序言

Java工程师的出路，做到最后，无非就两个，一个是继续深耕技术，向架构师的方向上发展，另一个就是转型做管理。

但是目前的市场行情、尤其是疫情过后，大部分企业的经济状况很不理想，所以很少会去单独的设立架构师这么一个岗位。大部分情况下，都是技术管理人员，同时肩负着架构师的职责。所以对于java工程师来讲，管理能力也尤为重要。

但是很多做技术的人，并不擅长去做管理。因为在做技术时，更多的是与代码打交道。代码是一个很单纯的东西，只要你认真的对待他，它也一定认真的回报你。但是做管理就不一样了，做管理更多的是与人打交道，人可没有代码那样单纯，人要比代码复杂的多得多。一个经常与代码打交道的人，突然让他与人打交道，他往往会感觉不适应。

所以一个java工程师，本来做技术的时候，做的非常好，非常优秀。但是当他转型去做管理时，却是一团糟。可能他会因此而沮丧、灰心，甚至自信心也受到了严重的打击。

但是我想说的是，这是一种很正常的现象。任何一个人，当让你去做一件你不擅长做的时候，谁也不可能一开始就做的很好，谁都会有一个过程。我也是一样，我在刚刚转型做管理时，做的也是一团糟。我也是经过了几年时间的磨砺，才做的越来越好的。

所以今天，我把我这些年做管理的一些经验，分享给大家，供大家参考。希望大家可以从中获益，一起把管理做的更好。

**作者简介**

**个人信息**

·姓名：贠学文 ·性别：男



·出生日期：1988.02.26 ·户籍：吉林省四平市

·现居：吉林长春 ·工作经验：11年

·毕业院校：黑龙江科技大学 ·专业：信息与计算科学

·学历: 大学本科 ·手机（同微信）：18612230443

·邮箱：18612230443@163.com ·爱好：编程、读书、写作、象棋

·个人的gitHub地址：https://github.com/aloofWolf

**获得荣誉**

·曾在技术领域获取过国家专利，估值800万元。

·经常在今日头条上发表一些技术的文章，目前已有1500+粉丝。今日头条账号：贠学文。

·任职期间多次被评为优秀员工。

·曾被评选为年度杰出面试官。

**自我评价**

· 对软件编程行业也很浓厚的兴趣，曾在技术领域获取过国家专利，估值800万元。

· 对一些技术点有自己独到的看法。

· 有很强的时间观念，工作中严以律己，宽以待人 。

· 有上进心，不安于现状。

· 性格偏内向，但为人真诚。

· 有自己的想法，不喜欢随波逐流。

· 对工作认真负责，有担当。

· 注重过程，不过分在意结果。

· 高调做事，低调做人

· 谦虚但不自卑，自信但不骄傲 。

**专业技能**

|  |  |  |  |
| --- | --- | --- | --- |
| **语言** | java | **关系数据库** | mysql |
| **缓存数据库** | redis | **web框架** | struts2、springmvc、springBoot、jfinal |
| **持久层框架** | hibernate、mybatis、mybatisPlus | **web容器** | tomcat、weblogic |
| **网络传输** | tcp、udp、ftp、socket、http、https | **定时任务** | quartz、xxl-job |
| **json框架** | fastJson、gson、jackson | **模板引擎** | velocity |
| **日志管理** | slf4j、log4j、logback、log4j2 | **消息队列** | rocketMq、kafka |
| **OFFICE框架** | Apache POI | **微服务** | dubbo、springCloud |
| **网关服务** | nginx，keepalived 、gateway | **注册中心** | zookeeper、eureka、nacos |
| **服务调用** | feign、openFeign | **负载均衡** | nginx、ribbon、loadbalabcer |
| **限流熔断降级** | nginx、hystrix 、sentinel | **配置中心** | springCloudConfig、nacos、apollo |
| **分布式事务** | seata、自研分布式事务解决方案 | **分布式锁** | redission |
| **分布式session** | Redis统一存储、jwt | **数据同步** | canal |
| **大数据** | elasticsearch | **加密算法** | md5、rsa、3des等等 |
| **Jvm监控** | arthas | **容器管理** | docker、k8s |
| **版本控制** | svn、git | **项目构建** | maven |

**目录：**

[一、身份的转变 5](#_Toc143870688)

[二、技术能力过硬 5](#_Toc143870689)

[2.1 架构选型与搭建 5](#_Toc143870690)

