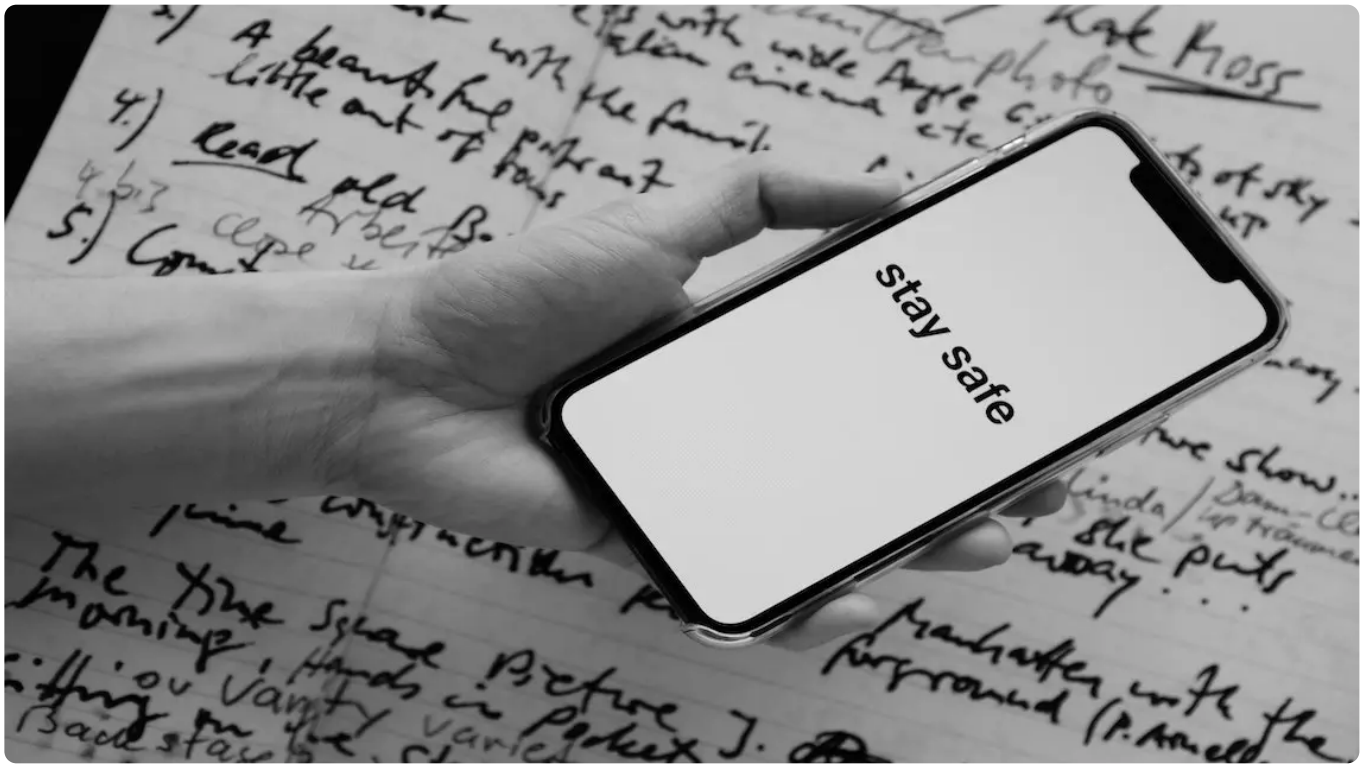
17 | 通用技能（下）：架构师如何保障交付与沉淀知识？

2022-03-01 郭东白

《郭东白的架构课》 课程介绍 >



讲述：郭东白
时长 28:49 大小 26.40M

你好，我是郭东白。架构师在架构活动中主要有四个作用，分别是建设共识、控制风险、保障交付和沉淀知识。上节课我们讲了前两个，这节课就来讲保障交付和沉淀知识这两个。

保障交付

保障交付意味着**架构师能够降低大型架构活动的不确定性和复杂度，最小化架构方案，最终保障高质量的交付。**其中关键动作有三个：降低不确定性、控制复杂度和最小化架构方案。

