《聊聊架构》——12条识墨笔记

Author: 王概凯

Publisher: 电子工业出版社

Publish Date: 2017-5

1

在软件生命周期拆分树中,软件架构师本身的业务也会发生架构拆分,拆分出不同位置的架构师就有了各自不同的架构范围:有负责部署架构的架构师,如系统架构师;有负责数据架构分拆的架构师,如数据架构师;有负责应用代码编写架构的架构师,如应用架构师,等等。还有很多其他不同类型的软件架构师种类多少取决于架构师业务分工的细化程度,这里不再一一列举。

架构师的种类。可能我当初想做的就是应用架构师

Page 67 · 2017-07-02

2

而架构师则不一样,架构师很冷静、很平等地对待所有的技术。对于软件架构师来说,他们要深刻地领会软件的开发生命周期和软件的运行生命周期,以及业务的生命周期,把它们合理地结合在一起,并能够通过软件的快速增长来支撑和顺应业务的增长。这些是软件架构师的核心能力。为达到这些目标,无论是新的还是旧的技术、先进的还是落后的技术,只要场景合适,他们就会采纳。因为架构师是技术的使用者,他们把技术当作解决增长问题的手段和工具,而不会被技术束缚住。

软件架构师的核心能力，也就是解决问题。

Page 69 · 2017-07-02

3

技术本身是没有好坏的。比如最早的计算机软件是单机运行的,逐渐发生了架构拆分,形成了Cs的架构,也就是 Client / server,随着互联网技术的发展。变成了Bs,也就是 Bnowserfserve。其实是换汤不换药, Browser还是一个

Ciet但是因为 Browser的普适性,其实就是又做了一个架构拆分,成了一个通用的Cint,做到了很多Cs时代做不到的事情。互联网时代的技术人员对C架构视得一塌糊涂。到了App时代,又回到了Cs时代的做法,因为HTML5响应慢无法适应手机客户端。随着手机端带宽的逐步改善,以及手机端计算能力的逐步提升,慢慢可能又会走向Bs模式。所以说,随看人们问题的不断变化。会有不同的技术周期出现,技术有自己的生命周期。所谓"三十年河东,三十年河百",没有水远最好的技术,也没有水远最差的技术,而问题总是在不断发生变化的对于这一点,架构师要有清醒的认识。

Page 70 · 2017-07-02

---

4 架构师是拆分生命周期，技术人员是实现生命周期。

比如最简单的 一个公司的架构师说自己的编码需要前后端分离 其实这也是架构 前端生命周期和后端生命周期 由不同的人员来实现

2017-07-02

---

5 在软件设计开发的过程中经常会看到,很多所谓的架构讨论实际上只是在讨论某些技术的技术讨论。在很多人看来,特别是软件工程师,架构和技术实际上是等同的。多学会了几种技术,就觉得可以做架构师了,或者学会的技术越多,就觉得自己的架构水平越高。对于别的架构师当然也采用同样的标准来评判。要知道,任何技术都是为了解决某种问题面存在的,学会了很多技术,并不代表能够利用这些技术来解决问题。学会的技术的多少,所带来的差别只是解决问题的手段多了些而已。但是手段多了就一定是好事吗?学会的技术越多,很多时候越不知道采用哪种技术更好,所谓"乱花渐欲迷人眼"。

并不是掌握的技术越多，架构水平就会增强

2017-07-02

---

6 还是自己重新实现一个呢?重新实现一个,就是很多人所谓的重新发明轮子。下面就对几种不同的情况分别讨论一下:大,度十应,了示(1)当技术所要解决的

问题和拆分出来要解决的问题完全匹配时,这是最完美的。比如需要提供web访问的服务(Service),而很多Mvc的框架(Framework)就是解决这个问题的,可以很好地满足这一点。这种情况如果非要重新实现一个,很有可能是重新发明轮子。(2)当技术所提供的能力远远超过需要解决的问题时,往往掌握技术和维护技术就会成为负担。越复杂的技术,成本越高,带来的问题也会越多。如果自己实现一个仅仅是解决当前问题的技术,可能成本反而更低。(3)当技术所提供的能力和我们所要解决的问题部分匹配时,要判断是否要采用,最终还是要看成本。比如当人们需要一个锤子的时候,手边正好没有,但是却有一只高跟鞋,勉强也可以用来替代锤子来解决问题。但是长期这么用不划算,因为高跟鞋的价格比锤子高很多,耐用性却差很多,而维护成本也高很多。所以,在架构拆分的基础之上,识别并平衡技术的能力,也是架构师所要具备的能力之一。考虑的主要因素是长期的成本和收益。

