03 | 架构设计的目的

2018-05-03 李运华 来自北京

《从0开始学架构》

周二，我们聊了架构出现的历史背景和推动因素。以史为鉴，对我们了解架构设计的目的很有帮助。谈到架构设计，相信每个技术人员都是耳熟能详，但如果深入探讨一下，“为何要做架构设计？”或者“架构设计目的是什么？”类似的问题，大部分人可能从来没有思考过，或者即使有思考，也没有太明确可信的答案。

架构设计的误区

关于架构设计的目的，常见的误区有：

> 因为架构很重要，所以要做架构设计

这是一句正确的废话，架构是很重要，但架构为何重要呢？

例如：不做架构设计系统就跑不起来么？

其实不然，很多朋友尤其是经历了创业公司的朋友可能会发现，公司的初始产品可能没有架构设计，大伙撸起袖子简单讨论一下就开始编码了，根本没有正规的架构设计过程，而且也许产品开发速度还更快，上线后运行也还不错。

例如：做了架构设计就能提升开发效率么？

也不尽然，实际上有时候最简单的设计开发效率反而是最高的，架构设计毕竟需要投入时间和人力，这部分投入如果用来尽早编码，项目也许会更快。

例如：设计良好的架构能促进业务发展么？

好像有一定的道理，例如设计高性能的架构能够让用户体验更好，但反过来想，我们照抄微信的架构，业务就能达到微信的量级么？肯定不可能，不要说达到微信的量级，达到微信的 1/10 做梦都要笑醒了。

不是每个系统都要做架构设计吗

这其实是知其然不知其所以然，系统确实要做架构设计，但还是不知道为何要做架构设计，反正大家都要做架构设计，所以做架构设计肯定没错。

这样的架构师或者设计师很容易走入生搬硬套业界其他公司已有架构的歧路，美其名曰“参考”“微改进”。一旦强行引入其他公司架构后，很可能会发现架构水土不服，或者运行起来很别扭等各种情况，最后往往不得不削足适履，或者不断重构，甚至无奈推倒重来。

公司流程要求系统开发过程中必须有架构设计

与此答案类似还有因为“架构师总要做点事情”，所以要做架构设计，其实都是舍本逐末。因为流程有规定，所以要做架构设计；因为架构师要做事，所以要做架构设计，这都是很表面地看问题，并没有真正理解为何要做架构设计，而且很多需求并不一定要进行架构设计。如果认为架构师一定要找点事做，流程一定要进行架构设计，就会出现事实上不需要架构设计但形式上却继续去做架构设计，不但浪费时间和人力，还会拖慢整体的开发进度。

为了高性能、高可用、可扩展，所以要做架构设计

能够给出这个答案，说明已经有了一定的架构经历或者基础，毕竟确实很多架构设计都是冲着高性能、高可用……等“高 XX”的目标去的。

但往往持有这类观点的架构师和设计师会给项目带来巨大的灾难，这绝不是危言耸听，而是很多实际发生的事情，为什么会这样呢？因为这类架构师或者设计师不管三七二十一，不管什么系统，也不管什么业务，上来就要求“高性能、高可用、高扩展”，结果就会出现架构设计复杂无比，项目落地遥遥无期，团队天天吵翻天……等各种让人抓狂的现象，费尽九牛二虎之力将系统整上线，却发现运行不够稳定，经常出问题，出了问题很难解决，加个功能要改 1 个月……等各种继续让人抓狂的事件。

架构设计的真正目的

那架构设计的真正目的究竟是什么？

从周二与你分享的架构设计的历史背景，可以看到，整个软件技术发展的历史，其实就是一部与“复杂度”斗争的历史，架构的出现也不例外。简而言之，架构也是为了应对软件系统复杂度而提出的一个解决方案，通过回顾架构产生的历史背景和原因，我们可以基本推导出答案：**架构设计的主要目的是为了解决软件系统复杂度带来的问题**。

这个结论虽然很简洁，但却是架构设计过程中需要时刻铭记在心的一条准则，为什么这样说呢？

首先，遵循这条准则能够让“新手”架构师**心中有数，而不是一头雾水**。

新手架构师开始做架构设计的时候，心情都很激动，希望大显身手，甚至恨不得一出手就设计出世界上最牛的 XX 架构，从此走上人生巅峰，但真的面对具体的需求时，往往都会陷入一头雾水的状态：

