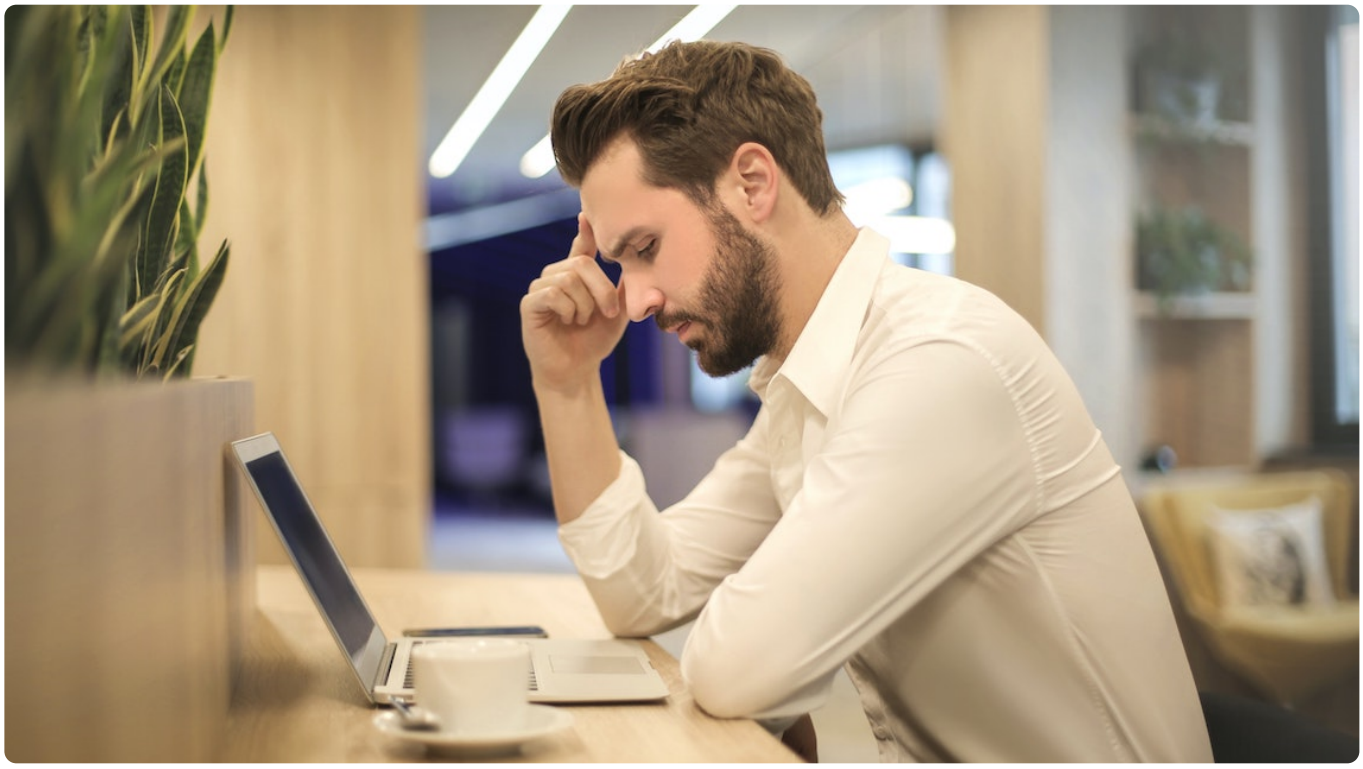
23 | 技术决策（2）： 拥有辩证思维，才能在纠结中负重前行

2020-10-12 许健

技术管理案例课 进入课程 >



讲述：许健
时长 20:04 大小 18.39M

你好，我是许健。今天我们聊一聊技术决策的辩证思维。

但凡重要的技术决策，决策过程一般都是很艰难的，决策时需要考虑的点又是千头万绪，再加上决策效果通常是影响重大的，这些直接导致决策者处于焦虑纠结的状态。

我自己也时不时陷入这样的状态，为此我很想总结出一些方法论指导我做决策，就像专栏前面的文章中多次提到的那样，我会从自己实际已经经历的事情和正在经历的具体案例中做分析、做反思。

最后我发现自己总结出来的指导思想就是辩证思想，但是思路还是比较零散的。为了把零散的思路整合起来，我就去看了有关辩证法的资料，其中的重点就是毛主席的《矛盾论》，重新整理了思路，我才觉得较之以前系统了很多。

