是开源与否。所以这就是为什么我们要做通用数据库的核心原因，因为价值天花板更高。

还有一点需要强调的，开源并不是一个商业模式，而是一种更好的软件开发和分发模式。另外，我认为商业模式和软件本身一样，也是需要设计的，这个设计取决于产品特性和公司的属性，这就意味着适用于 A 产品的商业模式，不一定适用于 B，甚至同一个产品，不同的公司，可能适合的商业模式都是不一样的。

我认为如果想要通过软件创造可规模化的持续利润，需要两个关键点：

* 生态，软件能形成生态或者和现有生态有机整合，由生态补齐单一产品的能力，从而才能进一步能形成解决方案。
* 渠道，高效的分发渠道和支持渠道，这确保在用户规模化后，作为厂商的销售和售后成本不会随着客户的增长而增长（至少成本增长的斜率需要更缓）。

两者缺一不可。对于第一点，开源软件构建生态是很天然的，开发者和解决方案提供商会很自然的通过不同开源软件的组合做到解决方案的覆盖，这个效率是闭源定制化软件很难跟上的，这点不赘述。

第二点，其实理想的渠道就是云。云标准化了硬件，标准化了计算力，甚至标准化了计算力的交付方式，尤其是公有云。一切都是标准化的好处就是可以自动化，这个对于软件供应商来说才是真正的价值。

所以开源 + 云的模式，在开源这端，完成了开发者的心智占领和解决方案的成型，然后在云这端完成极其高效的分发和价值传递。看上去很美不是吗？理论上确实没问题，但是一定会有朋友挑战我说：这个模式里面没有你们开源软件厂商什么事情啊？云为什么不自己提供开源软件服务？这几年沸沸扬扬的 AWS 吸血事件逼得一堆开源公司和项目改协议，就是一个例子。

关于这个问题，我的看法可能和主流观点有点不一样：

Cloud is eating Open-source? No, Open-source is eating the cloud.

云厂商就像当年的运营商一样，占据着和客户对接的第一位置，当然很自然的在关键路径上放自家的产品。但是移动梦网和飞信的故事后来大家都看到了，拿飞信做一个例子，大家还记得作为移动的飞信，当年是没有办法和联通电信的手机号码互通的，直到后来微信的出现，终于事实上打通了各个运营商，所以市场格局就出现了很明显的分水岭，运营商是谁不重要，只要保证网络通信号好就行。对于云也是一样，AWS 肯定不会为 GCP 提供舒服的迁移和打通方案，反过来也不可能，但是对于客户来说，这个选择就像逼着用户选移动的飞信还是联通的沃友一样，用户肯定说：不好意思，两个都不要，我选微信。从另一方面来说，对于在云上提供开源软件服务这件事情，云厂商本身的投入其实不一定有这个开源项目背后的公司多，一个很好的例子是 Databricks 是 Spark 创始团队的公司，也是一个 100% 在 AWS 上提供 Spark 服务的公司，相比起 AWS 的官方 EMR，Databricks 完全不占下风甚至客户和产品都胜过原生的 EMR。就像飞信的开发团队的质量肯定没有微信高，是一样的。

由于开源软件的中立性，使得开源软件成为用户在多个云厂商之间保持统一体验和统一服务的几乎唯一选项。因为开源软件和开源服务商的存在，市场我相信会进入一个平衡：云厂商会持续优化它擅长的东西，真正的将云基础能力变成水电煤一样的规模化生意，开源软件厂商基于云的标准基础设施构建服务并交付业务价值，开源软件项目和社区由于厂商的持续持续，不断的蓬勃发展，占领更多用户的心智。三者形成一个价值链的闭环。不要着急，让子弹飞一会儿。