

67 | 架构思维篇：回顾与总结

2019-12-24 许式伟

许式伟的架构课

[进入课程 >](#)



讲述：姚迪迈

时长 10:08 大小 9.30M



你好，我是七牛云许式伟。

到今天为止，我们第五章“架构思维篇”就要结束了。今天这篇文章我们对整章的内容做一个回顾与总结。

架构之道

架构思维篇的内容大体如下图所示。

第五章 架构思维	67 架构思维篇：回顾与总结	
	66 架构老化与重构	
	架构范式	65 架构范式：文本处理
		64 不断完善的架构范式
	架构思维 准则	63 接口设计的准则
		62 重新认识开闭原则 (OCP)
		61 全局性功能的架构设计
		60 架构分解：边界，不断重新审视边界
		59 少谈点框架，多谈点业务
	58 如何判断架构设计的优劣？	
	57 心性：架构师的修炼之道	

在前面几个章节，我们已经陆续介绍了架构的全过程：

🔗 17 | 架构：需求分析 (上)

🔗 18 | 架构：需求分析 (下) - 实战案例

🔗 32 | 架构：系统的概要设计

🔗 45 | 架构：怎么做详细设计？

但架构师面临的问题往往是错综复杂的。

给你一个明确的需求说明文档，干干净净地从头开始做“需求分析”，做“概要设计”，做模块的“详细设计”，最后编码实现，这是理想场景。

现实中，大多数情况并不是这样。而是：你拿到了一份长长的源代码，加上少得可怜的几份过时的文档。然后被安排做一个新功能，或者改一个顽固缺陷（Bug）。

我们应该怎么做架构设计？

架构设计架构设计，设计为先，架构为魂。用架构的系统化和全局性思维来做设计。

整体来说，我们这个架构课的知识密度比较高。这在某种程度来说，也是一种必然结果，这是因为架构师需要“掌控全局”带来的。

所以这个架构课对大多数人而言，多多少少都会有一些盲点。如果遇到不能理解的地方，从构建完整知识体系的角度，建议通过其他的相关资料补上。当然也欢迎在专栏中提问。

相比一般的架构书籍来说，我们这一章架构思维篇的内容写得并不长。原因是架构思维的本源比架构规则重要。规则可能会因为环境变化而发生变化，会过时。但是架构思维的内核不会过时。

所以我们把关注的焦点放到了不变的思维内核上。

架构之道，是虚实结合之道。

我们要理论与实践相结合。架构设计不可能只需要熟读某些架构思维的理论，否则架构师早就满天飞了。如果两者只能取其一，我选实践。

从实悟虚，从虚就实，运用得当方得升华。这其实是最朴素的虚实结合的道理。对学架构这件事来说尤其如此。架构思维的感悟并不能一步到位，永远有进步的空间，需要我们在不断实践中感悟，升华自己的认知。

这个架构课内容的前四章为“基础平台”、“桌面开发”、“服务端开发”、“服务治理”。

从内容上来说，由“基础平台（硬件架构 / 编程语言 / 操作系统）”，到“业务开发（桌面开发 / 服务端开发）”，再到“业务治理（服务治理 / 技术支持 / 用户增长）”，基本上覆盖了信息技术主体骨架的各个方面。

有了骨架，就有了全貌，有了全局的视角。

前面四章，我们内容体系的侧重点放在了架构演变的过程。我们研究什么东西在迭代。这样，我们就不是去学习一个“静态的”、“不变的”信息技术的骨架，更重要的是我们也在学信息技术的发展历史。

有了基础平台，有了前端与后端，有了过去与未来，我们就有了真真正正的全貌。

我们博览群书，为的就是不拘于一隅，串联我们自身的知识体系，形成我们的认知框架。

信息科技的整体架构，与我们的应用软件架构息息相关。架构分基础架构和应用架构。选择基础架构也是构建业务竞争优势的重要组成部分。

从技能来说，我们可能把架构师能力去归结为：

理解需求的能力；

读代码的能力；

抽象系统的能力。

但架构师的成长之旅，首先是心性修炼之旅。这包括：

同理心的修炼，认同他人的能力；

全局观的修炼，保持好奇心和学习的韧性；

迭代能力的修炼，学会反思，学会在自我否定中不断成长。

业务的正交分解

架构就是业务的正交分解。每个模块都有它自己的业务。

这里我们说的模块是一种泛指，它包括：函数、类、接口、包、子系统、网络服务程序、桌面程序等等。

这句话看似很简单，但是它太重要了，它是一切架构动作的基础。

架构行为的三步曲：“需求分析”、“概要设计”、模块的“详细设计”，背后都直指业务的正交分解，只是逐步递进，一步步从模糊到越来越强的确定性，直至最终形成业务设计的完整的、精确无歧义的解决方案。