“这么多需求，从哪里开始下手进行架构设计呢？”。

“架构设计要考虑高性能、高可用、高扩展……这么多高 XX，全部设计完成估计要 1 个月，但老大只给了 1 周时间”。

“业界 A 公司的架构是 X，B 公司的方案是 Y，两个差别比较大，该参考哪一个呢？”。

以上类似问题，如果明确了“架构设计是为了解决软件复杂度”原则后，就很好回答。

“这么多需求，从哪里开始下手进行架构设计呢？”

——通过熟悉和理解需求，识别系统复杂性所在的地方，然后针对这些复杂点进行架构设计。

“架构设计要考虑高性能、高可用、高扩展……这么多高 XX，全部设计完成估计要 1 个月，但老大只给了 1 周时间”

——架构设计并不是要面面俱到，不需要每个架构都具备高性能、高可用、高扩展等特点，而是要识别出复杂点然后有针对性地解决问题。

“业界 A 公司的架构是 X，B 公司的方案是 Y，两个差别比较大，该参考哪一个呢？”

——理解每个架构方案背后所需要解决的复杂点，然后才能对比自己的业务复杂点，参考复杂点相似的方案。

其次，遵循这条准则能够让“老鸟”架构师**有的放矢，而不是贪大求全**。

技术人员往往都希望自己能够做出最牛的东西，架构师也不例外，尤其是一些“老鸟”架构师，为了证明自己的技术牛，可能会陷入贪大求全的焦油坑而无法自拔。例如：

“我们的系统一定要做到每秒 TPS 10 万”。

“淘宝的架构是这么做的，我们也要这么做”。

“Docker 现在很流行，我们的架构应该将 Docker 应用进来”。

以上这些想法，如果拿“架构设计是为了解决软件复杂度”这个原则来衡量，就很容易判断。

“我们的系统一定要做到每秒 TPS 10 万”

——如果系统的复杂度不是在性能这部分，TPS 做到 10 万并没有什么用。

“淘宝的架构是这么做的，我们也要这么做”

——淘宝的架构是为了解决淘宝业务的复杂度而设计的，淘宝的业务复杂度并不就是我们的业务复杂度，绝大多数业务的用户量都不可能有淘宝那么大。

“Docker 现在很流行，我们的架构应该将 Docker 应用进来”

——Docker 不是万能的，只是为了解决资源重用和动态分配而设计的，如果我们的系统复杂度根本不是在这方面，引入 Docker 没有什么意义。

简单的复杂度分析案例

我来分析一个简单的案例，一起来看看如何将“架构设计的真正目的是为了解决软件系统复杂度带来的问题”这个指导思想应用到实践中。

假设我们需要设计一个大学的学生管理系统，其基本功能包括登录、注册、成绩管理、课程管理等。当我们对这样一个系统进行架构设计的时候，首先应识别其复杂度到底体现在哪里。

性能：一个学校的学生大约 1 ~ 2 万人，学生管理系统的访问频率并不高，平均每天单个学生的访问次数平均不到 1 次，因此性能这部分并不复杂，存储用 MySQL 完全能够胜任，缓存都可以不用，Web 服务器用 Nginx 绰绰有余。

