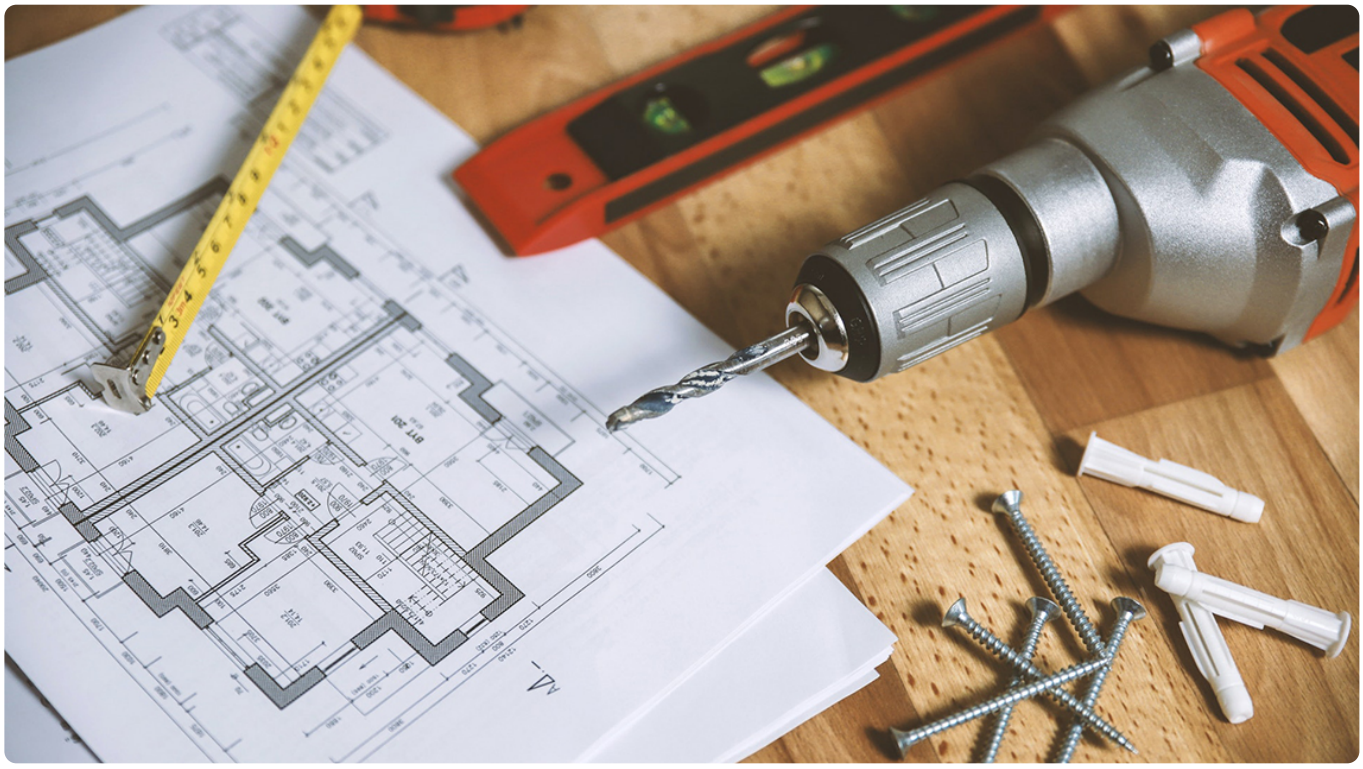
18 | 组织管理：如何突破团队效率提升的三大关？

2020-09-30 许健

技术管理案例课 进入课程 >



讲述：许健
时长 19:21 大小 17.73M

你好，我是许健。今天我们来谈谈如何提高团队效率。

关于提高团队效率，我先和你分享一个故事吧。这次新冠疫情期间，我回顾了自己这八年做云计算的经历，我觉得我们的团队做得非常累，我们团队的工作时长在 eBay 中国研发中心一定是排在前面的，但是我们的口碑却不一定最好。

为什么这么说呢？因为同僚和领导承认我们很辛苦，却不觉得我们很优秀，从客户满意度、工程卓越性来说，我们取得的成效都和我们的付出不成正比。

为了提高效率，我们也多次抓过可靠性、代码质量、搞 CICD，连执行组织形式也尝试过多次变更，但是我觉得做这些事情都不是最最关键的。事实上，当我们去总经理那里汇报组织提效方案时，得到的反馈是“**没有触及灵魂**”。

