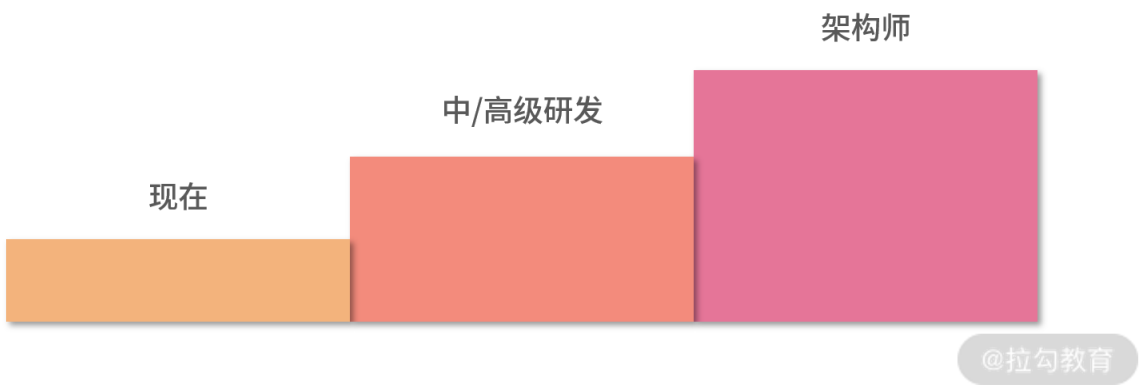
二

# 01 研发工程师想提升面试竞争力，该具备这三个技术认知

今天是咱们第一节课，我想和你聊一聊：研发工程师想提升面试竞争力，需要具备的三个技术认知是什么。

在我参加研发同学述职的时候，发现几乎每个人最后都会用一页 PPT 来规划自己的未来职业发展，比如：我目前处在初级研发工程师阶段，下一步要成为中高级研发工程师，最终要成为一名研发架构师。




但是在进一步地追问后，大多数研发同学对自身技术发展的认知，仅停留在学习了哪种新的技术，掌握了哪种新的开发框架，觉得这样就能把技术做好，就能成为架构师。

可是现实情况是：**你觉得技术满足应聘部门的要求，可还是面不到想要的职位。**这其实与技术认知不足有很大关系，你达不到一个高级研发或者是架构师该有的思维 层次，在面试时，自然很难讲出自己的技术价值与亮点，就会影响面试竞争力。

而今天这一讲，我会从架构设计认知、分析问题的认知、能力边界认知三个角度出发，讲解研发工程师如何提高自己的技术认知，在面试的过程中更加体现价值和竞争力，进而获得满意的 Offer。

## 对架构设计的认知

我在开篇词中提到，面试官通常会在考察完候选人基础技术能力之后，再问一些关于系统架构设计上的问题，这时如果你回答得比较好，很容易得到面试官的认可，也会掩盖个别技术问题上回答的不足。

但实际上，很多研发同学对架构设计的掌握和理解是欠缺经验的，系统设计问题只能回答出表层的技术名词，落地没有实际经验，拔高没有理论支撑。那你怎么回答面试中的架构设计问题呢？

**关于架构设计的问题，一定要立足于点、连接成线、扩散成面**，用这样的思路回答才能让面试官满意。下面我就通过一个例子，来帮你理解什么是回答架构设计问题该有的认知。

### 例子

我曾面试过一名研发工程师，他在介绍过往经历时，称自己在重构一个负责交易流程的系统时，将其拆分成报价系统、促销系统，以及订单系统，而当时他们只有两个人负责交易系统的开发工作。

针对他的经历，我的问题是：你们只有两个人负责这个交易系统，为什么还要做系统架构拆分？而且拆分之后会带来其他的复杂度，你是怎么考虑的？

系统拆分的架构设计问题，在面试中很常见，候选人给出了四个层面的回答。

> **从订单系统层面来看**，由于交易流程中的订单系统相对来说业务稳定，不存在很多的迭代需求，如果耦合到整个交易系统中，在其他功能发布上线的时候会影响订单系统，比如订单中心的稳定性。基于这样的考虑，需要拆分出一个独立的子系统。
>
> **从促销系统层面来看**，由于促销系统是交易流程中的非核心系统，出于保障交易流程稳定性的考虑，将促销系统单独拆分出来，在发生异常的时候能让促销系统具有可降级的能力。
>
> **从报价系统层面来看**，报价是业务交易流程中最为复杂和灵活的系统，出于专业化和快速迭代的考虑，拆分出一个独立的报价系统，目的就是为了快速响应需求的变化。
>
> **从复杂度评估层面来看**，系统拆分虽然会导致系统交互更加复杂，但在规范了 API 的格式定义和调用方式后，系统的复杂度可以维持在可控的范围内。