事实上，不确定性和复杂度也是交付架构活动所面临的两大困难点。毕竟互联网时代下的竞争环境、技术环境、监管环境和商业环境，让所有研发行为都具有非常高的不确定性。

此外，由于参与方众多，那么每个参与方所带来的不确定性也都会传递到架构活动中来。

而不确定性本来就会带来复杂度，再加上互联网企业天然存在的沟通障碍，那么复杂度自然会进一步被放大。因而降低不确定性和控制复杂度就是架构师在保障质量交付中需要克服的主要困难点。

降低不确定性

不确定性的来源有多个方面。**首先是目标的不确定性。**这主要是赞助方对目标的不确定而导致的。关于这一点，我在第 18 讲会讲系统的应对方法。

第二是**资源的不确定性**。在我看来，这是互联网时代架构师所面临的最大挑战。无论是国内还是国外的互联网企业，往往通过类似于虚拟机超卖的方案去刺激团队的产出。

企业往往会同时有多项研发活动，而企业的整体研发资源不足以完成分配给所有研发活动的任务。那么架构师就必须在有限的研发资源池里争抢份额，以保障架构活动的交付。

用国际化电商的项目来举例。一方面，项目需要全球多个国家的运营、产品、研发和测试人员的配合，而这些角色都有各自的 KPI 和必保项目。另一方面，其他项目的负责人也在争取这些角色的支持和配合。那么我们怎么拿到自己应有的资源份额呢？

我们之前强调的两个生存法则是必要条件：确保架构活动的目标正确，以及最大化项目的商业价值。这是王道。当然，还有其他法则要遵守。

比如尊重人性，发掘参与者的利益诉求，最大化参与者的个人投入度。否则当架构目标与参与者个人及其团队利益并不完全一致时，就需要投入额外的激励来保障他们的投入度。我曾见过有家公司为了保障一个为期半年的项目的成功，便为所有参与者多发了半年工资作为奖金。如果你听说互联网公司的工作时间是 9/11/7，不要奇怪，重赏之下必有勇夫。

除技术资源外，还有运营资源、办公环境等等也是有限的资源，我们需要与合作的项目经理来保障这些资源是充足的。项目上线后，相关功能在多个场景下的入口流量保障（首页、搜索、推荐、大促承接页、详情页等），对于项目的成败也至关重要。

其实在互联网的研发环境下，**只要是有限的资源，最终都会变成稀缺资源**。发布窗口、物理机、计算资源、算法、A/B 测试，都可能成为意想不到的交付障碍。就像协调多数人参

与的时间窗口资源，经常是大企业里架构项目的瓶颈。

在电商类项目中，协调时间窗口就极其困难。每个国家都有固定的节假日，分布在全国各地的用户还有不同的工作与休假习惯。而每个国家的业务都有各自的大促时间和项目交付时间。那么真正留给架构师去协调团队，参与大规模集成测试与验收的时间窗口就非常少，一般每个月只有一两个。这种时间窗口必然是各大项目拼抢的对象，因此务必提前锁定并看护好。

不仅国际化的项目面临着时间窗口的挑战，国内的项目也一样。假设一个电商项目在第四季度启动，必然会撞到双十一和双十二大促，时间窗口就更稀缺了。总之，盘点并保障好稀缺资源的供给，对于项目的成败非常关键。

第三，商业与技术环境的不确定性。针对这一点，我们在法则五里已经做了详细的应对办法的描述。除此之外，我还想从交付确定性的角度来谈谈应对办法。

最好的办法就是**在缩短阶段性交付周期的同时，增加技术方案的抽象性**。缩短阶段性交付周期，怎么理解呢？在一个大项目的初期，无论是架构师还是其他参与者，对项目的理解都比较有限。如果把架构活动拆分成多个阶段性的交付点，在线上看反馈。那么我们就可以根据线上数据来看商业或技术环境变化对架构目标、商业效果的影响，而不是凭空猜测。

