代码只要写得多，就能成为顶尖的架构师?

CSDN 4月14日

以下文章来源于大飞码字 ，作者大飞码字

大飞码字

在国内知名互联网公司工作快十年了，平时喜欢思考一些事情，所以开个公众号跟大家分享...

作者 | 大飞

本文经授权转自“大飞码字”

同时期进入到同一间公司，参与同一个项目的同学，时间长了之后，有同学的架构能力很强，有的却还像一个新手，造成这种差别的原因除了个体素质的差异，还有一个是工作方式和思考方式上的差异。

其实，在工作中，架构的学习和经验的积累，是有一些比较好的方法的，这里，我就来分享一下这方面的一些经验。

01 CSDN

项目，相比数量，规模更重

毫无疑问，在实际工作中，积极参与实际工程项目是快速积累经验最好的办法。

相对于项目的数量，项目的规模更加重要。我们没办法在一个项目开始的时候，去判断一个项目的质量。但项目的规模是可以比较容易判断的。实际服务用户的数量，参与工程实施的各类人员的数量，都可以反应出项目规模的大小。

为什么更应该追求项目的规模呢？因为项目的规模越大，可能遇到的各种架构问题就会越多，你能从中学到的东西自然也会越多。

一个亿级用户的项目比一个千万级用户的项目的复杂度，不是只高一倍的，项目的复杂度是成指数增长的。你在一个千万级用户项目中遇到的一个小问题，在亿级用户的项目中，却有可能是最难解决的问题。

之所以说去大公司好，除了流程上更加规范，也因为用户量更大，系统复杂度更高，个人也能得到更大的锻炼。

当然，在实际工作的时候，你或许没办法选择自己的项目，那就尽量做好手上的吧，一旦有机会，就积极地去争取。

02 CSDN

由点及面的了解系统

规模大的项目，可能不常有，而且人的时间和精力也是有限的，不可能参与无限多的项目。做每个小项目的时候，如果可以尽量多地汲取设计经验，会成长的更快。

那怎么来做呢？ 你需要学会由点及面的了解系统。

当你接到一个小需求，在一个已有的项目上面，增加一个小功能，比如就是数据库的CRUD的操作。你可能觉得很无聊，没啥技术含量，如果你这么想，那你可能错失了一个更好的理解系统，精进经验的机会。

以前，我参与QQ系统后台项目的时候，刚开始接到的几乎都是很小的需求，有一段时间，甚是无聊。后来，有一个前辈跟我聊，他说你要学会由点及面的去了解系统，半年之后，你对系统的理解程度肯定会更全面，深刻，后面有大需求的时候，你才有可能hold得住。

比如，我接到一个增加一条新的客户端协议的需求。这个需求本身实现起来比较简单，因为接口都是现成的，只需要按照规范去设计字段，配置上去就可以了。如果是一般的做法，做到这个程度也就结束了。

但如果你采用由点及面的办法，你应该去了解整个协议链路的设计，你会发现，为了保证协议的可靠性，系统做了很多额外的设计，这个才是系统设计真正有难度的地方。

我后面通过搜索和查找资料，还发现了业界通用的做法 --- XMPP协议。

当时我如果不深入的理解和挖掘这部分，估计到现在都不知道有这个协议。

发现这个协议，对我像是打开了一片新的天地，原来类似的系统设计和协议，早已经有一堆的人研究过，并给出了很好的解决方案。

时间长了，这种工作习惯，能给自己带来很大的成长。很多同学问我，他每天在公司就是CRUD，感觉技术没成长，那你确定自己深度的了解过你在CRUD的系统吗？你有去深入的学习和扩展这部分吗？

（CRUD是指在做计算处理时的增加(Create)、读取查询(Retrieve)、更新(Update)和删除(Delete)几个单词的首字母简写。）

03 CSDN

多思考，这个特别关键

如果只是简单的实现业务功能，很多人都可以做到，根本不需要厉害的人，那厉害的人是怎设计的?