可扩展性：学生管理系统的功能比较稳定，可扩展的空间并不大，因此可扩展性也不复杂。

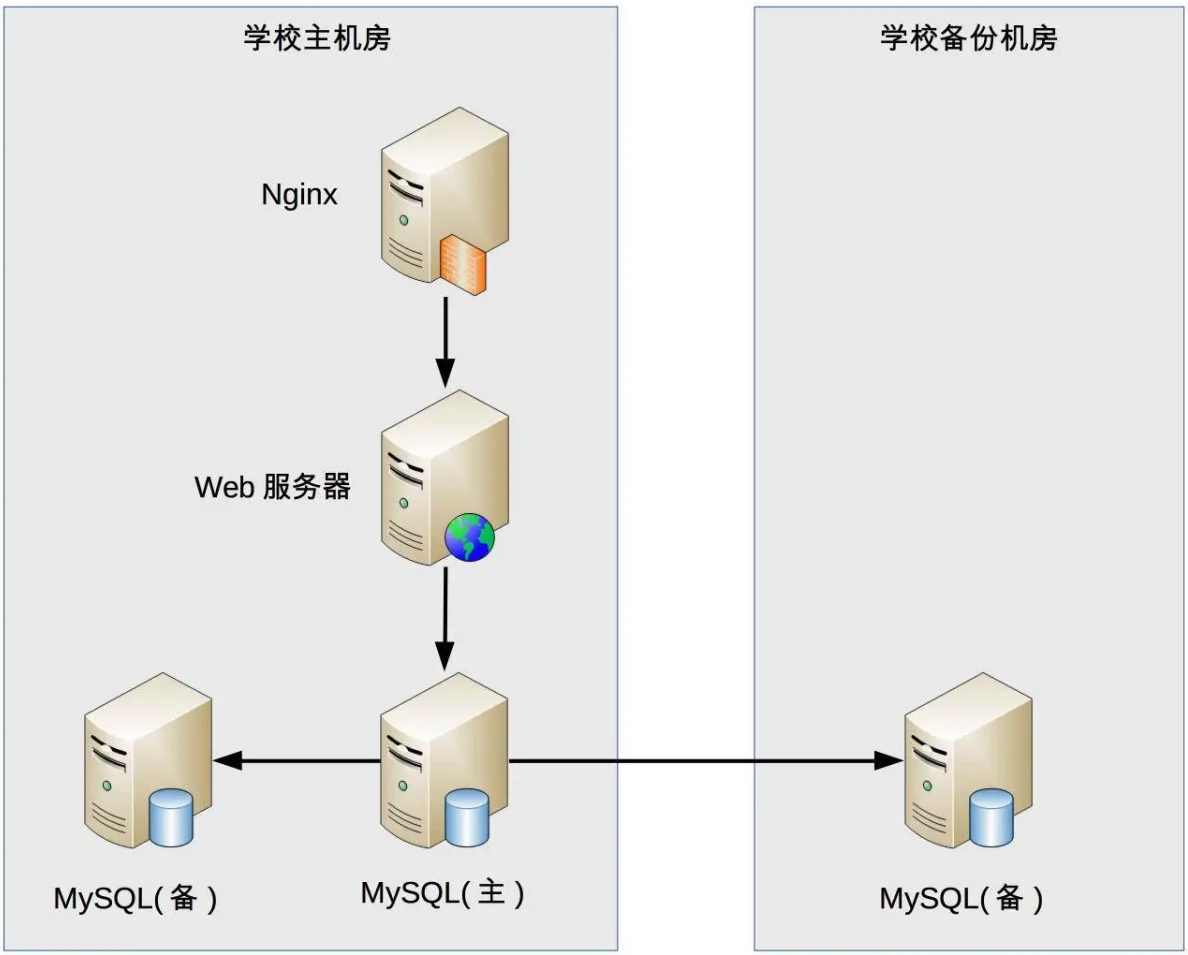
高可用：学生管理系统即使宕机 2 小时，对学生管理工作影响并不大，因此可以不做负载均衡，更不用考虑异地多活这类复杂的方案了。但是，如果学生的数据全部丢失，修复是非常麻烦的，只能靠人工逐条修复，这个很难接受，因此需要考虑存储高可靠，这里就有点复杂了。

我们需要考虑多种异常情况：机器故障、机房故障，针对机器故障，我们需要设计 MySQL 同机房主备方案；针对机房故障，我们需要设计 MySQL 跨机房同步方案。

安全性：学生管理系统存储的信息有一定的隐私性，例如学生的家庭情况，但并不是和金融相关的，也不包含强隐私（例如玉照、情感）的信息，因此安全性方面只要做 3 个事情就基本满足要求了：Nginx 提供 ACL 控制、用户账号密码管理、数据库访问权限控制。