增加技术方案的抽象性，指的是尽量提升 API 设计对技术选型的鲁棒性，也就是提升接口和模型设计的抽象性。那么在之后的交付阶段，我们就能对次优的技术选型做更正或者升级。

第四，用户需求的不确定性。指用户需求与我们的期望不一致，或者用户需求随着时间发生了较大变化。

应对方案除了从人性出发的设计思考外，还可以**基于增量价值来交付单元。**通过线上用户的真实行为来判断用户需求是否与之前的调研一致，同时根据预期行为偏差和效果分析，来决定是否需要对用户体验做出调整。

除了上述因素外，还有文化环境、组织结构等其他不确定性因素存在，不过它们很少在架构活动期间发生巨大变化，我们就不需要做相应的应对策略了。

复杂度控制

复杂性和不确定性看起来是一码事情，其实差异很大：

> 不确定性是指问题随时间推移，发生了不连续的、不可预测的变化。
>
> 复杂性则强调问题或者解决方案，很难用几个简单的维度去描述。

那么如何控制复杂度呢？第一，从问题域层面分解架构规划和交付方案。

具体而言，就是将整个架构活动按照**问题域**，分解成不同领域的子问题。然后在每个问题域，从粗到细一步步分解成细粒度的执行方案。

事实上，研发团队日常的任务分割往往也是按照问题域拆解的。所以你可以最大程度地利用日常已经形成的沟通协作机制，最小化交付风险。

还是来看个例子吧。假设我们在做一个大规模的电商系统内容化升级的项目，需要根据日常的垂直领域划分——流量、导购、搜索推荐、交易、资金、评价、商品、商家、服务、履约、物流等，把整体任务分配到各个领域中去。而各个领域呢，则会根据内容化的需求来决定各自领域的规划。比如流量领域，需要有内容承接部分的定制、投放、生成等。而履约和物流领域，则跟这个项目几乎没有任何关系。

不过在划分的过程中，你可能发现之前的领域划分不是最优的。还是这个例子。当每个领域都完成进一步的方案细化后，我们可能会发现，流量、导购、商品、商家、服务等领域存在共性的模块，也就是内容生成和其他的内容操作。如果每个领域都各自开发这些模块，那就会造成大量的浪费。这个时候，可以把这些跨多个垂直领域的内容相关问题，组合成一个新的内容域。由这个新的垂直领域，来完成与内容相关的所有规划和交付，从生成到审核、编码、投放、播放、埋点、监控和分析等等。

第二，增加架构设计方案本身的结构性。

结构性，指的是贯穿企业所有软件实现方案的统一模式。反映在设计上，就是结构化设计（Structured Design），意思是不同领域、不同模块的设计是同构的。

一个很简单的例子。如果所有参与方都采用微服务设计、响应式设计或者前端低代码设计。那么这种结构化的设计有什么优势呢？除了运维测试等方面的优势外，它最大的优势

就在于未来的改动成本比较低。我们做架构永远都不是为了现在，而是为了未来。只有容易适应未来环境的设计，才能最大化企业的生存。这也是我们在法则五中重点强调过的。

举个例子，如果做电商化系统，那么我们可以把整个技术架构分成三层：业务层、中台和没有业务语义的基础设施层。随着我们对业务理解的深入，会发现，中台需要被分成两层——上面一层是业务中台，下面一层是数据中台层和共享技术层。这两个步骤都是水平切分的。

然后我们可以根据业务域，进一步把业务层的模块切换成多个垂直的领域，在每个领域内做再进一步的方案细化。虽然每个领域的商业逻辑有差异，但是它们使用同样的编程语言、微服务框架、测试框架和发布流程。这种企业层面上宏观的结构性，就是我们要追求的结构化设计。

