



下载APP



38 | 能力维度二：如何提升解决跨领域冲突的能力？

2022-05-24 郭东白

《郭东白的架构课》

课程介绍 >



讲述：郭东白

时长 24:20 大小 22.30M



你好，我是郭东白。今天我们来讨论架构师核心能力的第三个层次——解决跨领域冲突。

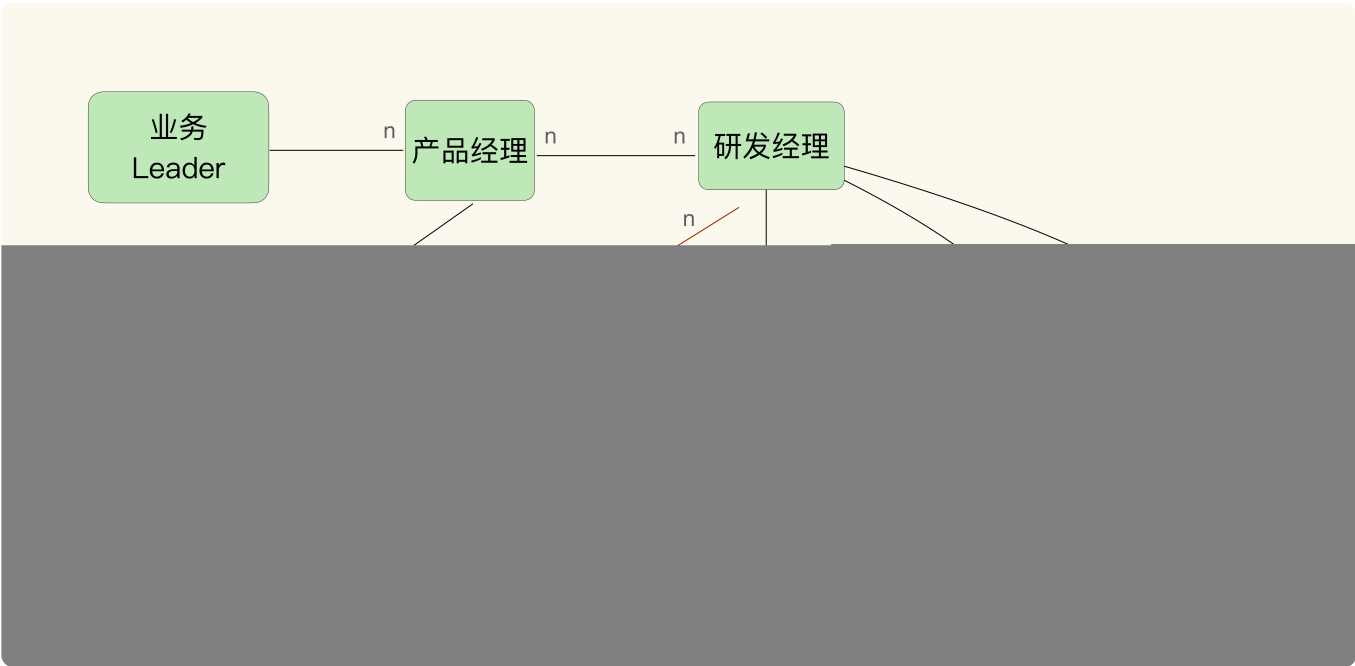
上节课我们讲了从程序员到兼职架构师的跨越，也就是如何搭建解决横向问题的能力。不过，在兼职架构师这个角色中，架构能力是一个加分项，写代码实现需求仍然是主要工作。我们今天要介绍的能力就不再是加分项了，而是作为架构师的主要增值。

这是我们架构师职业成长过程中的又一个重要能力跃迁。在跨领域的架构师或者全职架构师的角色中，代码产出不是我们的主要价值所在，反而变成了一个加分项。角色转换如此之大，以至于很多人虽然多年顶着架构师这个头衔，但却从未完成这个角色的真正转变。