今天我就想跟你分享一下怎么用辩证的眼光来看待和分析技术决策，希望可以帮助你做出更合适的技术决策。

让你觉得痛的才是决策关键点

我在专栏中曾多次提到，想要从根本上有提高的事情都不是简单的事情，如果一个重要的问题已经存在很久了，那你也不要有不经过艰苦奋斗就能解决这个问题的幻想。

但是即使有了这种认识，对我们做技术决策又有什么帮助呢？接下来我们先看一个例子：

我们公司之前的配置管理系统 ODB 已经服务十年了，里面存放了各种系统的模型，一方面由于很少去除过时的模型，这导致很多模型动辄上百个字段，而且里面还有很多无用字段；另一方面，数据量的膨胀使得 ODB 系统性能越来越无法满足需求。

一开始我们尝试在原有系统上做改进，但改进版本也就是 ODB 2.0 系统研发失败后，领导最终下定决心做一套新系统，取名 CMS。又花了半年多开发完 CMS 核心，接下来迎面而来的挑战就是原来围绕 ODB 的周边系统搬迁问题，对于大部分人来说都会将未知的事物默认为有风险的，这时没有人愿意做小白鼠。

所以，我们必须把一套众人皆知的、具有足够代表性的复杂系统从 ODB 无缝搬迁到 CMS，这样才能有底气说服其他系统搬迁，而这个足够有代表性的系统就是第一代云计算系统 Stratus。

第一代云计算系统做迁徙的项目由我们云计算部门负责，我让 J 做了项目实施评估，最后 J 告诉我预计需要 24 周，这个时间投入是我自己预计的两倍还多，而且还需要全组每周两次加班。

当时我问 J 为什么要这么大的投入呢？要知道这个决策一旦定了，基本上就意味着我们全组半年内任何其他的事情都干不了。J 给了我一张单子，上面列出了所有需要修改的源代码文件列表，还有所有的测试需要的投入。

另外，J 跟我解释了他特别设计了两处细节保证整个迁移步骤顺利：

第一处细节是设计 ODB 和 CMS 读写开关，一开始 Stratus 仍然读 ODB，然后保证下游还没有迁移到 CMS 系统能正确工作必须双写 ODB 和 CMS，逐步过渡到 Stratus 读 CMS

双写 ODB 和 CMS，再过渡到只写 CMS。

第二处细节是在切换过程中如果一个任务启动的时候是读 ODB，那么即使在执行过程中全局切换到读 CMS，为了数据一致性该任务还是应该继续读 ODB 直到任务完成。

我还记得 J 跟我说：“许健，要么你找别人来负责这个项目，你要是让我来负责，我就是需要这么多时间和这么多人。”我当时有一种孤注一掷，也就是把所有家当都投进去博一把的感觉。

后来实际执行中，我们发现上线前的各种测试比之前预计的更复杂。最后的结果是在 J 的预估投入之外，又加了一个月的高强度集成，我们才交付 Stratus On CMS 这个任务。但是我们上线没有导致任何生产事故，总部的云计算负责人 K 事后专门给我和 J 写邮件，K 说我们做的工作就像是在高速公路上不熄火换引擎，我们做这么大的动作没有出事故非常不容易。

多年后在做 AI Ops 异常检测平台的项目时，J 还让我做过更难的选择：“许健，我们现在有两种做法：一种是我们按照现在的方法做，三个月后我能交付一个效果比现在好的方案，但是这个方案还是没有办法使客户大规模上线；第二种是我们用新办法投入六个月，到时我们有可能取得突破，使得大规模上线客户成为可能，也有可能什么都做不出来。”我最后选择了第二种方案。

因为 J 比较资深，所以一般我让他处理的问题都很关键，而每次他在跟我落实具体投入的时候都让我觉得挺痛的。这么多年下来我开始“喜欢”上这种痛的感觉，如果我认定的重要问题在做决策的时候自己不觉得痛，我反而会觉得不对头，一定遗漏了什么关键点，毕竟世上没有天上掉馅饼的事情。

总结下来就是：**如果你不觉得痛，宁愿相信有什么地方不对劲，那么就应该继续挖，直到挖到那个让你痛的点为止。**这个思路在我做决策的时候屡试不爽。

比如最近我们要在年底前交付基于 Kubernetes 的流量管理方案，老孟是总负责，我问老孟需要我做什么吗？他说不需要，我当时就跟老孟说我的感觉不对，这么重要的一个项目，我怎么觉得你给我提的要求一点也不痛呢，然后老孟说了两点要求：