对业务进行分解得到的每一个模块来说，最重要的是模块边界，我们通常称之为“接口”。

接口是业务的抽象，同时也是它与使用方的耦合方式。在业务分解的过程中，我们需要反复地审视模块的接口，发现其中“过度的（或多余的）”约束条件，把它提高到足够通用的、普适的场景来看。

在架构分解过程中有两大难题。

其一，需求的交织，不同需求混杂在一起。这是因为存在我们说的全局性功能。其二，需求的易变。不同客户，不同场景下需求看起来很不一样，场景呈发散趋势。

但无论如何，我们需要坚持作为一名架构师的信仰：

任何功能都是可以正交分解的，即使我目前还没有找到方法，那也是因为我还没有透彻理解需求。

怎么做业务分解？

业务分解就是最小化的核心系统，加上多个正交分解的周边系统。核心系统一定要最小化，要稳定。坚持不要往核心系统中增加新功能，这样你的业务架构就不可能有臭味。

所以业务做正交分解的第一件事情，就是要分出哪些是核心系统，哪些是周边子系统。核心系统构成了业务的最小功能集，而后通过不断增加新的周边功能，而演变成功能强大的复杂系统。

这里有一个周边功能对核心系统总伤害的经验公式：

$$\sum_{\text{对每一处修改}} \log_2(\text{修改行数} + 1)$$

同一个周边功能相邻的代码行算作一处修改。不同周边功能的修改哪怕相邻也算作多处。

这个公式核心想表达的含义是：修改处数越多，伤害越大。对于每一处修改，鼓励尽可能减少到只修改一行，更多代码放到周边模块自己那里去。

在 “[62 | 重新认识开闭原则 \(OCP\)](#)” 这一讲我们介绍了开闭原则。它非常非常重要，可以说是整个架构课的灵魂。总结来说，开闭原则包含以下两层含义：

第一，模块的业务要稳定。模块的业务遵循 “只读” 设计，如果需要变化不如把它归档，放弃掉。这种模块业务只读的思想，是很好的架构治理的基础哲学。

这告诉我们，软件是可以以 “搭积木” 的方式搭出来的。核心的一点是，我们如何形成更多的 “积木”，即一个个业务只读、接口稳定、易于组合的模块。

我平常和小伙伴探讨架构时，也经常说这样一句话：

每一个模块都应该是可完成的。

这实际上是开闭原则业务范畴 “只读” 的架构治理思想的另一种表述方式。

要坚持不断地探索各类需求的架构分解方法。这样的思考多了，我们就逐步形成了各种各样的架构范式。这些架构范式，并不仅仅是一些架构思维，而是 “一个个业务只读、接口稳定、易于组合的模块 + 组合的方法论”，它们才是架构师真正的武器库。

第二，模块的业务变化点，简单一点的，通过回调函数或者接口开放出去，交给其他的业务模块。复杂一点的，通过引入插件机制把系统分解为 “最小化的核心系统 + 多个彼此正交的周边系统”。回调函数或者接口本质上就是一种事件监听机制，所以它是插件机制的特例。

领域理解

应对业务需求的变化，最好的结构就是：最小化的核心系统 + 多个彼此正交的周边系统。

但是光理解了这一点，并不足以根本性地改变你的架构能力，因为这里面最难的是领域理解。所以需求分析很关键。怎么做需求分析？这一点要讲透真的很难。

我们用的是笨方法。把整个信息科技的演进史讲了一遍。

我们用穷举的方式来讲信息科技的半部演进史。为什么我说是半部？整个信息科技的发展，我们把它分为程序驱动和数据驱动两个阶段。

程序驱动的本质，是自动化的极致。以前，自动化是非常机械的，要完成自动化需要极大的难度。但是，软件的出现让自动化成为一种普惠价值，这是信息科技的上半部演进史带来的核心收益。

但到了数据驱动，事情就变了。我们甚至有了新的专有名词，比如“智能时代”，或者“DT 时代”。很多人想到智能，想到的是深度学习，想到的是机器视觉。但其实这非常片面。马云把上半场叫 IT，下半场叫 DT（数据科技），非常形象而且深刻。