其中的原因何在呢？我们就先从跨域架构师这个职能的缘起来分析一下。

我们无区别一个跨域架构师和兼职架构师这两个角色在概念上的区分。如下图所示，是一张实体关系图：



这张图表示在软件企业或者企业的研发部门中，不同职能在软件架构这件事情上是如何协作的。这里我特别对跨域架构师和兼职架构师这两个角色做了拆分。

一个业务中（也就是一个大公司的 BU，或者是一个小公司）有多个产品线，每条产品线都有各自对应的产品经理。一个产品经理，又往往对应着一个或多个研发经理。一般来说，每个研发经理在负责一个研发领域的同时，还会带领一个团队。团队中有多名程序员，每个程序员各自负责一个或多个软件模块。

跨域架构师和研发经理形成了一对多的关系。也就是说，跨域架构师和研发领域是一对多的关系。这也是为什么我将这个角色命名为跨领域架构师，简称跨域架构师。程序员和研发经理，则和研发领域形成一对一的关系。如果程序员和研发经理同时具备兼职架构师身份的话，这名兼职架构师就和研发领域形成了一对一的关系。

因而这张图表明，从兼职架构师成为跨域架构师，需要完成**从一对一关系到一对多关系的跨越**。

也许你会问了，假设某个研发总监兼任架构师，有多个研发经理向他汇报，那么他负责的领域也很多。在软件架构这个上下文里，他这个架构师和研发领域不是一对多的关系吗？



语权。在兼职架构师这个身份上，他还是在处理一对一的关系，只不过他负责的大领域包含了多个独立子域。

到这里，我们就能找到跨域架构师这个角色的真正特殊性了：跨域架构师对**多个**领域的软件架构**间接负责**，只能通过各领域的研发管理者来**间接影响**自己所负责的领域架构。

你可能要问了：与一个领域相比，多几个领域有什么了不起吗？

请结合第 36 讲中软件架构师的定义来思考一下这个问题：

软件架构师就是为相对复杂的业务定义并引导实施一个结构化软件方案的能力，其中结构化，代表这个软件在其涉及范围内的设计理念、代码结构、实现方式上是同质的。

问题就在这里。假设你负责多个领域的软件架构，而每个领域都有对应的研发经理，甚至领域内还有对应的兼职架构师。那么问题来了，为什么这些领域的设计理念、代码结构和实现方式是处处一致的呢？

在现实世界中，这些领域之间的设计理念等肯定不一致。这些领域有各自的领域目标、领域挑战、需求优先级和相对独立的工作环境，要是一致，反倒不合理了。

现在你应该明白跨域架构师的核心挑战了：**要持续抵抗天然的熵增**，将多个子领域中不同的设计理念、代码结构和实现方式，往同质的方向上进行整合。

上述分析，对于那些由多个独立决策团队构成的大型组织而言，都是适用的。这意味着**跨域架构师的存在是大型组织的需要**。子领域的目标和挑战各异，导致各自的设计理念、代码结构和实现方式也存在不小的差异，进而破坏全局的结构性。这种局部和整体冲突，需要从全局层面优化解决，因此一个组织才需要跨域架构师。

对比一下兼职架构师，这个角色只需要关注自己领域或模块的架构问题就行了，解决的是内部技术债的问题，比如性能、安全、可维护性之类的问题。如果发现问题，自己动手解决即可，不需要化解多个团队之间的冲突。





跨域架构师对这些领域没有管理控制权。每个领域的大佬都有各自的理念和行事方法，不但互相之间有矛盾，甚至每个人都有自己的小算盘。这就是跨域架构师所面临的巨大挑战。

想想看，你从第一天写程序开始，所有的成就感都来自自己的亲身实践。哪怕是解决横向问题，也是靠自己查 Bug、做优化、重构代码。在成为跨域架构师之前，动手能力就完全定义了你！如果把写代码比喻成武力，你就是一个 100% 靠武力生存的人。