现在总部领导一直在强调中美团队合作不够好，信任有问题，但是这个项目要说服总部领导把执行全权交给老孟负责的中国团队完成。

有一个关键依赖是全局流量切换模块 GTR，该模块负责人身上的高优先级工作很多，能不能让他承诺把我们对 GTR 的依赖排进优先级。

这两个要求就有点难了，特别第二个点如果我搞不定我只能自己贴人去做 GTR，这意味着我要砍别的项目了，相当于割那边的肉补这边，这对我来说是很痛的。但是这个感觉让我至今难忘，因为这种痛这说明我们在解决核心问题了。

后来我回顾这个项目，发现项目的交付和后来老孟提到的两点紧密相关，可以说是绕不过去的“痛点”。中美合作的要求如果不落实，就会导致决策缓慢、拖慢整体节奏；关键依赖不解决，后续迁移就跟不上，因为一个新系统如果不去接手原来系统的流量，那就不能算作“完整交付业务价值”。

矛盾的普遍性和特殊性

前面我们更多的是从决策的艰难程度去谈，但只是靠着这样的感觉去指导我们做决策还不够，所以我们还要考虑到矛盾的特性。

毛主席在矛盾论中说：“我们的教条主义者在这个问题上的错误，就是**不懂得必须研究矛盾的特殊性，认识个别事物的特殊的本质，才有可能充分地认识矛盾的普遍性，充分地认识诸种事物的共同的本质。**”

我现在看到这句话特别有感触，我觉得我们部门之前就是没有懂得这个道理，才导致我们做了很多努力，却没有特别出色的业绩。这个道理就是想要研究矛盾的普遍性，先要研究矛盾的特殊性。我给你举个例子说明：

我之前在组织管理那一节的思考题里面曾经提到过开发效率的问题，事实上提高业务开发的开发效率是我们的核心任务，我们有一个团队专门负责测试效率这个事情。

这个团队开发了不少工具来提高开发效率，比如 CD Pipeline（持续集成流水线）；测试覆盖率和自动化率统计报表，我们内部又叫这个报表耻辱墙，因为他们把这个报表贴在公司的各个人流量大的场合；还有一个叫 My Stage 的可以给开发人员创建独立的测试环境的系统。

但问题是这么多年了，业务团队始终在不停地抱怨测试效率低下，而负责测试效率的团队除了抱怨测试环境的基础架构不如生产环境稳定之外，也抱怨开发团队没有对这个问题足够重视，没有编写足够的测试用例。

在今年年初的时候我曾经思考过这个问题，我觉得以一个团队去服务上千名业务开发，如果平均用力，很难起到显著作用，还不如集中力量，盯准一个业务团队在一个点上深入挖掘，我建议在上海集中把支付团队的测试效率提上来，我选择这个方向考虑了下面三点：

第一点，支付是我们公司这两年的重中之重，支付团队的测试效率是值得投入的。

第二点，必须深入一线，并且派遣我们组织内的高级别技术人员到支付团队内部去。以保证我们能够在这个点取得突破。

我认为之前团队最大的失误就是浮在表面而不是深入一线，因为组织内最优秀的人都在做平台而没有去一线了解具体情况，所以最后做出来的平台，总是由于这样那样的原因，让业务开发的测试效率提升无法达到预期。

第三点，这也是我考虑问题的重要方面，上海有 150 多人的支付团队，如果我们在上海按照第二点提到的思路来做，是有本地的支持的。

真正做起来以后，其实也碰到了一些问题，进度没有我想象的顺利，但是我们认为找支付部门做攻坚就是正确的策略。

为什么这么说呢？因为我们正是通过给支付团队做服务这个**“特殊用例”**，才陆续发现了持续集成流水线不规范、消息中间件系统在支付测试场景使用不便、集成测试用例不稳定、多测试环境跟外部第三方支付供应商集成遇到的安全限制等一系列问题。

在努力了一个月之后，支付部门有两个 Scrum 团队已经开始在日常工作中采用集成发布的流水线。这正是因为我们深入到别人的业务中了，才能体会到这个部门需求的特殊性，再然后把众多特殊的个案做归纳汇总，才会慢慢积累经验，总结出共性。