这个反馈对我的触动还是蛮大的，我开始认真地反思组织没效率，问题到底出现在哪里？

我这个人很喜欢从历史里汲取教训，于是就把我们这三代云计算系统的经历排了一遍，分析哪些事情做得好，哪些事情做得差，然后对比它们之间的区别，最后总结了提高组织效能的三个关键问题。接下来我们就来详细聊一聊。

1. 选择正确的事情

踏上管理岗位以后，你有没有意识到你的一个决定下去，会直接影响少则几个人、多则几十个人，少则几个月、多则半年甚至数年的投入呢？

很多时候，做正确的事情，远比正确地做事更重要。做经理的，想要提升团队效率，首先就要看清楚我们身在何处，又想去向何方。

那怎么定义什么才是正确的事呢？我给你分享一个思路，就是做决定时反问一下，如果这是你自己拥有的公司，员工的工资都是你自己掏的，你还会这么决定吗？

我回顾云计算这些年在 eBay 的变迁，我们不是做得太少了，而是做得太多了。我们有的项目是技术驱动然后绑架客户来用，而不是真的客户需要，接下来我给你分享三个具体的故事来说明这一点。

第一件事儿，就是我们把内部的云计算 CICD 从 Jenkins 改成了 Prow，这件事儿投入了我们很大的精力。我认为影响云计算团队交付效率的关键在于测试环境不够稳定，边界测试和性能测试的测试框架不够健全。我们总可以列出 Prow 优于 Jenkins 的点，但这是解决团队交付效率问题的核心吗？

第二件事，我们启动了 Account Resource Quota，我承认公有云系统的资源都是属于 Account 的，我们也列了很好的交付目标，比如可以让每一个业务部门自己管理自己的预算，从而帮助 Capacity 团队提高管理效率。

但后来 Capacity 团队的负责人却告诉我，他们最大的痛点不是缺乏按账户管理，而是如何在保证资源利用率的前提下，加快业务部门获取资源的速度。也就是说如果财务模型不转换成每个部门单独结算的话，就算我们把系统做成按每个部门的账户结算，也不会提高资源利用效率。看来单单从 Account 角度变更 Capacity 的管理方式，并没有解决用户真正的痛点。

第三件事，我们做了好几代网络管理系统了。在这个过程中，模型驱动是不停在强调的一个标准，模型驱动其实没有问题，但问题是花了这么大力气做系统变更以后，长期困扰我们的难点（比如安全流量迁移、网段分配冲突和泄漏）并没有得到解决。而且我们在搞出新系统的过程中，并没有干掉上一代系统，甚至更上一代的系统现在还在生产环境运行。

从刚才讲的三件事儿中，你可以看得出来，我们只是在积极地做事儿，但并没有选择做正确的事情，这些事儿的共性就是**投资收益比不高**，却白白花费了团队很多精力。所以，在我们做决策以前，必须先理清做什么才是正确的事儿。

那什么叫正确的事情呢？**我认为正确的事情就是在做决策的那个时刻，管理者所能选择的可以最大化交付业务价值的事情。并且这个业务价值不是管理者主观认定的业务价值，而应该是客户认可的业务价值。**

其实我自己整理了一个文档帮助我理清思路，文档中的例子还不止上面这些，也不仅仅限于云计算部门。这里的关键点是第一出发点的选择问题，就是说技术经理要从客户认可的最终业务价值考虑，而不是把技术先进性当作第一出发点。只有从最终的业务价值出发，我们后面的努力才有意义，组织效率才能真正提高。

凡是可以从根本上提高组织效率的事情都不简单，那么我们想干掉那些“不正确的事情”，难点在哪里呢？

第一个难点在于凡是有能力在组织内提出新项目，甚至有能力组织一部分员工做雏形系统的人，一般都是组织内能力较强的人。如果技术经理经过评估后要关停这些人的项目，抽走支撑这些项目的资源，很大可能会让这些骨干很不爽，那我们有没有这个感情强度和能力落实呢？

第二个难点是也不乏有些项目就是我们技术经理自己启动的。我们有这个气量来承认自己之前错了，然后纠正自己的错误，而不是不停去找理由证明自己是对的吗？

难点列出来了，我们要怎么解决呢？虽然有些一言难尽，但这里的本质问题就是做好冲突管理。我们要在组织内部统一思想和认识，有魄力“下刀”。因为我们一旦确认了某些事情不是我们要选择的方向，那再做这些事情不仅毫无意义，而且还会浪费企业的资源，要知道在错误的路上走得快还不如在原地不动，所以我们必须删除这些项目。