但现在不一样了，外交能力突然间取代武力，变成你最需要的核心能力。哪怕代码写得再好，也不能把别人推开自己去改代码。现在只能靠驱动他人来作出正确的判断，才能让想法变成现实。换句话说，你在程序员和兼职架构师阶段崇尚的武力的信念，突然间变成了自己在跨域架构师角色上的最大障碍。

在这种工作场景下，背锅很有可能成为你日常面临的冲突和问题。

从背锅谈起

你有没有观察到这样一个现象。在大厂里，那些工作多年的跨域架构师，“背锅”是他们口中的高频词。你有没有想过为什么呢？

我先给你模拟一个工作场景吧。

假设你是交易域的跨域架构师，一周前刚刚接手这个烫山芋，背景是三个独立的团队分别负责交易、支付和资金领域的开发。昨天公司出现资金损失问题，最直接的原因就是交易团队调整了交易模式，并通知了支付团队。支付团队完成了自己的改造，支付成功后，支付模块将通过调用资金的接口做结算。但是因为资金团队没有收到交易模式调整的通知，所以没有做相应的账户配置变更，导致资金计算错误，公司遭受了不小的损失。

故障追责的时候，三个团队互相掐架。责怪来责怪去，谁都不愿承担故障：



资金团队不承认问题是自己的，认为自己都没发布变更，凭什么要承担这个责任？



支付团队同样不认账。支付代码没问题，问题根因也与支付无关。毕竟项目是交易团队发起的，资金团队沟通不到位，那是交易团队的问题。

同时，讨论中还有人提起架构师也有责任：前任架构师没规划好，你们交接不畅，导致沟通不到位，你有更大的责任，这个资损的故障肯定得由你承担。言外之意就是让架构师来背锅。

当然你也可以选择不背锅，但是就要以中立方的身份来建议应该由哪个团队来承担责任。不过仔细想想，哪个团队都得罪不起，最后只好认命，自己来背这个锅。可类似的事情难免再次发生。

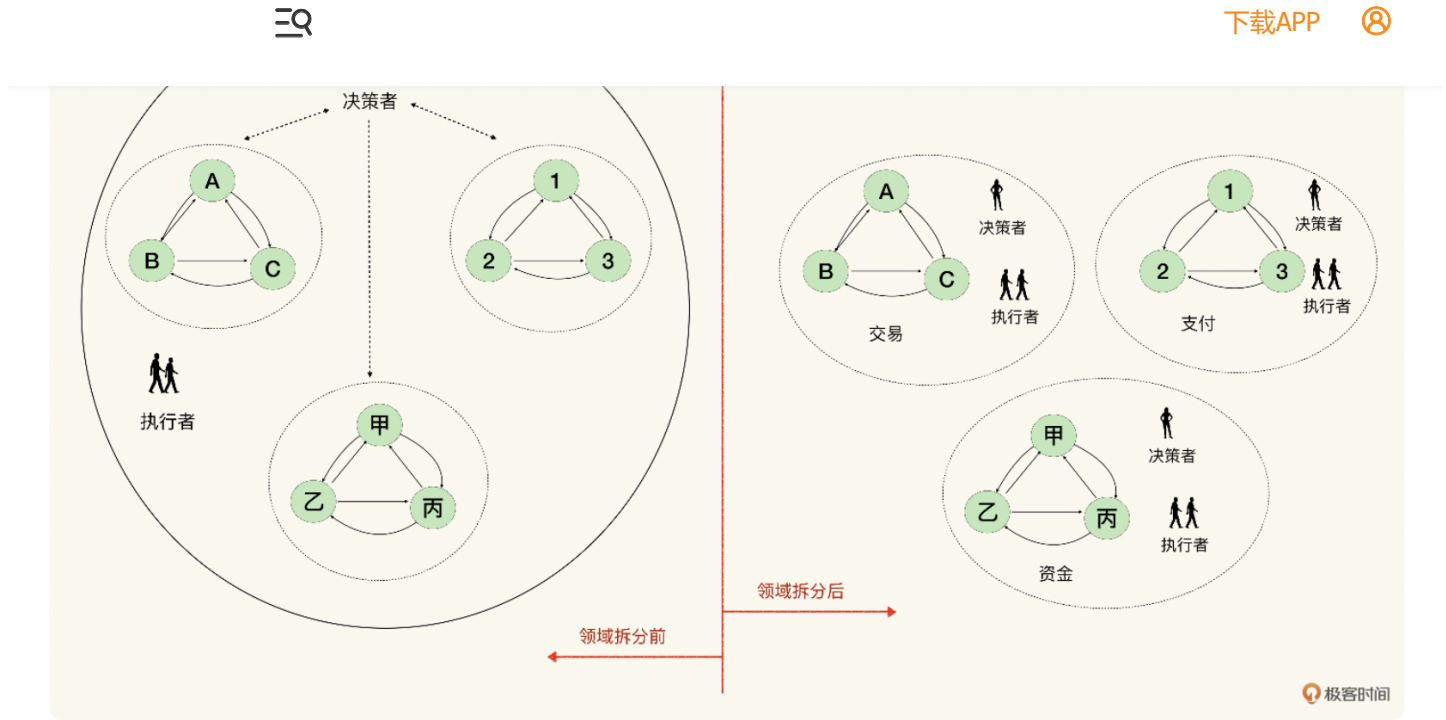
想想看，如果你是一个**有解决跨领域冲突能力的跨域架构师**，会怎么处理这个问题呢？