我现在回头看，这其实就是给我们做技术决策提供了一个很好的思路，**不要一上来就说我要做一个平台，更好的方式是你有一个做平台的心，但是从解决一个具体的客户的问题开**

始，把这个客户服务好，从中了解实际的需求，解决好了以后再去找下一个客户，在特殊性中总结普遍性。

这个思路到底靠不靠谱，我们还是要用实际操作去验证。最近我们重新启动云原生体验（Cloud Native Experience）的开发，其实去年我们做过一次，但是做出来的效果并不好，每一个试用的客户总有一些需求我们不能很好地满足。

于是这次我们重启项目之前，内部先开了一个会，团队的骨干老 T 在会上就直接说：“我很不喜欢我们的项目叫 **Generic** On Tess （Tess 是 eBay 的基于 Kubernetes 的云解决方案的代号），我觉得如果现在就奔着 Generic 去，我们很难成功。” 这里老 T 说的“Generic”其实中文就是通用的意思，我认同老 T 的想法，觉得不能一上来就搞普遍性。

那个会议我们讨论了 3 个小时，会后我找到项目负责人，我说我们在 2020 年第四季度和 2021 年第一季度各选取两个具体的客户吧，我们先特殊，再普遍，具体计划如下：

第一，上海的数据分析部门因为安全合规要求，正在搬迁他们的应用到新的符合安全要求的 Security Zone， 我们 2020 第四季度的目标是让这个部门的所有跟云平台的交互都能在我们新的 Cloud Native 的平台上无缝完成。

第二，年底前，我们云计算部门应该能够完成 PCI DSS 的合规审核，这样我们 2021 年第一季度就专注支付部门的应用采用云原生的方式部署，对我们自己的要求也应该是支付部门能够在我们的平台上无缝完成日常工作。

这样我们就从两个具体的部门用例中学习他们的特殊性，并总结公共的普遍性。在跟数据分析部门接触的这几个礼拜下来，我们确实发现他们实际碰到的问题跟我们的想象有不少差别，比如他们把一个系统下的不同组件放在同一个应用实例下，而我们原先的模型是一个组件一个应用实例；再比如他们的应用很多是有状态服务，但原先我们认为他们大部分是无状态服务。

根据实际情况，我们及时做了技术解决方案的调整，如果我们不从一招解决所有客户问题的“普遍性”思路，转变成先解决一个具体的客户的“特殊性”思路，那么这个调整就不会发生，我们就很容易重蹈覆辙。

这里我还要额外提一句，如果你去看主席的矛盾论，关于矛盾的普遍性和特殊性我在本节引用的那句话，还有后半句：“**另一方面，不懂得在我们认识了事物的共同的本质以后，还必须继续研究那些尚未深入地研究过的或者新冒出来的具体的事物**。”

这给了我未来做决策的指导，我在这里可以预测一下，当我们明年对数据分析和支付部门的用例学习整理进行平台化推广给更多部门后，我们还会再次碰到下一个阶段的特殊性问题。

比如说，数据分析部门跟数据打交道的特殊性，以及我们公司没有成熟的数据测试平台在测试环境提供高质量测试数据，很有可能我们的云平台需要去适配实际业务需求，也就是在生产环境下部署数据分析部门测试应用，使得这些测试应用可以获取生产环境数据。这就是主席说的“新冒出来的具体的事物”。

用发展的眼光看待矛盾

我清楚地记得有一次公司的高管 Debasish 在到上海考察，那时他说我们整个部门（包括中美）是**“一直活在明天”**的部门，意思就是我们老是说我们要用最新的技术，用下一代的系统来解决基础架构管理的难题，却不踏实解决现在眼前的问题。

我在之前的文化建设中提到我们部门要变成“说话算数”的部门，这里我认为最大的阻碍就是要求我们做的新东西太多了，所以我一直对做新东西持保守态度，但是最近我对做新的系统这个问题的认识有修正。

我在组织管理那节课中提到的 Account Resoure Quota 的例子，我在相当长的时间内是认为该项目不能有效解决 Capacity 管理的难题，而且该项目并没有得到 Capacity 团队的充分认可。

但是随着项目的推进，我有了进一步的认识，我开始问自己 Capacity 团队已经按照现有的方式管理公司资源这么多年了，为什么在控制资源利用率的前提下，加快客户获取资源的速度这个问题一直没有显著进展呢？