这样的回答很好地表达了应聘者对系统设计的思考与理解。因为他说出了原有系统中关于订单、促销和报价功能耦合在一起带来的实际问题，这是**立足于点**，又从交易流程的角度做系统设计串联起三个系统的拆分逻辑，**这是连接成线**，最后从复杂度和成本考量的方向夯实了设计的原则，**这是扩展成面**。

### 案例分析

如果你是这名应聘者，会怎么回答呢？很多研发同学一提到架构设计就说要做拆分，将一个系统拆分成两个系统，将一个服务拆分成两个服务，甚至觉得架构就是做系统拆分，但其实并不理解拆分背后的深层原因，所以往往只能回答得比较表面，无法深入背后的底层设计逻辑，那这个问题的底层逻辑到底是什么呢？有这样四点。

- **为什么做架构拆分**？通常最直接目的就是做系统之间解耦、子系统之间解耦，或模块之间的解耦。
- **为什么要做系统解耦**？系统解耦后，使得原本错综复杂的调用逻辑能有序地分布到各个独立的系统中，从而使得拆封后的各个系统职责更单一，功能更为内聚。
- **为什么要做职责单一**？因为职责单一的系统功能逻辑的迭代速度会更快，会提高研发团队响应业务需求的速度，也就是提高了团队的开发效率。
- **为什么要关注开发效率**？研发迭代效率的提升是任何一家公司在业务发展期间都最为关注的问题，**所以从某种程度上看，架构拆分是系统提效最直接的手段。**

所以，**架构拆分其实是管理在技术上提效的一种手段**，认识到这一点后，就不难理解为什么很多架构师在做系统架构时，会做系统设计上的拆分，甚至认为架构的本质就是拆分了。

## 对分析问题的认知

在实际工作中，技术人员在做系统设计时需要与公司或部门的战略定位对齐，才能让你的技术有价值。因为对于系统技术架构升级的问题，业务方、管理者和技术人员的关注点是不同的。

- 业务方的诉求是在技术升级后，系统有能力迭代功能来满足市场的要求，所以**关注点在系统能力。**
- 管理者的诉求是在技术升级后，系统研发团队的开发效能得到提升，所以**关注点在人效管理。**
- 作为技术人员的你，需要找到自己做系统设计的立足点，来满足不同人对技术的诉求，而这个立足点通常就是**系统设计原则。**

所以你应该认识到，系统的设计原则不是乱提出来的，而是**针对系统现阶段业务发展带来的主要矛盾提出，才会更有价值且被认可。**

### 例子

之前我做过一个对原有老系统进行架构改造的系统设计，当时的背景是这样的。

早期，业务发展比较简单，团队规模也不是很大，单体系统可以支撑业务的早期规模，但当业务不断发展，团队规模越来越大时，之前的一个业务团队逐渐发展成了多个业务团队，这时每个业务团队都会提出自己的功能需求。

然而，系统现状仍然是单体架构，研发同学都在同一个系统里进行开发，使得系统逻辑复杂，代码耦合，功能迭代和交付变得非常缓慢，牵一发而动全身，研发同学都不敢轻易修改代码。

这个时期系统的主要矛盾就变成了：多人协作进行复杂业务，导致速度缓慢，但业务需求又快速迭代。说白了，就是**研发效率不能匹配业务发展的速度，并且单靠加人不能解决问题。**

对于这样的一个系统，**此阶段的系统架构核心原则就不能随便定义为要保证高性能和高可用。**

那么应该怎么做呢？针对这样的问题，我们需要对原有系统进行合理的系统边界拆分，让研发人员有能力提速，来快速响应需求变化，这就要求架构师对业务领域和团队人员有足够的了解。

类似这样的情况也是面试中经常出现的考题，比如面试官在问你历史项目经历的时候，要重点关注你是如何解决系统核心问题的，所以不要一张口就是高性能、高可用，这会让有经验的面试官觉得你很初级。