我们暂时抛开具体的问题，先分析一下跨领域问题的本质：

1. **领域割裂**：本来属于一个大领域的交易支付和资金却被分割成多个子域，各自有独立的团队、数据模型和实体状态。而且，这些子域之间互相影响，数据和状态之间有一定的关系。子域之间需要同步变更，才能保障整个大域处在一个自治的状态。
2. **决策割裂**：每个子域都有各自的决策者，没有一个能为三个子域做统一决策的人。
3. **执行割裂**：每个子域各自独立执行，缺乏执行过程中的同步（Synchronization）。
4. **沟通割裂**：每个子域内部的沟通，无法全部同步给其他子域。

我们看一下这张图，展示了跨领域问题的本质：





在这张图里，左侧大图中的三个虚线圆圈，代表了三个领域的实际关系。在理想状态下，这三个子域应该只有一个决策者和一个执行团队。决策者能够感知每个子域的状态变化，指挥执行者协调执行。各个领域之间也不存在沟通障碍，因为它们同属于一个决策者和一个执行团队。

右侧三个实线的圆圈，代表三个子域被完全切割后的状态。这三个领域原本应该和左侧的圆圈一样，相互之间是关联的。但是这时候，每个领域独立决策、独立执行、内部信息和状态变化也对外不可见，所以跨域架构师只能在三个子域之外干着急。