[2.2 编码能力 5](#_Toc143870691)

[2.3 生产环境维护 5](#_Toc143870692)

[2.4 解决问题 6](#_Toc143870693)

[2.5 任务评估 6](#_Toc143870694)

[三、团队管理 6](#_Toc143870695)

[3.1 梯队建设 6](#_Toc143870696)

[3.1.1 初级java工程师 6](#_Toc143870697)

[3.1.2 中级java工程师 6](#_Toc143870698)

[3.1.3 高级java工程师 7](#_Toc143870699)

[3.2 人才培养 7](#_Toc143870700)

[3.3 不要试图去改变一个人 7](#_Toc143870701)

[3.4 人才招聘 7](#_Toc143870702)

[3.5 学会放权 8](#_Toc143870703)

[3.6 与团队成员的关系 8](#_Toc143870704)

[3.7 奖惩机制 8](#_Toc143870705)

[3.8 团队利益 8](#_Toc143870706)

[3.9 团建活动 9](#_Toc143870707)

[3.10 团队氛围 9](#_Toc143870708)

[3.11 团队稳定 9](#_Toc143870709)

[3.12 拒绝内卷 10](#_Toc143870710)

[3.13 个人能力的提升 10](#_Toc143870711)

[四、质量与进度的保证 10](#_Toc143870712)

[4.1 事前风险评估 10](#_Toc143870713)

[4.2 事中提前发现风险 10](#_Toc143870714)

[4.3 突发事件导致影响进度 10](#_Toc143870715)

[4.4 与测试的沟通 11](#_Toc143870716)

[4.5 与产品和用户的沟通 11](#_Toc143870717)

[4.6 代码检查 11](#_Toc143870718)

[五、向上汇报 11](#_Toc143870719)

[5.1 汇报方式 11](#_Toc143870720)

[5.2 汇报内容 12](#_Toc143870721)

[六、前瞻性 12](#_Toc143870722)

[6.1 业务的前瞻性 12](#_Toc143870723)

[6.2 技术的前瞻性 12](#_Toc143870724)

# 一、身份的转变

技术人员转做管理时，首先你一定要认识的自己的身份已经变了。看待问题的角度也就随之变了。

以前做技术的时候，是从局部的角度看待问题，更关注的是局部问题的实现细节。但是做管理以后，要跳出局部，从全局的角度上，掌控整个项目的进度与质量。

当然细节的问题同样要关注，但是不应该由你自己亲自关注，而是应该由你下面的人去关注，然后向你汇报。而你要亲自关注的内容，是从宏观上把控整个项目，以及对项目做出近期以及长期的规划。

一个优秀的管理人员，首先一定要认识到这种身份的转变，才能做的越来越好。

# 二、技术能力过硬

做为一个技术管理人员，你的技术能力一定要过关。一个技术能力过关的管理人员，才会让别人信服。你下面的人，才会心甘情愿的听你指挥。

而且，现在很多公司的技术管理人员，同时也肩负着架构师的职责。这也就意味着，如果你的技术能力不过关，是无法做好管理这个岗位的。

那么对一个技术管理人员来讲，技术能力要达到怎样的水平，才算是过关呢，我认为主要体现在以下几个方面：

## 2.1 架构选型与搭建

当公司想要去搭一个架构，或者在已经搭好的架构中，启用某个框架或者中间件，你要可以根据深入业务，去分析并找出适合当前业务的架构，或者从一些框架或者中间件做选型，从而完成架构的搭建。