需要注意的是，结构化设计的过程不是你这个架构师口头说说就可以了。**这种宏观上的结构性，来自架构师与其他参与者在每个领域的决策上对结构性的追求。**

比如交易模块，需要先对不同国家的业务需求做分类，从中抽象出共性需求到交易中台。而个性需求呢，则和该国家的特殊业务形态或者监管环境有关，需要留在业务层，避免复杂性侵入到业务中台。

不管这个过程又会影响数据中台和交易相关的实体，比如营销活动、订单、资金账号等实体的抽象。所以对实体的建模，肯定会通过数据中台影响到营销中台和资金中台，以及依赖这两个中台的前台业务。这个时候，我们只有钻研到所有相关模块的领域模型这个层次上，才有可能对整个架构活动的复杂性形成完整认知。

如果你和团队都追求这种宏观结构性，那么在垂直切分的过程中，随着你和团队对细分场景的梳理，像 API 设计、领域模型、消息机制等都将逐渐清晰、更加合理、更加结构化。反过来，如果有一个人出于贪婪，试图破坏自己领域的结构合理性，那么这种行为将会传递到中台，并扩散到其他领域中去。这也是为什么我在法则六里反复强调，求真和有良知的文化对于一个企业来说极其重要。

这个梳理的过程，如果做得比较认真，也会帮助你降低交付的风险。而且随着梳理，你的方案会变得越来越完整，每个模块交付时间的确定性也在不断提升。研发的任务，也从宏

观的领域模型切分，到库表切分，再到字段和消息的切分，粒度越来越细。风险也从刚开始相对宏观的大风险，转移到具体设计上的方案层面上，以及细分模块的交付日期上。

事实上，许多架构活动的唯一目标就是提升企业软件架构的结构性。你肯定参加过技术栈统一、数据模型统一、全站埋点标准化这样的项目，它们的目标都是提升软件系统的结构性。可是，把一个混乱的系统重构成一个结构化的系统的成本非常高。所以我们应该通过日常架构活动的自律，来避免软件结构性的退化。可以这么说，任意破坏企业整体的软件结构性的行为都是可耻的！

当然，有些架构活动的首要目标是构造新的商业模式或者最大化商业价值。这种大的商业模式的变化，往往同时覆盖多个领域和软件架构的多个分层，而且时间非常紧迫。在这种架构活动中，想提升软件设计的结构性就比较难。因为这种项目对结构性一般是有破坏性的。

这个时候，如果你能想办法约束架构活动的爆炸半径，就可以间接地减少对软件结构性的破坏了。关于这一点，你可以参见我在第 5 讲给出的性能优化的项目案例。我通过对性能优化的商业价值的抽象和准确度量，也就是性能损耗概念，约束了整个架构活动参与者的探索范围和探索深度，避免参与者毫无意义的研发资源的浪费，同时也避免了系统结构性的破坏。这种通过强调单一的结果指标来约束架构方案的范围，可以说是屡试不爽。

第三，按照多种方式分割交付模块。

分割交付模块的理念与小步快跑是类似的。比较常见的办法是分期交付，比如按部门、按领域或者按架构分层进行分期。

按部门或领域交付，意思是按照垂直领域分别完成架构项目的交付。按架构分层交付是指先完成最底层的变更，然后再逐渐往上。每一层的交付都保持向后兼容，从而最小化对上层业务的侵入。在未来的课程中，我们还会提到按最小价值单元交付的方法，指的是按照用户场景和需求划分来交付模块。

不论哪种交付方案，好处都在于整个交付过程跟一个松耦合的灰度发布的过程是类似的，以便你在小范围内试错，提前发现问题。另一个好处是降低联调和验收的复杂度。此外，我们开篇提到的不确定性和复杂度的问题，基本上都可以迎刃而解。我团队中的技术项目几乎都是按照这种方式来交付的。