如果你是这个跨域架构师，该如何突破这个困境呢？

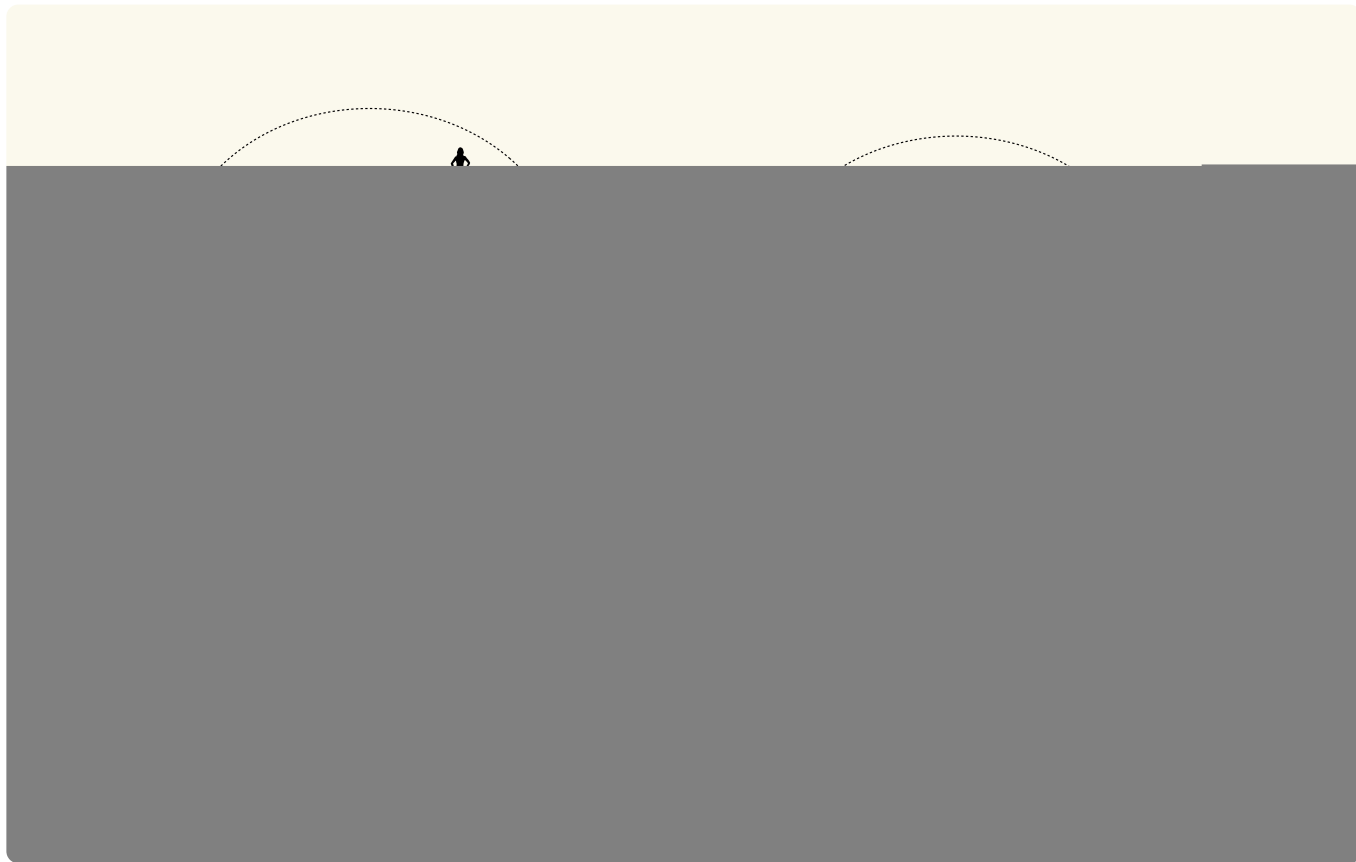
从全局视角做架构干预

换个角度思考，背锅其实是跨域架构师这个岗位价值的另一种体现方式。

可以回看上一张图，与左侧的团队布局相比，右侧的团队布局有它的独特优势。右边的团队更大，每个子域能够钻研得更深，执行得更快。公司成长到一定体量，这么做是必然的选择。每个独立的子域，不可能每天都和其他子域做信息同步。每个子域的决策者，t ☆ 有自己的决策自由度，这样才能快速迭代，子域的分化也能更彻底。而每个子域的目标、挑战和需求不同，以执行上也不可能一致。也就是说，前面提到的领域、决策、执行和沟通的割裂，是为最大化子域增速而付出的必然代价。




从而让三个子域组成的整体比割裂状态下更高效。如下图所示：



在这张图里，我们把每个子域的边界改成虚线，并在子域之间建设决策、信息和状态流转的通道。跨域架构师的作用就是在其中协调子域的决策和执行，从而让新的沟通结构和图中右侧的沟通结构更加接近。