2. 选择合适的技术方案

前面说的留下高收益比的项目，是决定了我们到底做什么，那么选出合适的技术方案，就决定了我们怎么做。

不打移动靶

我先说说方案选择的关键原则，**技术方案的选择请务必直指核心问题的解决，不要打移动靶，不要去追求技术的纯粹性导致不断扩大战局，最终造成投入成本的快速增长**。

为了让你理解这一点，我就拿 C3（基于 Openstack 的云计算平台）到 Tess（基于 Kubernetes 的云计算平台）的迁移为例做个说明。

假设 C3 环境下我们要创建一个带有 100 个虚拟机的应用，那么就要先准备 100 个虚拟机，然后一个批量操作把负载均衡器配置好，后续部署代码的时候重用这 100 个虚拟机。也就是说，多次代码部署的时候不用重建虚拟机或更改 IP。

可是迁移到 Tess 以后，每一个 POD 创建完成都会触发一次负载均衡操作（加 LB Member），Tess 不是 Fire-and-Forget（发后即忘）模式，而是不停地进行 Reconcile。Kubernetes Native 每次部署时都会重新创建所有 POD，而在胖容器环境更换 Image 也是需要重建全部 POD 的。在这个过程中，其实 API 的调用次数是明显高于原来的 C3 环境的。

现实情况是，并非整个生态都已经在 Tess 上，我们还有很多外围系统。所以我们耗费了大量时间试图解决性能问题。我给你说说当时我们的尝试过程：

> 第一回合，首先我们把 Tess 调用 LBMS（Load Balancer Management Service）的方式改成了 Bulk Call，并且让 LBMS 去除了多余的输入有效性检验来换取性能，但还是不行，于是我们又联系数据中心添置额外的 LB 硬件设备来分摊调用量。
>
> 第二回合，硬件扛住了，可是我们的配置管理系统 CMS Sync 在高压下还是会出现数据不一致问题，改了好几版这些才解决掉这个问题。
>
> 第三回合，解决了数据不一致，我们又发现重建 POD 后的胖容器还需要重新部署代码，于是 CMPAAS（eBay 的代码部署工具）Schedule 性能问题就因为不堪重负而暴露出来了……

我们本来只是上一个新系统，结果变成了要改造整个生态。在这个过程中我们的实施成本成倍增长，最后的交付时间一拖再拖。回过头来看，**我们是不是需要思考一下，当引入一个新的系统的时候，到底最看中的是什么？**

我们看中 Docker 的“Build Once，Run Anywhere”（一次编译，随处运行），看中 Kubernetes 的 Spec Driven（规范驱动），而在这个例子里，Rolling Upgrade 是否需要坚持 POD 和对应的 IP 重建值得商榷。

很巧合的是类似事件屡见不鲜，所以我们一定要提高警惕。最近我还在跟总部一位同事讨论，我们一个项目的核心只是为了给 Squid 的 Proxy 加上 ACL， 但是为什么谈着谈着，就变成了要把整个 Squid 集群换成 Envoy 集群呢？这么多年来这样的事情发生得太多了，其模式如下：

一开始，我们要解决问题 A，大家都认同 A 是值得解决的；接下来，解决问题时我们偏向新技术，觉得能搞定新技术，结果在过程中还想顺带解决别的问题，而且搞定新技术的时间超过预期。再然后业务突然有需求，新技术栈还没有好，只能让老技术栈来扛，这时人手已调往新技术；最后总是祸不单行，老技术栈扛不住业务突发需求，拖死。

所以我给你简单总结一下，做技术经理的一定要时刻提醒自己，我一开始启动这个项目的初衷和想解决的问题是什么，我够不够专注，特别是在项目推进中碰到周围干扰时我有没有坚持足够专注？有没有把一开始想要解决的问题**踏踏实实地解决彻底。**

突破关键瓶颈

刚才我给你讲了方案选择的关键原则——不打移动靶，专注于一开始要解决的问题。但是除了这个原则，我们还需要解决关键瓶颈怎么突破的问题。要知道，决定整个战役成败的，往往就是那一两场关键战斗。

我先和你分享一个故事吧，我们的监控组交付 Metrics 耗时了四年，记得对这件事做复盘的时候，副总觉得最最关键的问题是团队不够专注，所以他决定停掉监控组所有的项目，强迫监控组只专注 Metrics 这一件事上。

但我跟副总说，对于监控的复盘我有不同的看法，关键瓶颈没有突破，就算停掉所有的工作专注 Metrics 还是不行的。我为什么提出这样的看法呢？我们先看看监控组在 Metrics