最小化架构方案

不论是分期交付，还是通过单一目标约束架构活动的范围，这些都是最小化架构活动的例子。**这个最小且必要原则，是提升交付成功率的最重要的方法。**

然而在实际项目中，我们看到的多数是与之完全相反的好大喜功的行为，在大厂里尤其常见。有些项目负责人恨不得把项目铺开到整个企业。甚至为了扩大影响力，故意放大项目的工作半径。这种行为是项目交付的大忌，你可以在任何一本项目管理入门书籍中看到。

老生常谈的一个事实是，大多数架构活动都是以失败告终的。虽然很多架构活动负责人不这么对外讲，但真实的情况一定是败多胜少。如果架构活动只有一个成功秘诀的话，那肯定是最小化！

有一个非常重要的架构活动的用户体验指标与最小化有关，即，**被架构活动影响到的业务体量的占比**。这个比例越低越好，最完美的情况是零，也就是架构活动对上层业务完全透明。在交付的过程中，上层业务还可以持续迭代，不受任何影响。

反过来，最差的架构活动的交付体验则是完全阻断上层业务迭代。我曾见过一个国际化中台项目，原本预期三个月交付，然后逐渐推迟到 6 个月、9 个月，最后做了一年，还烂尾了。最恐怖的是，这个项目负责人不听劝阻，选择了一个非常极端的阻塞式交付模式：在项目进行过程中，所有相关的上层业务，都不能做任何交易和营销模式上的变更。这样一来，上层业务就像是杨家将在打仗，三个月没有粮草已经很艰难了。等到被绝食一年之后，根本不用对手挑落到马下，自己就饿死阵前了。

所以请一定记住，务必最小化项目对上层模块的冲击！

沉淀知识

在架构活动中，架构师有着区别于其他参与者的宏观视角，因而有必要通过有效的知识沉淀来保障架构活动的思考与决策质量，也有必要为企业未来的架构活动提供宝贵经验和方法论。一方面，架构师需要沉淀完整和真实的过程记录。另一方面，还要为企业注入逻辑思考，引导企业走向正确的决策。

有些人误以为沉淀知识就是收集、整理和编写文档。的确，不断积累架构活动全生命周期的数字或文档记录十分重要。从某种角度而言，这个记录就是整个架构活动的数字化镜

像。不但能为架构活动参与者提供完整且全面的信息，也能为其他项目的架构师和依赖方提供宝贵经验。比如 Confluence，这个商业化的文档工具就做得非常成功。

然而这不是沉淀知识的全部。我们更需要做的，是**通过各种文档工具、设计工具、沟通工具和复盘工具，为架构活动注入理性思考**。

你可以先花几十秒思考一下这两者的差异：

> 收集、整理和编写文档，是一个数字化镜像的过程。真实世界的行为发生在前，数字化的过程发生在后。这是一个**被动**的过程。
>
> 而注入理性思考的过程，则是靠文档中的严密逻辑来驱动理性思考的过程。文档和设计发生在前，驱动我们及其他参与者理性的、基于事实的思考和决策，以期改变真实世界。这是一个**主动**的过程。

在被动的过程中，架构师就像一个自动化埋点和日志收集系统，忠实地记录着项目过程中的所有行为、现象和结果。

而在主动的过程中，架构师更像是一台计算设备，通过文档来驱动架构活动参与者的思想实验，通过理论推演来提升思考质量，而不是通过写代码、发布、线上试错来完成架构方案的迭代。

还记得我在第五讲提到的性能优化的案例吗？在我们还没有动手改代码之前，我就先定义了性能损耗这个概念，在文档上推演和证明了性能损耗的公式，又在白板上讨论了不同页面、不同场景下的实际度量办法，等等。这个由文档推动的思想实验，使得我们出手之前就已经对它能产生的价值成竹在胸。