## 2.2 编码能力

核心代码的编写，比如如何抽象出一个接口，供下面的人去写实现。比如如何做低耦合，高内聚。如何提升整个系统的安全性、可维护性以及扩展性等等。

## 2.3 生产环境维护

对于生产环境的维护，有一套成熟的方案，并且可以实施。以保证生产环境的高可用。和出现问题时，对问题的有效追踪。

## 2.4 解决问题

当生产环境出现问题时，你可以解决。当下面的开发人员遇到难点无法解决时，你可以解决。

## 2.5 任务评估

可以准确的评估每个功能实现的难易程度，然后准确的对工时做出评估。这样才能有效的保证项目进度。

# 三、团队管理

团队管理的目的，就是要用最小的成本，来组织一个积极向上的、战斗力强的团队。

## 3.1 梯队建设

在一个团队中，不可能都是高级的开发人员，这样虽然战斗力会很强，但是成本也会特别的高。也不可能都是初级的人员，这样虽然成本降下来了，但是战斗力也随之减弱了。

所以，我们要在战斗力与成本之间找到一个平衡点，这就是所谓的梯队建设。

我觉得，一个良好的队伍，高级与中级+初级的比例，应该控制在1：5左右，是比较合适的。当然这个也不是绝对的，还是要根据公司的团队的情况来具体分析。

那么，对于初级、中级、高级这三个级别来说，该如何划分呢？我认为这三者的区别 如下：

### 3.1.1 初级java工程师

1、按照要求实现好代码。

2、思维逻辑、理解能力要强。这点很重要，技术能力可以稍差一些，毕竟经验不足，但是思维逻辑和理解能力一定要强。当别人在给你讲解一些复杂的业务或者代码逻辑时，你最起码要听得懂并且理解，减少无效沟通。

### 3.1.2 中级java工程师

1、对于如何写出优秀代码、如何做好一个系统、常见问题的解决，要有自己的想法，可能想法不是很成熟，但是必须要有。因为对于中级的java，一般都会让你去做一些某个具体功能的代码设计和数据库设计，然后交给架构师来评审。如果你连如何做好的想法都没有，你是做不好设计的。

2、基础一定要掌握扎实。

### 3.1.3 高级java工程师

1、当你的对于如何写出优秀代码、如何做好一个系统、常见问题的解决得想法，都成熟以后，利用这些想法，自己或者带领团队，可以独立维护一个大型的模块或者大型的系统。  
  
2、对于一些突发事件，或者生产问题，可以快速定位并解决问题。

## 3.2 人才培养

做管理人员，一定要切记，不要花费太多成本去培养一个人。做好能做到招之即用，零成本培养。

有很多做过管理人，都会有这样的一个经历：我觉得一个人的潜力很大，虽然距离目标差距很大，但是我愿意培养你。结果呢，花费了大量的精力和成本，把人家培养出来以后，人家直接走了。你花费的成本，全部浪费掉了，任何的回报都没有。

所以不要花费太多成本去培养一个人。做好能做到招之即用，零成本培养。即使我做不到零成本培养，我也要选择一个举例目标差距不大的人培养，尽量减小培养成本。这样后面即使人家走了，你也没有什么损失。

## 3.3 不要试图去改变一个人

记得几年前，我去一家公司面试的时候，面试官问了我这样一个问题：如果你的一个下属，他的能力特别的强，但是工作态度特别的不认真，非常的懒散，经常不按规定时间完成任务，这种人你会怎么管理他呢？

我回答：这种要分情况，如果他之前工作都特别的认真负责，只是突然变成这样的，那他一定是生活中遇到了什么事影响了他的工作，这时候我会去关心他是不是遇到了什么事，然后尽我最大能力帮他解决，帮助他回到之前的工作状态。

但是如果这个人的性格就是这样，那就干脆换人吧，不要试图去改变他。江山易改，本性难移。大家都是成年人，怎样去对待工作这件事，不需要别人来教。但是他既然已经形成这样的性格了，就很难改变他。而且公司也不会允许我花费大量的成本来改变他。

## 3.4 人才招聘

作为一个管理者，人才招聘，是很关键的一个环节。如果你可以让你的招聘工作变得即高效、又有质量，那将会大大的减轻你工作的压力。

至于如何做好管理这件事，大家可以看我写的另一篇文档《如何成为一个优秀的java面试官.pdf》。这里面有很详细的介绍。



## 3.5 学会放权

可能很多技术人员转做管理时，都有这样一个通病：我把任务分配给别人时，别人做的又慢，质量又不高，做的很难让自己满意。还不如我自己做。所以又开始自己做上了。

但是你要想到，你之前做技术的时候，做的是一个人的工作量。现在你管理的是一个团队，如果团队的工作都由你自己来完成，那就相当于你一个人做了一个团队的工作量，你能做的过来吗？