成本：由于系统很简单，基本上几台服务器就能够搞定，对于一所大学来说完全不是问题，可以无需太多关注。

还有其他方面，如果有兴趣，你可以自行尝试去分析。通过我上面的分析，可以看到这个方案的主要复杂性体现在存储可靠性上，需要保证异常的时候，不要丢失所有数据即可（丢失几个或者几十个学生的信息问题不大），对应的架构如下：



学生管理系统虽然简单，但麻雀虽小五脏俱全，基本上能涵盖软件系统复杂度分析的各个方面，而且绝大部分技术人员都曾经自己设计或者接触过类似的系统，如果将这个案例和自己的经验对比，相信会有更多的收获。

小结

今天我为你分析了架构设计的误区，结合周二讲的架构设计的历史背景，给出架构设计的主要目的是为了解决软件系统复杂度带来的问题，并分析了一个简单复杂度的案例，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧。请按照“架构设计的主要目的是为了解决软件复杂度带来的问题”这个指导思想来分析一下你目前的业务系统架构，看看是否和你当时分析的结果一样？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

最后给你推荐一个课程，极客时间新上线了《Java 核心技术 36 讲》，由 Oracle 首席工程师杨晓峰老师给你精讲大厂 Java 面试题，帮你构建 Java 知识体系，你可以点击下方图片进入课程。

精选留言 (268)

公号-技术夜未眠 酥

2018-05-03

架构是为了应对软件系统复杂度而提出的一个解决方案。个人感悟是:架构即(重要)决策，是在一个有约束的盒子里去求解或接近最合适的解。这个有约束的盒子是团队经验、成本、资源、进度、业务所处阶段等所编织、掺杂在一起的综合体(人，财，物，时间，事情等)。架构无优劣，但是存在恰当的架构用在合适的软件系统中，而这些就是决策的结果。

需求驱动架构。在分析设计阶段，需要考虑一定的人力与时间去"跳出代码，总揽全局"，为业务和IT技术之间搭建一座"桥梁"。

架构设计处于软件研制的前期，一方面，越是前期，如有问题，就能够越早发现，修改的代价也就越低；另外一方面，也意味着，软件实施后期若有架构上的修改，也需要付出更多的代价。

今日得到:
1 架构是为了应对软件系统复杂度而提出的一个解决方案。
2 架构即(重要)决策
3 需求驱动架构，架起分析与设计实现的桥梁
4 架构与开发成本的关系

作者回复: 收下我的膝盖，大神写个心得就已经把我后面的内容剧透了，6666👍👍😃

共 4 条评论 > 654

李志博
2018-05-03

今天读完文章，思考了下我们系统的复杂点，我们系统是一个承上启下的系统，根本没自己的表，所有数据都是调第三方接口取，然后汇总聚合给前端浏览器，突然明白最近老大为什么要搞es去异步聚合第三方数据了，这样以往我们需要调第三方多次接口取的数据，以后调自己es查询一次就可以了，这样性能更高，且逻辑更简单，更容易维护，以往优化这种性能问题的方式，就是多线程，然而多线程也是要消耗资源调，而且代码反而更难以理解，原来最好的优化方式不是把串行变并行，而是把串行干的多个事的数量去减少，首先要根据系统复杂点想到合适的解决方案，其次才是用什么优秀的框架叫代码更牛逼一点，否则一开始就算错的

作者回复: 你已经参透天机👍👍

共 10 条评论 > 121

gevin
2018-05-18

我觉得做软件架构是为两件事服务的：业务架构和业务量级，这应该算是“软件系统复杂度带来问题”的具体化吧。

业务架构和业务量级都是从每个具体项目的实际应用场景中提炼出来的。

业务架构是对业务需求的提炼和抽象，开发软件必须要满足业务需求，否则就是空中楼阁。软件系统业务上的复杂度问题，可以从业务架构的角度切分工作交界面来解决。设计软件架构，首先是要保证能和业务架构对的上，这也是从业务逻辑转向代码逻辑的过程，所以软件架构的设计为开发指明了方向。另外架构设计也为接下来的开发工作分工奠定了基础。