如果我们全面地做分析，就应该把事情和人都考虑到：Capacity 团队用现有方式来管理资源，而 ARQ 的发起人对于现有管理方式不满，这个矛盾事实上打破了原有的平衡。**矛盾暴露了，正确地解决了，事物就发展了。事物就是在矛盾的不断产生和解决过程中得到发展的。**

ARQ 这个新项目的启动暴露了矛盾，迫使整个组织重新审视 Capacity 和 Quota 的管理方式的意义是积极的。很残酷的一点是，不启动新项目搞“革命”，就很难引起 Capacity 团队的足够重视。

我再用同样的思路看上文中提到的 CMS 取代 ODB 的历程，对公司最大的意义不在于 CMS 的性能更高，而在于 ODB 到 CMS 的迁移这件事的效果，它会迫使围绕 ODB 打造的所有周边系统重新整理。

进一步说，周边系统整理意味着什么呢？这意味着这十年来积累的所有业务模型得到系统梳理，很多臃肿的模型就可以瘦身了，这就跟一个长期不盘点的仓库因为要引入电子账务系统不得不做一次大的盘点一样。

辩证地看问题，不再只看矛盾的一个方面，也要看到其对立面。“活在明天”的云计算部门因为既要做新的系统，又要维护现有系统，头尾不能兼顾导致人员辛苦但是客户反馈却不好。这个情况当然要改善，但这个矛盾同时也存在积极的一面，就是一直在暴露公司基础架构管理的问题，在解决矛盾的过程中，不断推动更先进的方式引入进来。

辩证地看问题不再静止地看问题，而要发展地看问题。在当前系统的客户满意度不够的时候，主要矛盾是解决当前客户问题，所以应该以提升现有客户体验为主，研发下一代系统解决明天的问题为辅助；而当前客户满意度提升后，要用更少的资源满足更多需求的这个效率上的矛盾，就会成为新的关注点，那时我们就可以考虑以研发下一代系统为主，维护当前系统为辅。

当我用这样的想法再来看待 Openstack 和 Kubernetes，Service Mesh 和集中式流量管理，零信任网络和传统防火墙自动化等问题时，顿时觉得清晰了。

其实，**这些新技术的引入就像一条鲶鱼，可以起到暴露矛盾的作用，打破原来的平衡，关键问题就会逐渐浮出水面，这是好事不是坏事。**我们会因此更加清醒，及时地发现问题，从而想方设法解决问题。

所以这里我想给你的建议就是，我们要辩证地看待和分析技术决策，首先看方向和脚踏实地要兼顾；其次要客观看待矛盾，关注到困难的积极意义，通过“变革”解决问题不断前行；最后不要幻想有“一招鲜”的办法，而是要用发展的眼光看问题，根据当前主要矛盾选择合适的决策方案。