上的历程：

第一版是基于 Storm 来实现的流式处理引擎。

第二版我们发现眼下 Flink 才是趋势并且觉得 Flink 有很多优点，但是因为改造成本过大，于是选择了 Storm On Flink 的方式。

第三版又有变化，因为第二版走到后来发现 Storm On Flink 有很多限制，于是决定走 Native Flink 模式。

第四版，这时美国的一位资深架构师 M 指出公司内已经有很多部门在使用 Prometheus，我们也意识到我们基于 Flink 的实施方案有问题。因为这个方案需要自主开发处理各类时序数据的函数，但我们没有足够的投入可以去开发这么多各式各样的函数。于是开始转为解决 Prometheus 的高扩展下的性能问题。

这四版的历程我刚才给你交代时只是简短的几句话，但实际过程都是我们团队以年为单位计数的成本投入，直到第四版的方向确立后，架构师 M 亲自实施了 Prometheus 的扩展原型，性能调优落实到 Prometheus 内部实现，最终论证了可行性。

这个关键技术瓶颈解决后，监控组半年就交付了可投入生产环境的成熟时序数据监控方案。其实整个监控 Metrics 的交付，核心问题就是高可扩展性下的性能问题，整个团队前期耗时三年半却没有交付，但最后半年就交付了的根本原因是什么？在我看来就是一个高水平技术人员在关键点做了突破。

后来又是这位架构师，确立了使用 ClickHouse 来构建我们的下一代 Events 监控方案，可扩展性和性能的问题也一并解决了。

类似的事情还有很多，这些都让我深深意识到从事基础架构工作中要去找关键瓶颈。这类难题只靠堆更多的人是不行的，就是需要高手，要么外面引进，要么内部有合适的人能攻坚。

我一直强调人和事的并行，我们找了高手，也总得搞清楚关键瓶颈在哪里吧，那关键瓶颈到底怎么找到呢？我一般会用这两个问题帮助自己整理定位关键瓶颈：

我们的着眼点是不是足够聚焦？不停逼问自己哪一个点突破可以极大提升产品竞争力。注意就只挑一个点。

问题真的是关键瓶颈么？关键瓶颈一定不能轻易解决。要么是技术难度极大，要么是关系很复杂，要么要耗时很久……

这个思路怎么落地呢？我们还是用一个实际案例来说明，比如我们监控目前在做**异常检测平台**，需要解决的问题看起来有这 3 个：

1. 根据当前选定的一两个业务的实际生产环境，找到一个可以符合**性能和精度要求**的算法。

2. 算法精度所依赖的底层数据质量不过关，所以需要增强算法的**鲁棒性**。

3. 找到一种可以**快速自适应不同场景**，并能保证一定精度的算法。

这 3 个问题第一眼看上去都很重要，但是如果我强制说一定要排一个优先级，并且推断出最高的优先级，就会迫使我们进一步分析筛选。

我们的目标是构建一个平台，产品的竞争力到底在哪里呢？就是异常检测问题解决的投入产出比。也就是说，我们选择突破的点一定是能够让大批客户上线试用的，问题 1 能够让一两个用户上线但是无法实现大批客户上线，这不能让我们的异常检测成为平台去服务很多人，也就是解决方案的覆盖面不够广。

问题 2 单独看着挺重要的，但如果和 3 对比一下，我们就能找到不足了。问题 2 其实是问题 3 的必要条件而不是充分条件，因为即使解决了 2 还是不能达到大批用户可以上线试用的平台要求。所以最终我们确定了关键瓶颈是 3，因为这是将 eBay 的大量对异常检测有需求的场景，进行平台化解决的关键。

3. 如何激励好组织内的员工

确定了团队做什么和怎么做，既然是经理，最终还是要回到人这个话题上来。刚才在突破关键瓶颈的问题上，我也强调了高级别人才的关键作用，那我们经理要怎么激励他们呢？接下来，我结合自己的感受给你说一说：

在相当一段时间内，eBay 中国研发中心都很忌讳讲 Ownership 这个词，我们只强调 Responsibility。原因是美国有些领导觉得中国动不动就要跟他们谈 Ownership，他们感觉这就是要抢活，没有 One Team Mindset。

我最近对这件事有了新的看法，我跟总部的副总和诸多领导都直说了，我觉得只谈责任不谈权力不谈担当是不符合人性的，而且我不认为 Ownership 和 One Team Mindset 有什么冲突。