其实我们日常就建立有沟通机制，微信群、邮件组、Wiki、会议、架构规划会都是比较有效的办法。我们在模块二里介绍的架构环境搭建的方法，在这里也同样适用。

总结一下，**跨域架构师的存在就是协调子域之间的决策、执行和沟通，从而平衡全局结构化和局部个性化之间的冲突，最终最大化全局目标的实现。**

你可能会问了，他们互相之间都在高效交流了，那么我这个架构师干什么呢？是不是我要面临兔死狗烹的命运了？不会的，技术能力依然是你的制胜法宝。在这个基础之上，要判断有哪些信息和状态必须在子域之间流转？这些信息的颗粒度需要维持在什么样的颗粒度和频次，才能在保障整个大域状态自治的同时，最大化每个子域的独立迭代速度？



局部，引导大家发现那个最优架构。

这时候你的专业会让大家少走技术弯路。也就是说，你的武力同样有价值，那就是带领大家寻找最优解的价值，而不是直接修复一个大 Bug 的价值。从某种角度而言，你这时候的武力更重要，因为你要靠**真实可靠的技术论据**和**完美的逻辑**来说服大家采用正确的判断。

还是我们刚才拆解的架构设计的案例。在一个高速发展的企业里，往往还有合规、审计、风控和安全等其他维度的问题被各团队忽视了，比如在不同的发展阶段，重新分配不同子域之间的研发投入。这类问题的共性，可以统一归结为之前局部最优的架构决策在全局视角下不再最优。此类决策都是跨域架构师的核心增值所在。

分析到这里，你应该能想到一个跨域架构师该如何突破局部视角的障碍了：

1. 理解整个全局领域的目标。
2. 最大程度地熟悉每个子领域，理解每个子领域的目标、挑战和需求的差异性。
3. 围绕统一的目标去分析局部视角上的差异性，引导各个决策者和执行团队从全局视角上看问题，最终引导多个子域在目标和全局目标上形成对齐。
4. 设计公开的沟通机制，促进子域之间的信息对称，使得每个决策者和执行者能够看到全局的优化目标，以及其他团队的重大决策、执行方式和当前状态。
5. 逐步解决具体的设计理念、代码结构实现方式的冲突，达到全局最优。

关于如何设立公开的决策机制和解决跨领域冲突机制，我们在模块二的各个环节里都有深度的讨论，在这里就不多解释了。

如果能持续不断地重复上面的过程，不断提升你在子领域的知识和全局视角。那么你最终达到的架构设计必然是全局结构化的，从而成功跨过从兼职架构师到跨域架构师的最大的障碍。



尾声：这个锅谁来背？



我们再来分析一下刚才的案例。其实这个案例并不是那么重要，我更期望你从接下来的分析中学会做思想实验的方法。这是在几个团队之间争论不休的场景下，我以架构师的身份做故障裁决时常用的思考方法。

案例里三个团队有各自的局部视角，但是每个团队都缺乏完整的全局视角。我们站在各自的局部视角中，很难判断谁对谁错。

而类似多个团队之间的判责，你可以**引入这样一个思考实验**：如果没有三个团队，只有一个具备超级大脑的人实现了整个系统，那么他是在什么阶段引入了这个故障呢？

这种思考方式把所有讨论者带到了全局视角。虽然交易团队不直接调用资金团的接口，但交易团队对资金团队形成了一个隐含依赖。交易的变更，必须和相关的资金变更同步，才能保证全局业务语义的一致性。

也就是说，交易团队发起了一个开始原子事务的语句“Begin Transaction”，却没有实现结束原子事务“End Transaction”和失败回滚的逻辑。根据这个思想实验，我会判定**故障的责任人是交易团队**。

因为他们发起了具有原子事务属性的变更，但没有保障相关支付和资金变更的完整性。也就是说，在交易发起一个没有保障的原子事务的那一刻，这个故障就产生了，所以交易团队应该负主要责任。

具体到这个案例，如果团队足够大，线下交流无法保障，那么跨域架构师就要在未来系统的设计上，通过机制来保证类似变更的原子性。你可以对交易模式设置版本号，要求相关资金逻辑必须引用对应的版本号才能实施操作。如果一个订单的版本号校验不通过，就不允许对这个订单做资金流转的操作。