总结

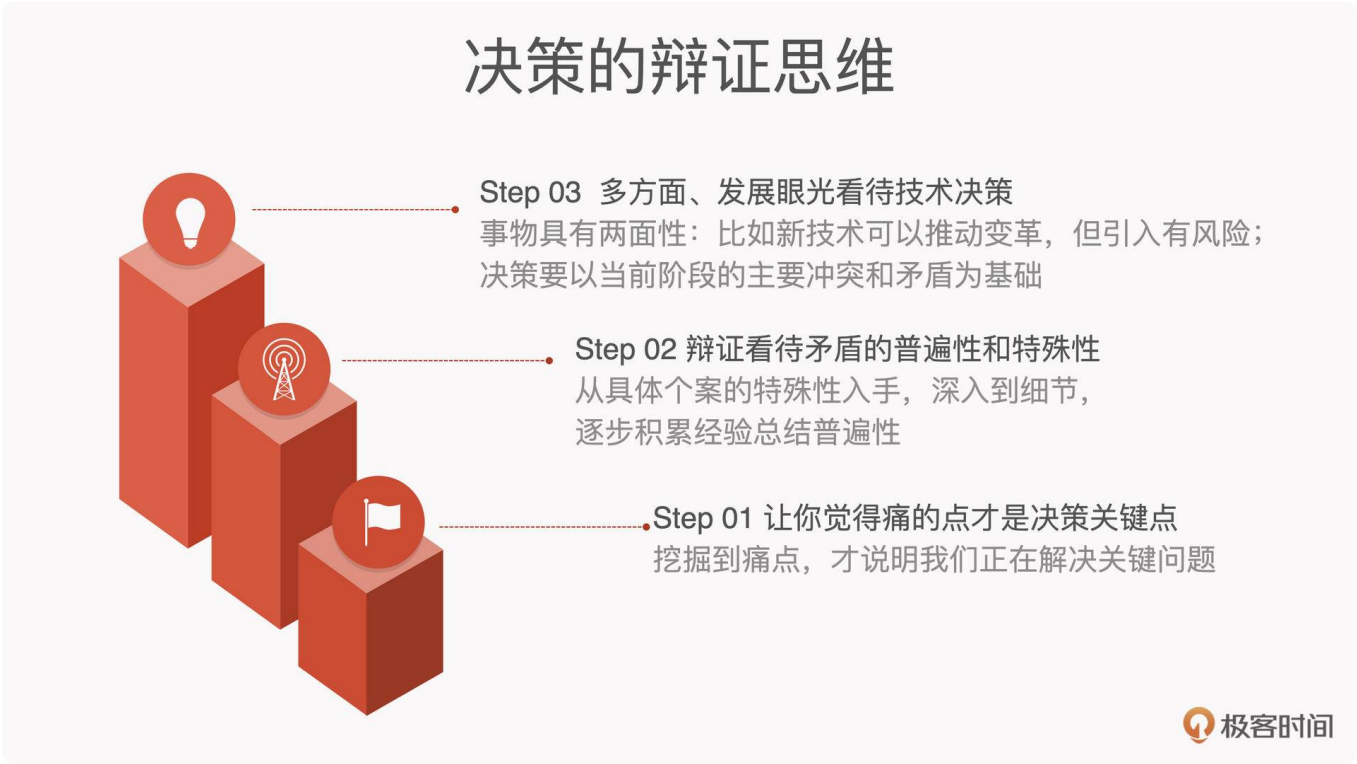
今天我跟你分享了做技术决策的辩证思维，这些思维方式都是我这些年来在不自觉中总结出来的，也希望可以帮助你更加系统地进行技术决策。

首先，如果你在决策中不觉得“痛”，就是一个很强的信号。因为**所有值得你去努力的方向都没有捷径，所有能根本上提升组织和个人的方法操作起来都不容易。**这个痛的信号提示你可能有什么关键的点没有考虑到，这时候不要自我感觉良好，请一定把那个让你觉得痛的点挖出来，这个方法在我所做的技术决策中被反复证明有效。

接下来，在你准备执行决策，特别是解决一个比较大的平台级别问题时，建议不要一上来就要做又大又全的平台，而是应该脚踏实地找一个细分客户方向入手，在一个一个特殊性的解决过程中总结问题的普遍性。**要带着一颗做平台的心，从具体的案例开始**。这样才接地气。

作为技术经理，我们越到后来需要处理的技术决策就越复杂，当你**看到事物的一个方面，一定要提醒自己去看一下它的对立面。**比如在你看到一个技术的好处时，一定要想一下你需要付出的代价。看到新技术所引入的不确定性等风险的时候，也要考虑到它打破原有平衡的积极意义和潜在的大幅提升效率的可能。

最后提醒自己要用发展的眼光看问题，每一个阶段有该阶段的主要冲突点，你的技术决策需要以该阶段的主要冲突和矛盾为基础。而周边环境的变化很可能导致矛盾重点的变化，这时我们也要及时作出调整。



思考题

我们副总最近确立了我们部门以平台安全为第一优先级，并成立了一个单独的团队来负责这件事，副总说平台安全需要做好几年，但是目前我们并没有很清晰的平台安全架构和执行计划，如果你是这个新成立团队的负责人，你准备怎么做？

能否找一个你现实中的项目的技术决策，最好是那种很纠结的技术决策，然后我文中提到的辩证思维来分析这个技术决策案例并分享出来。

欢迎在留言区晒出你的经历和疑问。如果有收获，也欢迎把这篇文章分享给你的朋友。

提建议

上一篇 22 | 技术决策（1）： 技术管理者做什么，团队效率才最高?

精选留言 (1)

写留言

好好学习
2020-10-12

和我最近看的另一个专栏《从0开始学架构》里的架构师的架构原则的思想吻合：适用性，简单性，演化扩展性～

展开