以我自己来说，副总最近让我全权负责云计算产品的入口体验，我对这个事情的投入程度和你让我辅助别的领导来做就是不一样的，我不是说我辅助别人就不卖力了，但是卖力程度可以不一样，我花 100% 的力气你也说不了我什么，问题上你怎么能让我花 120% 甚至 200% 的力气呢？

其实答案很简单，**信任和授权**。给高级别员工授权让他们去独立负责一个大项目，给他们自由让他们按照他们的方法去实现目标，用经理的信任去换他们的承诺。

对于部门里我们看好的有潜力的员工，要敢于给机会，要高标准要求，出了问题我们也要兜着，因为对于经理来说，这些潜力股未来的成长更重要。

最后，如果他们真的高质量达到了高标准，不要吝惜奖励，并且要以超出常规的方式去奖励他们。我对比我们部门和数据基础架构部门新人培养速度的差异，为什么他们不断地有明星员工浮现出来？我觉得关键的点就在这里：**我对我们部门有潜力的员工的要求不够高，并且在奖励上不够刺激。**

最后就是淘汰部门内业绩差潜力差的员工。具体的操作方式我在裁人那一篇谈过了。心要慈，刀要快，有些事我们不喜欢做，但是为了这个组织能够有更好的发展，就是需要去做这些不开心的事情。而且级别越高的经理，最后留给我们去裁的人越难办。

总之，能者上庸者下，为了团队效率的提高，员工激励这件事的原则就是：**赋能有潜力的人才和淘汰业绩差的员工。**

总结

组织管理上我们可以定一个基调，所有能从根本上提高组织效率的事情，都一定是高成本、高难度的。

一招鲜吃遍天的绝技不存在，天上掉馅饼的好事更不会存在。“高光时刻”的背后，更多的是在整个过程中无数个平凡的日日夜夜的坚持，在我们成功之前，也要做好没有多少鼓励和关注的心理准备。

在提高组织效率的路上，我们有三大关卡要突破：选择做正确的事情、确定合适的技术方案以及激励好员工。

首先，**正确的事情就是在做决策的那个时刻，管理者所能选择的可以最大化交付业务价值的事情。**要注意，这个业务价值不是管理者主观认定的，而应该是客户认可的。想要干掉“不正确的事儿”，要么会动骨干的蛋糕，要么就是纠正自己的错误，本质上是我们做冲突管理，需要有足够的魄力去落刀。

接下来，决定了做什么后，在具体实施过程中要不停提醒自己目标是什么，然后不停地质问自己，我们的技术方案是否始终聚焦在最开始的那个问题上。不要打移动靶，而是要聚焦。尽量减少依赖，不要轻易扩大问题范围，总之就是减少变量数目。

关键技术点的突破对交付效率的影响是决定性的，我建议你把自己发现的问题写出来做比较分析，结合产品竞争力定位最关键的问题，然后通过外部引入或者内部资源寻找高手攻坚。

在人的问题上我给你分享了三点心得，第一给高级别技术人员授权并且给决定权，第二给高潜质员工更高的标准和火线提拔的机会，第三要淘汰组织内业绩和潜力差的员工。

最后我再强调一下，留给我们解决的问题大多是“硬骨头”，**没有轻轻松松可以提高组织效率的事情。这也正是需要你来做经理的原因。**

思考题

公司里说要提升效率，于是提出了测试代码覆盖率，CD 覆盖率，手动重复劳动自动化率等指标并要求各部门执行，你怎么来看待这些提效的举措？

我们说到给予高级别员工决定权，你怎么来平衡给予下属的决定权和你作为部门主管经理的控制力？如果他们做出的技术决定跟你想的不一样呢？

欢迎在留言区晒出你在组织管理方面的经历和疑问。如果有收获，也欢迎你把这篇文章分享给你的朋友。

提建议

上一篇 17 | 招募高手：看人看本质，优秀的人才都是内驱的

下一篇 19 | 危机管理：摆明态度，不要做名义上的领导

精选留言 (1)

写留言

可怜大灰狼

2020-09-30

我个人觉得还是要积极看待。这和职级晋升很像，虽然有时候觉得晋升规则复杂且流程不一定公平，起码是符合大多数人利益的。公司为提高效率提出各种指标要求各部门执行，虽然不一定符合每个部门每个项目组，起码也是符合大多数的情况

展开

作者回复: 我其实对这些指标持保留意见的，我觉得负责该效率的部门应该把自己最优秀的人派到一线去，去寻找那些关键瓶颈。一定要深入到客户中去。