Page 78 · 2017-07-03

---

7 为什么会这样呢?这就是因为业务人员和软件开发工程师分体的缘故。业务员是出资方,有全面的业务话语权,可是软件开发的核心是软件工程师,对件有绝对的话语权。两个话语权分散了,权责不对等,导致软件工程师没有办法真正地理解业务,开发出的软件自然就长得歪歪扭扭变形了。而让业务方完全放手是不太可能的,这就是为什么传统企业转型这么难,要给自己“捅一刀”才行。

和龙哥说的有点相似

2017-07-03

---

8 换句话说,软件架构把软件看成虚拟的人,实际上就是虚拟人的组织架构。架构拆分的原则首先来源于业务自身的组织架构,使得软件架构保持和业务组织架构的匹配关系;其次来源于软件开发团队自身的组织架构;最后来源于用户的流量对软件本身的冲击。如果软件开发团队的组织架构和业务的组织架构一致,这就是损耗最小的方式,软件的架构也会更简单。

Page 99 · 2017-07-12

---

9 服务作为一个通道的含义是什么呢?通道的意思就是只负责调用业务逻辑自不包含业务逻辑。比如用户到银行存钱,柜员接到用户的钱后既不会把钱带回家,也不会给用户利息,用户同样也不会找柜员要本金和利息。柜员只是用户和银行之间的二传手,而不是银行业务逻辑的拥有者,这就是通道的意思。这也意软件工程师所写的服务代码中是不能够包含业务逻辑的。从上图可以看出,服务为完成用户访间生命周期,承担的责任包括了组合业务和存储这两个,还要提供给用户访问。这部分的代码任务太多,代码人员的负担比较重,也容易引起服务的代码失控。为解决这个问题,需要把用户访问生命周期再展开分析一下。用户要完成访问业务逻辑生命周期,需要做如下事情:

这里的服务层就像是api中的控制器方法中，主要为调用并组合业务逻辑，但mvc层c层做了服务和业务逻辑部分，没有拆分。

2017-07-13

---

10 再承上例,订单(Oder)、用户、账户(Account)和商品(Commodity)这几个业务生命周期完全可以放在同一个软件中实现,都通过服务来暴露出去,对外界提供服务。此时往往业务方也是一体的,并没有发生业务拆分。业务没有拆分,为何这些生命周期却拆分出来了呢?这些生命周期的拆分其实是人类有历史以来经历分工形成的行之有效的组织方式,即模式。每个企业成立后,都可以利用这些拆分的方式来组织自己的业务。当业务逐渐的长大,为了提高效率,用户、账户和商品会逐渐地切分出来,形成独立的生命周期,变成不同的部门单独负责。也就是说业务部门会发生架构的切分。此时几个部门同时来对一个软件提需求,部门之间的沟通就变成了问题,软件发布上线排队也成了问题,软件的开发生命周期的效率需要提升。另一方面,随着用户、账户和商品的增长,订单也会逐渐增长,而软件越来越庞大,一个软件的运行效率也成了问题,软件的运行生命周期的效率需要提升。因此软件必须

Page 114 · 2017-07-15

---

11 要进行拆分才能够提高软件的开发生命周期的效率,同时软件的拆分也会导致软件开发部门的拆分。软件的拆分必须要和业务的拆分对应起来,此时就可看出业务生命周期分析的好处。很多人总是拆不好的原因,就是忽视了业