我特别要强调一下，**跨域架构师千万不能充当和事佬**。也就是说，你明知道交易团队负责，但还是让自己来背这个锅。这种态度会影响你发现根因，最终无法为负责的领域引入正确架构。这是我们之前在模块一法则六里提到的架构师必须具备的勇气。





跨领域的无人认领的顽疾。

从我的观察来看，这是很多非常优秀的横向领域专家，不能成功成为跨域架构师的主要原因：**缺乏解决跨领域冲突的能力和勇气**。其中，往往勇气是更为主要的原因，因为前者可以通过训练甚至是失败后的修正来补齐。

或许你会想，我为啥要做跨域架构师呢？我做个兼职架构师不是挺好的吗？我钻研技术，有什么问题自己动手解决，也不用去求别人。解决好了都是加分项，解决不好也没人责怪我。

我在最初十年的职业发展中也是这个思维，但我觉得时代不同了。2000 年我刚进入软件行业的时候，互联网的竞争远没有今天这么激烈，团队间的协作复杂性也没有今天这么大。

现在，多数互联网公司越来越大，协作也从公司内部跨越到公司外部，企业对跨域架构师岗位的需求数量远远超越了以前。我认为这是个趋势，也相信这应该是你来学习这门课的原因之一。

小结

跨域架构师为多个子域的软件结构的合理性负责。跨域架构师是为冲突而生的，负责处理一个组织内在的，也就是局部和全局之间的冲突。跨域架构师的存在，是异构组织的需要；是因为某个领域内部的短期目标，与更大领域或者更长期的目标不一致。而这个冲突必须从更高层面看清楚，并结构化地解决。

一个跨域架构师的存在，就是让跨领域的结构性风险降到最低。你要通过设立跨领域的沟通和冲突解决机制，让不同子域的决策者和执行者能够预见、避免，并最小化子域与全局之间的冲突。此外，还要通过统一全局目标，引导所有子域整体的结构性。当然，更需要有足够的勇气去发现、面对和解决这些冲突。

解决冲突的过程，也就是你这个兼职架构师创造增值的过程。如果能持续能做好这件事，也就能跨过从兼职架构师到跨域架构师的最大障碍了！





下载APP



发现有些同学学得苦兮兮的，硬找各种角度给自己励志。我很不理解，也不太推崇这样的做法。

在学习的过程中，我认为找不到任何乐趣的人，是学不到任何真正的知识的。我总认为如果学习给了你多巴胺，才能真正拥抱知识。就像我有时候写文章、做抖音视频，忙得废寝忘食，但根因是我自己很喜欢做这个事情，很享受这个过程。我也希望你学这门课能学得开心。

思考题

今天我们改变一下作业形式，只留一道必做题。学完这节课，我相信你已经比较清楚跨域架构师和兼职架构师的差异了。那么请为未来的你，也就是一个跨域架构师，设计一个小工具，帮助你自己更好的跨越从兼职架构师到跨域架构师的障碍。这个工具能让自己更好地发现、预见、结构化地解决团队合作间的冲突。

我建议你用如下格式在评论区回复一下：“我要设计一下 XXX 工具，这个工具可以做 XXX。有了这个工具，我可以更好地发现、预见、解决 XXX 类型的团队间的冲突了”。当然，不需要讲怎么设计这个工具，而是简单描述一下它应该具备的功能。

对于这个作业，最后还有两个小建议。第一，请不要重新设计一遍 Jira、Confluence 这样常见的协作工具。第二，如果你在评论之前，发现别人已经发表了类似的想法。我建议你给这个想法点个赞，或者在已有的想法上做个改进。

好了，我们今天的内容就是这些，下节课再见！

分享给需要的人，Ta购买本课程，你将得 20 元

生成海报并分享



赞 2

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。



下一篇 39 | 能力维度三&四：如何从做技术到为企业创造生存优势？

精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