你可能会说这个案例太特殊了吧。并不是！只要是项目足够复杂，成本足够高，风险足够大，我一定会要求自己或者项目负责人做类似的推演。这是我实践过的提升大项目成功率的最好办法，也是我作为架构师坚持了很多年的习惯。推演不一定是公式，也可以是文档，甚至可以是被污名化的 PPT。形式不重要，关键在于“主动思考”。

或许你还会说，互联网时代大家就应该试错，花这么多时间推演值得吗？我的回答是：不但值，而且超值。

还是我自己的经历。我博士论文的主题与计算机视觉在医疗领域的应用有关。一般来说，博士论文要有具体的公式推导和结果呈现。但是其中有一个推导是有误的，写代码时并没有发现，直到落笔写论文时才恍然大悟。最后只能倒回去改代码。

而且这个错误发生在图像分割前的特征建模环节，因此我被迫改了前后花费两年时间写的所有代码，从特征提取到特性计算，再到图像分割、三维可视化、三维建模、应力计算。

为了完成论文，大概有半年的时间，从周五早上 8 点到周日下午 7 点，连续工作 50 多个小时。在高速公路上开车的时候，睡着过至少四次。现在回想起来真是后怕啊！

事实上，在我重新写论文，发现并证明这个推导错误，只用了一天多的时间。而我却差点儿因为这一天的大意葬送掉自己年轻的生命！自那以后，我就养成了一个习惯：只要是足够复杂的项目和工作，就一定要在纸上把完整公式、思考逻辑和决策逻辑写下来，反复推敲。并分享给周围同事，让大家一起帮我找问题。

无论在纸上把逻辑推演多少遍，这个成本远远小于让几十个人喊着口号开足马力以血肉之躯冲向一个悬崖的成本。我认为就是这个习惯，让我在架构师这个职业赛道上走得更远也更好。

思想实验不仅能帮助企业，更能帮助我们自己。你可以在网上找找关于亚马逊 6 页纸（Six Paper）的介绍。这也是一个思想实验。这个流程非常耗时耗力。我曾写过一篇文档，改版了 13 遍才通过评审。这些还不是小的改版。每次改版都有十几个人参与评审，其中还包括一个技术 VP。

想想看，这个思想实验的成本有多大！亚马逊的企业信条里有一条是“Bias for action（勇于试错）”。这个信条的完整表述应该是：“在充分的思想实验之后勇于试错。”

总结一下：**一个理想的知识沉淀的过程，不仅包括一个被动的、通过文档来记录活动历史的过程，还包括一个主动的、通过文档来驱动思想实验的、创造历史的过程。**

小结

架构师想要在整个架构活动中持续保障交付，就要持续控制不确定性和复杂度，同时最小化架构方案。不确定的根源来自目标、资源、环境和用户需求，这些外在的影响因素随着

时间会发生不连续的变化。而应对变化的不连续性，一方面，我们要及早并持续确认这些影响因素。另一方面，还要寻找不变量，寻找一个更高、更深层次的稳定抽象。

复杂性来自互联网企业本身的业务模式的复杂度。你可以通过对问题域的分割、对结构性解决方案的持续追求，以及对交付的合理分割来控制。不过这些都是“术”，最重要的还是最小必要原则。千万不要好大喜功，更不要将自己横在业务前面，阻断上层业务的迭代。

最后我们还讲了知识沉淀。一个理想的知识沉淀过程，既包括一个被动的、记录活动历史的过程，还包括一个主动的、驱动价值创造的思想实验过程。这个虚拟世界的知识沉淀与现实世界的架构活动，就好比 DNA 的两条链。知识指导活动，而在活动中，又可以不断创造和验证知识。这两者是互相激励、互相创造和互相影响的过程。

这就是架构师工作的最美妙之处啊！我们的工作，永远都能让自己的思考变得更完美。

思考题

三个思考题，任选一个：

1. 你在项目中经历过的最大的不确定性是什么？采取了什么应对方案呢？结果如何？
2. 我们这两节课讲了建设共识、控制风险、保障交付和沉淀知识这四个基本技能。除此之外，你认为还有哪些技能是架构师必须具备的呢？为什么？
3. 你听说过的最成功的思想实验是什么？它为什么能成功呢？