务生命周期的分析。因为软件的核心是模拟业务,而业务代码又是按照业务的生命周期组织的,软件拆分的目的就是把软件的业务生命周期代码进行生命周期拆分。不仅仅代码内部可以进行拆分,还可以直接把某些业务生命周期的代码拆分到另外一个软件中,并把相应的服务代码、管理者代码和存储代码一起拆分过去这个拆分方式就形成了新的软件,而对原软件的影响仅仅是对被拆分出去的业务调用方式发生了变化而已,从本地调用变成了服务调用。比如把用户生命周期切分到另一软件,形成用户软件之前,用户服务(Orderservice)代码调用用户管理者(Usermanaget)代码来访问用户的生命周期活动;在用户切分到用户软件之后,用户服务代码调用用户软件的服务来访问用户的生命周期活动。这个时候软件就形成了一个树状架构。同时,软件的服务就产生了用户软件切分出来之后,相对应的用户软件开发团队也要单独切分出来,直接面对用户业务团队。这个时候业务和软件开发团队的沟通是一对一的,不会有太大问题。软件因此就开始长大了,软件的开发团队也长大了,业务、软件开发团队和软件的效率都得到了提升从以上的软件拆分可以看出,业务生命周期的分析既是软件拆分的大前提又是架构的基因。失去了这一点,软件架构就会出现问题,软件开发团队的组织架构也会出现问题,会导致产生大量不必要的额外沟通来弥补。此时软件开发团队的组织就像人一样,人的身体失衡就开始生病了。一旦分析清楚了业务的生命周期,软件的运行架构和软件开发的组织可以很容易地进行拆分调整,软件和软件开发团队都可以顺利的拆分长大。

主要体现了软件拆分的思路。原先对软件拆分没有概念。

Page 115 · 2017-07-15

---

12 随着软件工程师数量的增多,他们中越来越多的人面临着同样的问题:如同管理硬件的资源?因此,一部分软件工程师转而去专门开发软件来管理硬件的源。形成了软件的分工,于是操作系统逐渐地出现,把硬件资源使一管理起来并给开发人员提供访问资源的接口。用户访问路径发生了架构拆分,在目标件和硬件之间形成了操作系统生命周期,变成了:用户→目标软件一操作系统一硬件。操作系统的出现,让软件工程师不再需要直接和硬件打交道,软件开发的门槛下降了,开发软件也变得更加容易计算机在早期还是非常昂贵的,贵的东西通过提高使用率可以降低成本。随着网络的兴起,软件被拆分为两部

分,一部分在服务端运行,一部分在客户端运行,客户端可以用非常简单的终端连上服务器。使得同一台计算机可以被多人同时共享使用。用户对软件的访问通道进一步发生了架构分拆,增加了客户端软件、网络等生命周期。这通常被称为C架构(Clientserver)因此用户的访问路径变成:用户→客户端→网络→目标软件一→操作系统一硬件。随着服务端编程复杂度地上升,服务端软件发生了架构分拆,软件分拆为应用服务器和运行在应用服务器上的软件,应用服务器的生命周期就出现了。应用服务器,顾名思义,就是为应用提供一个容器,负责生成端口,和用户的的请求交互,并把请求传递给应用服务器所容纳的软件来处理业务。因此用户的访问路径变成:用户→客户端→网络一→目标软件→应用服务器一操作系统→硬件。随着云服务的兴起,硬件的利用率问题慢慢浮上台面,虚拟化、容器化也成为了一个方向,在操作系统前形成了一个新的架构分拆,形成了容器的生命周期用户的访问路径变成:用户→客户端→网络→目标软件→应用服务器→容器一操作系统→硬件,或者:用户→客户端→网络→目标软件→应用服务器→操作系统→虚拟化→硬件。从这个发展路径上来看,用户的访问生命周期在不断地被拆分,分工在不断地增加。同时软件本身也在不断地拆分,从一个软件做所有的事情,拆分为软件和操作系统,业务软件不再需要直接和硬件打交道了,和硬件打交道变成了其他软件的业务,比如操作系统等。接着软件拆分出了应用服务器,软件不用再直接和网络打交道了。随着MVC框架的兴起,软件得以真正的专注于所服务的业务需求。软件访问路径的拆分导致软件的分工越来越多,也越来越细,不同的分工

用户访问周期的流程

2017-07-21