业务量级表现在存储能力、吞吐能力和容错能力等，主要是软件运维期业务的复杂度。做软件架构设计，是要保证软件有能力托起它在业务量级上的要求的，如果软件到运行使用期废了，前面所有的工作都付诸东流了。不同的业务量级，对应的软件的架构复杂度是不同的，所以对于不同的项目，业务量级不同，架构设计也不同。

做业务架构必须与其面向的实际应用场景相匹配，由于每个产品或项目的业务场景均有所不同，所以每次做新的软件开发前，必须先设计软件架构，试图不经分析直接套用先前的架构方案，十有八九会让当前的系统在某个点上报出大问题导致推翻重来，更不要说直接拿别人的现成架构方案了。

所以每个软件在开发前，都要结合自己的应用场景设计适合自身的软件架构，现成的架构方案只能借鉴，不能直接套用。

另外，由于业务架构和业务量级也会不断调整或长大，软件架构也不是一劳永逸的，会随业务不断调整。

作者回复: 赞，写的很好，对我的专栏内容是很好的补充

共 6 条评论 >

79

霍潘

2018-05-03

这篇内容完全符合我们公司目前现状，为了架构而架构。原本3个人负责的nodejs系统现在改用JAVA开发，光架构师招了快7，8个。不到2000人使用的系统没有一天不出点小故障，开发过程更是痛苦，三天两头的开发环境跑不起来。说是微服务，不过是把原来单体结构按功能拆分，按层级拆分，本来就拆分的够多了，为了开发要启动很多服务。每个新来的架构师都要引入一套自己的架构，结果可想而知

作者回复: 8个架构师？太夸张了，一个架构师一般可以支撑20人以上的开发团队，你们的架构师莫非是传说中的PPT架构师？😃

共 9 条评论 >

65

✎﹏ℳ𝒟𝓅𝓊ℓ丶陌
2018-05-05

架构就是。当你想拉屎的时候首先得去找厕所。选择一个什么样的厕所。
1.距离选择。如果太远可能就漏出来了
2.成本选择。如果拉屎太贵也不合适
3.性能选择。好的厕所可以拉的心情舒畅

作者回复: 角度刁钻😂😂

共 5 条评论 >　　59

懒人闲思
2018-05-03

伟大导师马克思说过：主要矛盾处于支配地位，次要矛盾。。。也会有影响，架构设计就是提前预判可能遇到的矛盾，用最小的代价解决它

作者回复: 用马哲来指导架构设计，好像也有道理😃

57

suke
2018-05-03

老师好，我们目前在做内部的对象存储中控台，满足的需求有以下几个:
1.方便内部非开发人员做数据存储
2.提供对象存储的图形化界面
3.监控整个存储集群的存储相关的指标，例如存储量，下行流量、api调用等等

这几个需求也算是我结合公司的现有业务状况，以及对象存储的特性，自己总结的。
系统上线以后，我预计的使用情况如下:
1.大的天使客户都是一些职能的业务部门，由存储集群提供底层的存储接口，通过中控台进行监控，图形化操作
2.其他零散人员，可存储一些工作上的数据

目前我只是做了个demo出来，整体架构:nginx做前端的负载，springboot提供服务，mysql存储元数据，redis做消息队列，以及缓存，数据通过接口传到存储集群
这个架构也没做太多考虑，就像老师您说的情况一样，只不过公司其他的系统也差不多都这样。

对象存储中控台的业务复杂度我思考有以下几个:
1.提供高可用，高性能的上传下载功能
2.提供bucket 以及对象的查询功能
3.提供低延迟的监控统计功能
4.并发访问量个人感觉都集中在存储集群的接口调用，中控台这边并不高，但是一定要做到系统的高可用，保证内部职工正常的工作使用，同时mysql也要做到主备和异地多活，以及redis的主备

老师这样的分析可以么，针对这样的业务，现有的架构有问题么

作者回复: 你分析的主要还是功能点，应该更进一步，分析这些功能点后的复杂度，而且要尽量量化。例如：
1. 高可用和高性能：到底要多高，为什么要高性能高可用?
2. 低延迟：到底多低？秒级和分钟级和小时级，复杂度差很大，秒级你可能要用流式计算，分钟级用后台计算可能就可以了，小时级直接用数据库就可以了
3. 系统高可用具体达到什么水平？是1分钟都不能停，还是可以停1个小时？是数据绝对不能丢，还是可以丢一部分数据然后其它方式修复?