#### 案例分析

面试中，研发人员在回答系统设计问题的时候，要根据系统所处阶段的主要矛盾来回答架构设计问题，在 20 世纪 60 年代，《人月神话》的作者就分析，软件复杂性来源于两点：本质复杂度和偶然复杂度。开发工具、开发框架、开发模式，以及高性能和高可用这些仅是偶然复杂性，**架构最重要的是要解决本质复杂性，这包括人的复杂性和业务的复杂性。**

技术是静态的，业务和用户是变化的，具体问题要从具体的业务领域出发。这时有人可能会说，我只想做技术，不想做业务，然而你会慢慢发现，在职业生涯中处理的最有价值的事情，一般都是利用技术解决了业务领域的某阶段的主要问题，这也是最复杂的。

而一个优秀的应聘者，在回答中应该向面试官展现出这样的技术认知。

## 对能力边界的认知

我在做研发的**晋升评审**时，常常会问候选人一个问题：**你觉得一个高级研发工程师和一个架构师的区别在哪？** 这个问题很多研发同学回答得都不是很好，有些人说需要足够的技术经验，懂得高性能、高可用，也有些人说需要懂得管理，带过团队。

这些能力固然重要，但不是作为架构师最核心的能力。下面我通过一个例子，来帮你理解一个高级研发工程师和一个架构师的本质区别在哪儿。

### 例子

我们先来看一下互联网一些大厂的中高级研发工程师晋升架构师的标准，如下图所示：

| | 中高级研发 | 架构师 |
| --- | --- | --- |
| 功能性需求 | 采用合理的技术实现系统功能性需求 | 采用合理的技术设计系统功能性架构 |
| 非功能性需求 | 无要求 | 在系统高可用、高性能、高扩展，容灾性等技术导向方面做出有价值的贡献。 |



可以看出，晋升架构师需要掌握架构知识体系以及互联网的设计经验。

那么是不是可以这么理解：想要成为架构师，需要在掌握原有技术框架原理与开发基础之上，再懂得分布式高性能、高可用的设计知识，这样就可以了？如果你真是这么认为的，那就存在一个技术认知的问题。

可以这样思考，一个中级或高级研发工程师就不需要懂高性能、高可用的设计手段了吗？这些在网上应该也不难找到通用的解决方案，那么他就可以成为架构师了吗？

其实不然，掌握互联网架构设计中的高性能、高可用、高扩展这些非功能性设计方案是基础，但还要看你是站在哪个角色上考虑的。互联网大厂职级体系晋升的一个很重要规则，就是你所做的事情的边界，所能影响到的范围。

比如，研发工程师和架构师能驾驭的边界可以如下概括：

- 一个中高级研发工程师对系统的驾驭边界至少是模块或者子系统层面；
- 一个架构师对系统的驾驭边界至少是全系统层面；
- 一个高级架构师对系统的驾驭边界至少是某一领域层面。

### 案例分析

我们常说，屁股决定脑袋，不在那个位置就不会真正体会到那个位置带来的问题。没有触达多系统层面的设计，就不会掌握多系统层面带来的复杂度和解决问题的思考逻辑。但是往往研发同学意识不到这样的问题存在，即便能碰到一个通盘考虑架构设计的机会，但价值、眼界、认知的形成，也不是一朝一夕的事儿。

那么，你要怎么做才能让自己更快速地成长呢？你要在工作中养成归纳总结的习惯，形成自己的知识体系，沉淀自己的方法论，提高自己的认知能力，并且跳出舒适区，多争取扩展自己能驾驭系统的边界的机会。

在接下来的课程中，我会基于架构设计面试考题的角度，在解决方案、原理理解、实践经验，以及知识体系和认知能力等方面，帮你提高应对架构设计问题的能力。

## 总结

我在今天的课程中，通过三个案例为你讲解了研发工程师在面试中如何提高竞争力，可以总结为三点。

- 首先要提高你对系统架构设计的认知能力，一个好的架构师的架构设计不是仅仅停留在技术解决方案上。
- 其次要提高你分析系统问题的认知能力，做架构设计要具备根据现阶段的主要矛盾来分析问题的能力。
- 最后你要扩大自己能够驾驭系统的边界，因为只有这样才能遇到之前没经历过的**问题层次，注意我这里说的是问题层次，而不是问题数量。**

上一页 下一页