如果这节课对你有帮助，欢迎你把课程转发给你的同事或朋友。顺便打个小广告，我刚开了个人抖音号，我会定期发表一些比较新、但是不一定那么成熟的观点。欢迎在抖音上搜索“郭东白”并关注，也欢迎你的批评指正。我们下节课再见！

上一篇 16 | 通用技能（上）：如何帮助团队达成共识与控制风险？

精选留言 (9)

写留言

Geek_fd0943
2022-03-03

卡尔·波普说，只有写下来的东西才是能够批判的东西。我们人脑并不是一台逻辑引擎，它只是一个信念和证据反复迭代的玻尔兹曼机，一个模式工具。我们头脑混乱、冲突却可以自以为和谐地相处，直到它被写下来接受自己的，他人的，历史的批判。

知识的沉淀这我有点不同看法。参考桥水基金和得到李育辉的组织管理课程。知识管理...

展开

1

Right
2022-03-03

数据的最大价值体现在运用，而不是存储。被动地记录、整理文档只是存储而已，如果不对其进行主动地思考和运用，那这些文档便毫无意义，更谈不上什么共识的建立。
我曾待过一个公司，强制要求项目过程中任何行为都得落盘文档，一个需求下来能产生大量文档内容，然而这些文档对于后续项目的帮助非常有限，同一个坑可能会被踩几次，即使这个坑之前早已记录。...

展开

1

罗均 - Jun
2022-03-03

非常感谢老师精彩的课程！
这节课的第二个问题，比较有挑战性。有一项技能，不知道是否合适？学生还无法用准确的语言表达这门技能，姑且暂用【洞察全局 Global Insight】，这个想法来自于《毛文选》的第一篇文章《湖南农民运动考察报告》——毛主席当年如何进行“接地气”地考察，又如何基于“数据”洞察当时湖南省乃至全国的革命形式。...

展开

术子米德
2022-03-02

🤔☕🤔☕🤔
* 📖：保障交付的三个关键动作：降低不确定性、控制复杂度、最小化架构方案
    * 🤔：都在喊复杂度要控制，可是多少人说不清楚，到底什么是复杂度。由此带来的恶果就是在毫无觉知的情况下，甚至自以为是正确无比的情况下，就种下恶的种子，直到恶之花开放，才发现复杂到处都是，就像昨晚一场暖暖的春雨，第二天各种花都齐刷刷开...

展开

人间四月天
2022-03-02

非常赞同，回顾自己做项目经历，也是用同样的方法，可能很多读者感觉比较理论，实际上，可能很多人没有做过复杂项目，或者没有想用最小成本创造最大收益去深入思考和实践，再次强调一点，靠无限堆人做项目，是对成本的蔑视和不负责任的，这应该就是架构师站在公司视角考虑问题的根本和价值。

展开

欧阳绍聪
2022-03-02

我前面不太确定，听完后才发言。少了人的维度，人的成长，不仅是自己，还要能成就别人。我个人觉得是少了成就别人，这个不知道算不算是基本素养了。


Ls_
2022-03-02

深有感触，接下来会应用到工作中。


spark
2022-03-01

郭老师，take away~~~
看完一遍，脑里，呈现项目上线倒逼排期的场景，需求澄清、架构设计、数据库设计、团队建设，这是我常用的保障交付的方法。什么是思考和结构化的思考？定义问题和拆解问题，tradeoff，系统全局最优。什么是沉淀知识？需要用递归思维和分治思维，系统化建立技术的宽度和深度~~~

展开


咏晨桃
2022-03-01

每周都更，但是我感觉全书最精华的部分就是生存法则篇，也是用来吸引人的篇，后面的内容索然无味了，能否分析下啥原因

共 2 条评论