对于高性能和高可用，千万不能说越高越好，一定要结合业务，例如，绝大部分内部系统的宕机容忍时间可以是一个小时。

共 3 条评论 >

47

张玮(大圣)
2018-05-03

1. 问答引导式由浅入深来讲解很赞

2. 一个简单的例子能说明相关问题也很赞，就是这样，大家有时不需要大而全的大公司大例子，反手一个赞，加油，运华兄

作者回复: 一起加油，大圣👊👊

43

加多
2018-05-03

赞，就是不要过度设计，架构设计要贴合业务，找到适合能解决业务复杂度的设计，才是好的架构设计

27

晴天

2018-05-04

我不是架构师，但是有一颗这样的梦想。小时不识月的时候，以为架构就是ssm，就是几种中间件的堆砌，简单粗暴，我上我也行的心态，后来接触的越多，业务复杂了，又开始忧来其如何了，为什么要这么设计，为什么用这个，接触概念也越来越高大上，开始如作者所说的大牛都该展现自己的技术，就是要上高性能高并发等等，就是要往复杂了去设计，盲目的去崇拜复杂的设计，而恰恰忽略了设计的本质是为了解决一定范围内的问题也就是软件复杂度，其实没必要那么疯狂，也没那么简单。适合的才是最完美的，脱离不了业务，也离不开现实，譬如成本控制，进度控制等等。所以我应该静下心来，认真的去学习那些优良的设计的背后，如何在选择上驾轻就熟，如何在这个规则里游刃有余……

作者回复: 现在的你就是曾经的我，一起加油👍👍👍

共 2 条评论>

25

@漆~心endless

2018-05-03

一切脱离业务的架构设计都是耍流氓

24

Will

2018-05-04

架构是解决复杂度的问题。那么复杂度有很多不同的来源，比如人（不同的代码风格，不同的编程习惯），比如业务，比如技术。那么架构不可能面面俱到的解决所有问题，必须要分析出所面对的一个或几个关键的问题，这样架构的设计就能有落脚点，而且问题解决也不会有大的冲突。

架构设计在发展的不同阶段面临不同的问题，例如我们公司刚开始就做了业务拆分，后端是多个服务，前端一个站点，并且提供了一个服务互相调用的公共代理，现在主站越来越庞大，涵盖的业务越来越多，所以要做业务拆分，公共代理所包含的服务也越来越多，也要进行拆分。另外一个业务要调用多个服务，如何去监控调用链的完整性，这也需要解决。

所以架构本身在不同阶段集中解决几个最主要的问题，之后随着业务，技术，问题的不断变化，架构的重点也在不断调整。

作者回复: 你已经参透天机，后面会讲到你说的内容。

22

重剑
2018-05-18

感觉自己好笨，看了好几章，说不出什么心得体悟😅，看各位大神踊跃发言好羡慕。

作者回复: 学完你也可以吹水了😃

18

罗烽
2018-05-03

架构设计主要是为了解决软件复杂度的问题：
现有业务复杂度分析：
性能： 提供外部接口，对性能有一定的要求，对对存储也有一定要求，但访问量每天不高，1万~2万，mysql+缓存，还是有必要的，量不高，但要快
可扩展性：因为需要经常对接其他支付接口，所以这里的可扩展性有一定要求
高可用：这是重点，因为是提供给商场使用的支付接口，哪怕一分钟宕机都是不行的，针对高可用方案，我现在只能想到，在不同机器上部署多套服务，然后用nginx做负载。不过晚上没有交易，可以停机的。存储数据都是订单，这个很重要，因为每天需要查账，不过已经使用了阿里云的rds主备方案，这里也不用担心
安全性：现在接口都是用sign验签，而秘钥都是独立的，所以这里也没有问题了

综合来说，我们对高可用，稳定性是要求最多的，这是这里的复杂度

作者回复: 1. 每天1~2万的话，性能要求其实很低，性能都是按秒计算的，所以缓存不一定需要
2. 分析的很好，白天要求高，晚上可以停机，这样的背景可以设计很多有趣的方案，例如每天晚上定时重启一次，晚上批处理对账等