我们的架构课，把话题收敛到了“如何把软件跑起来，并保证它持续健康运行”这件事情上。

但从企业的业务运营角度来说，这还远不是全部。“[🔗 54 | 业务的可支持性与持续运营](#)”我们稍稍展开了一下这个话题。但要谈透这个话题，它会是另一本书，内容主题将会是“数据治理与业务运营体系构建”。

我希望有一天能够完成它，但这可能要很久之后的事情了。

结语

今天我们对本章内容做了概要的回顾，“架构思维篇”到此就结束了。理解了本章的内容，对于如何构建一个高度可扩展的软件架构你就有了基本的认知。

但不要让自己仅仅停留在认知上，需要多多实践。

架构的功夫全在平常。

无论是在我们架构范式的不断完善上，还是应对架构老化的经验积累上，都是在日常工作过程中见功夫。我们不能指望有一天架构水平会突飞猛进。架构能力提升全靠平常一点一滴地不断反思与打磨得来。

在应对架构老化这件事情上，不要轻率地选择进行全局性的重构。要把功夫花在平常，让重构在润物细无声中发生。

从难度来说，全局性的重构比一个全新业务的架构过程要难得多。重构，不只是一个架构的合理性问题。它包含了架构合理性的考量，因为我们需要知道未来在哪里，我们迭代方向在哪里。

但重构的挑战远不只是这些。这是一个集架构设计（未来架构应该是什么样的）、资源规划与调度（与新功能开发的优先级怎么排）、阶段规划（如何把大任务变小，降低内部的抵触情绪和项目风险）以及持久战所需的韧性与毅力的庞大工程。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们正式开始进入第六章：软件工程篇。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

2020 奇幻礼盒

开盒有惊喜，价值¥458起

限量发售 **¥199** 最后 800 套



© 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。

上一篇 66 | 架构老化与重构

下一篇 68 | 软件工程的宏观视角

精选留言 (7)

写留言



tt

2019-12-24

周边功能对核心系统总伤害的经验公式的本质，我总结为，不论自己设计的架构静态看起来有多么完美，各模块之间的配合有多么精妙，如果从外部给它施加一个扰动（比如需要修改某个模块），会导致整体发生剧烈的变化，那么这个架构就不是好的架构。

就像机械系统，一定要考虑公差。系统要与误差共舞，要能抵抗变化，必须处于一个动...
展开 ∨

作者回复: 是的，自我感觉良好是没有用的。只有真检验了，测量了，才能确定是不是真的好。



3



知行合一

2019-12-26

没重构过就看不懂架构课，就像不实践就没法真正理解一个道理一样。边学边实践才能真正理解老师所讲的内容，而且真的是一步一步理解的。

展开 ∨

作者回复: 是这个道理，知行合一



1



Aaron Cheung

2019-12-24

在应对架构老化这件事情上，不要轻率地选择进行全局性的重构。要把功夫花在平常，让重构在润物细无声中发生。

拿小本本记下来

展开 ∨



1



有铭

2019-12-24

一门学科的历史，就是这门学科的本身。从老师的信息科技的演进历史收获很多，期待 数据治理与业务运营体系构建 这本书早日出来

展开 ∨



1



leslie

2019-12-25

许老师的课总能引发许多反思：理解业务、理解各个环节、最小化且可扩展设计。有时觉得一个好的架构师要有3个角色：产品经理、全栈工程师/DevOps、架构师【注：其中就包含整体的软件、数据库和系统架构】。

老师的课程就是纯正的干货：课上一分钟、课下十年功。虽不知老师的《数据治理与业务运营体系构建》：不过依然非常期待老师课程推出的一天，分享给大家学习和交流...

展开 ∨



ljf10000

2019-12-24

听老师的课，有一种行云流水的感觉，非常舒服。

展开 ∨



Jxin

2019-12-24

看了一些架构书籍，但一直觉得无法入门。直到今年大量重构项目后，才感觉有点打开这扇门了。面向对象思想，设计原则，领域建模，代码规范，重构手法这些亦是架构之道。

对于重构，我遵循的是<重构 第二版>的小步快跑的模式，在每次需求迭代顺带重构周边业务。我追求的是用最小的成本（改动少，验证范围收敛）解决最大的问题（可读，可扩...

展开 ∨