除了功能需求，还需要考虑安全需求，性能需求，可靠性，稳定性等。这些才是系统设计的难点和关键点。

这些需求，是不会从产品经理的口里提出的，这个是架构师的职责之一：从产品需求，业务需求里面提出安全，性能，可靠性，稳定性等系统层面的需求。

一个产品经理不会跟你说，你的系统要保证安全，能抵受黑客的攻击。他们默认，这些属于技术的范畴，应该由技术来解决，当然，这也合理。(他们甚至不知道这些还要设计)

所以，一个合格的架构师，在接到这些产品，业务需求的时候，一定要能够全面的思考，给出除了业务需求外的系统需求，并要求自己或其他同学要去设计和实现这些系统需求。

这是一种思维方式，也是做事的一种习惯。刚开始的时候，你可能没有这种意识和习惯，但你要有意识的去培养它们。

这种思考，到后面可以形成一种架构设计的直觉。比如，我有时候会接到一些很重要的任务，我进行一轮思考和设计后，却发现比预想的要简单，这时候，我的直觉就会告诉我，我可能是遗漏了一个关键的部分。

或者是对需求的理解不充分，或者是对关联系统的了解有盲区。然后我都会重新review 一遍，很多时候，这种直觉，帮我避免了不少坑。

04 CSDN

系统故障后的技术复盘

再稳定的系统，也会有故障。如果是业务高速发展中的系统，那故障的频率应该就更高了。你们的团队，有定期过故障的习惯吗?

我们就经常做这类的事情。

一个故障发生后，肯定是先处理，然后安抚用户，待一切处理完毕，我们通常会由系统的负责人，出一份故障报告。这份故障报告会详细的记录故障的处理过程，比如xxx时xx分，xxx做了什么操作，然后还会详细描述故障产生的原因和后续的改进措施。

这份故障报告写完后，会以邮件的形式发给整个团队，大家会一起来review 故障的处理过程和故障产生的原因。

我们会定期举行故障复盘会议，大家会在一起讨论问题的根本原因和改进的措施，更进一步的，会由点及面的延展开来，全局看待问题。

故障复盘会议，大概一个月执行一次。我们会拉上相关的负责人，一起来看这个月内发生的故障，分析故障的处理流程，分析设计和程序上的问题。

我们发现有的问题是设计的缺陷，有的问题是程序的bug，有的问题是已知问题，但因为成本或其他原因，所以暂不解决。这个过程使得团队成员对系统越来越熟悉，研发流程也被规范的越来越好。

定期的技术复盘，帮我们发现了很多问题，还预防了不少问题的发生，我们从中也发现了很多系统设计上的缺陷。

05csdn

对外的分享，总结，提升影响力

最后一点，项目整体完成后，要尝试去总结和分享，会带来很大的额外收益。

第一个收益。你可以总结自己做的这个项目，通常你都可以发现不少的问题和可改进的地方。这些存在的问题，你应该放到自己的脑子里进行思考。我觉得一个优秀的架构师和一个普通的架构师的区别，很大部分是源自思考的广度和思考的深度。

第二个收益来自于影响力，这个很多的同学都会有点不在乎，但这个影响力越到后面，就越显的重要。影响力地积累需要比较长的时间，所以越早意识到这点，越早有意识地去分享和打造自身的影响力是特别重要的。

对外分享，可以是写篇文章，可以是写个ppt，给组里，给整个项目团队，或者给一些外部会议做分享，都可以慢慢地积累起这种影响力。

在很多的公司，技术影响力也是技术职级评定的一个关键直指标，所以相当的重要。

结语

以上，是我这么些年来，架构设计方面的经验积累。个人觉得架构设计能力的提升和经验的积累，没有特别的捷径，但跟平时的工作习惯和思考意识有很大的关系。

有同学接到一个需求，做完就做完了，其它的也不理会，长久下来，肯定是没什么成长的。如果你想成为一个优秀的架构师，就需要培养这种做事的方式和思考的习惯。

作者：大飞。十年互联网人，资深架构师，技术leader。

【End】

☞曾断崖式跌落的三星，能否在中国东山再起?

☞前端开发这么多年，你真的了解浏览器页面渲染机制吗？ | 技术头条

☞fork()成为负担，需要淘汰 | 极客头条

☞程序员 996 再上热搜，黑名单增至 84 家!

☞与云原生及开源大神们的第二次亲密接触 | 全议程重磅发布

☞V神玩起freestyle! 5位以太坊核心大咖在悉尼的演讲精华全在这了! | 直击EDCON

☞“重构”黑洞：26岁MIT研究生的新算法 | 人物志

☞程序员为什么都爱穿冲锋衣？（最全总结）

```
System.out.println("点个在看吧！");
console.log("点个在看吧！");
print("点个在看吧！");
printf("点个在看吧！\n");
cout << "点个在看吧！" << endl;
Console.WriteLine("点个在看吧！");
Response.Write("点个在看吧！");
alert("点个在看吧！")
echo "点个在看吧！"
```

点击阅读原文，输入关键词，即可搜索您想要的 CSDN 文章。

你点的每个“在看”，我都认真当成了喜欢