共 3 条评论>

16

何鹏
2018-05-04

我要提问，容我问个题外问题，问题如下。
背景:我工作3年多，最近在学习uml，但是感觉程序设计帮助并不大。程序设计和编程还是靠

经验和加班加出来的。
所以问题是，方法论这个东西在程序设计方面真的作用大吗？编程是不是还就是靠经验和加班？
不知道版主哥对方法论怎么看的，并且你用uml吗？
补充，uml我才用了3个月，只用它开发了3个功能，所以也可能是我资历太浅，没掌握要领和精髓。

作者回复: 我的第一本书《面向对象葵花宝典》详细阐述了面向对象全流程的设计方法以及uml的应用场景，uml和架构设计一样，都是有具体应用场景的。

方法论非常有用，编程发展几十年了，方法论就是各路大牛，各种场景的经验总结，你再怎么加班，都不可能把所有的坑都踩一遍，站在巨人的肩膀上，基于前辈们的经验，能够减少自己走很多弯路。

我的架构设计方法论是积累和实践很多年总结出来的，也是基于以前的各种理论和经验提炼的，不可能完全靠自己摸索出来的。

共 4 条评论 >　　15

Snake
2018-05-19

一直有个疑问，一个项目已经在持续推进，而当前的每个迭代只是不停地增加新的业务功能，一般不涉及任何底层设施的变动，此时架构师一般都应该做些什么？是需要针对每个迭代的需求做业务架构设计，然后产出对应的代码架构图吗？如业务的活动图和类图等

作者回复: 这个不需要架构师设计，开发人员设计就可以，架构师需要关注项目架构是否会因为开发新业务而引入新的复杂度

共 3 条评论 >　　13

天使
2018-05-03

需要一个解决方案降低微服务的实施成本，创业公司面临的主要是需求变化剧烈的问题。微服务可以将变化局限在某个区域，不至于影响全局，所以最好一开始设计的时候就这样设计。我经历了三家创业公司，第一家大泥球架构，半年后就发现改动起来各种痛苦，牵一发动全身有时。第二家第三家上来就是微服务，基本没有这个问题。而且需求不同，可以采用完全不同的技术栈。相比第一家创业公司，人更少了，问题更少了，质量也更高了。另外，没有 devops 搞微服务就是天坑...

作者回复: 微服务有专门章节阐述

11

Mario
2018-05-03

架构设计的目的是什么?
这是个好问题，值得每个架构师和想成为架构师的技术人经常思考。
为了解决软件复杂性，是经典答案;
个人认为架构设计的目的有:
制定群解决业务问题复杂性的解决方案规范，包括核心业务流、扩展点位置、流程重组规范、服务编排方 法、等等，简而言之规范是架构的灵魂，方　法 '论（比如DDD）是架构的指导意见，统一技术认知和屏蔽横向业务影响是架构核心价值。
拙见，请大家指正

作者回复: 业务复杂度只是其中一种复杂度，并不是每个软件系统都有业务复杂度问题

9

MavenTalker
2018-05-22

有时候往往会为了一项牛逼的技术，想着法的引入到架构中去验证，这其实就是拿着锤子找钉子，忽略了技术实际的应用场景。

不同的市场时机，不同的团队素质，也会衍生出不同的架构设计，没有对错，只有合不合适。

作者回复: 非常正确，所以架构师的工作是一项头疼的工作，考虑东西很多

8

小伟
2019-02-28

今日笔记:
1. 架构，架者，实体也，构者，行为也。架即数据，构即服务。
2.架构分业务架构和技术架构，业务架构优先(决定)技术架构。脱离业务的技术架构不是好架构，不懂业务的架构师不是好架构师。
3.技术架构解决业务服务的复杂性问题。

4.架构可从性能、可用、可靠、安全、维护、扩展、人力、成本、团队经验、推动力等维度切分业务场景的复杂度，按复杂度高低顺序排列，找到解决该纬度复杂性的方案，通盘评估各解决方案间的冲突及可行性，最终得到一个可落地的架构方案。


7