这样不仅会把你自己搞的特别累，而且你下面的人又感觉无所事事，得不到成长。这对于团队的成长是不利的。

所以管理人员，必须要学会放权。

可能开始的时候，别人完成的任务，你不是很满意，但是你要给他们成长的空间，这样才有利于团队的成长。

## 3.6 与团队成员的关系

作为一个管理者，应该自己把控好自己与团队成员的关系。

不能和团队成员的关系，走的太远，如果走的太远，你就听不到他们的心声与诉求，这很不利于团队的成长。

当然也不能和团队成员的关系，走的太近。一旦走的太近了，在管理时，就会涉及到私人感情的问题，不好管理了。

所以这个度，做为管理者，一定要把我好。

## 3.7 奖惩机制

有很多人说，想管理好一个团队，必要要建立起一个良好的奖惩机制。但是我并不这样认为。

在我看来，奖惩机制未必要公平，但是奖一定要大于惩。

对于一个平时表现的都非常优秀，而且几乎不犯错误的人。偶尔犯了一次错误，可以不惩罚他。可能这在别人眼里是不公平的，为什么我犯错误了就惩罚我，他犯错了就不惩罚呢？

但是这个社会本来就没有那么多公平，你想要公平，你可以变得像他一样优秀。如果你做不到像他一样优秀，那你就必须要忍着社会对你的不公平。忍不了可以走人。对于管理者来说，只需要尽力留住那些优秀的人才，对于那些没有那么优秀的人，随时可以走。

## 3.8 团队利益

做为管理者，要时刻想着为自己的兄弟们争取一些利益，比如项目奖金，优秀员工奖金等等，这样兄弟们才会觉得为你卖命是值得的。

还有就是对产品经理或者用户提出的一些不合理的需要，要强硬的拒绝掉。如果不拒绝掉，难受的不仅仅是你自己，还有你的兄弟们，他们要为此付出很多的辛苦，也会对你怨声载道。这样软弱的管理者，是没有人愿意跟着你干的。

## 3.9 团建活动

有很多管理者，为了提升团队的凝聚力，提升战斗力，会去喜欢组织一些团建活动。但是效果往往不好。这是为什么呢？

首先我们要明白团建活动的目的，它的目的，并不是什么要提升我们团队的凝聚力，也不是为了提升团队的战斗力。它的目的其实很简单：就是大家工作时间长了，累了，带大家出来玩玩，放松放松而已。

有很多这样的案例，公司组织团建活动，目的就是提升团队的凝聚力，提升战斗力。然后会组织很多大家很不愿意参加的一些小游戏，逼着大家参加。结果团建活动结束后，就会有大批量的员工离职。那其实这样的团建活动，做了还不如不做。

所以我们要明白团建活动的目的，就是让大家放松放松，玩的开心。你一定要基于这个想法，去组织团建活动，这样才能这团建活动变得有意义。

## 3.10 团队氛围

团队气氛很重要，每个人都喜欢在一个轻松、愉快、并且积极向上的氛围里工作。

所以如果有人想破坏这个氛围的时候，一定要把这个害群之马驱逐出团队。

比如：有些人可能天生就是喜欢抱怨，如果是偶尔的抱怨，我可以理解，也可以接受，但是如果他长期的抱怨，影响的不仅仅是他自己，而且也会影响整个团队的氛围。作为一个管理者来说，为了维护团队的氛围，一定不要让这种负能量的人留在团队里。

可能大多数技术能力特别强的人，性格上都有些内敛，不是特别的外向。但是如果整个团队都是这种性格的人的话，团队的气氛就会变得很压抑。所以这个时候，作为一个管理者，有必要招聘一个活泼开朗的人进来，调节一下团队的气氛。这个人的技术能力可以稍弱，他的存在，就是为了活跃团队的气氛的。

## 3.11 团队稳定

有很多管理人员，会特别看重团队的稳定性。

但是我觉得，人留人走，是一件很正常的事情。地球离开谁都照样转，没有谁是代替不了的。只要留出充足的交接时间，做好工作交接就可以。

而且技术人员和销售人员又不一样，销售人员，离职的时候可能会带走一些客户。给公司的收入造成影响。但是技术人员能带走什么呢？

所以我们不需要过分的在意这种事情，当然了，能稳定尽量还是保持稳定，但是如果人家真的要走，也不需要过分的在意，在招人就是了。

## 3.12 拒绝内卷

在一个团中，首先大家要明确一个目标，就是把系统做好，让用户用着满意。

然后我们做的所有工作，都是为了这一个目标努力，所有与此目标无关的事情，我们都不需要关注。也要拒绝在一些毫无意义的事情上互相攀比，比如：无效加班、无效开会等等。

做为一个管理者，更应该以实际工作任务为导向。只要你工作做的好了，那么其他的事情，都可以睁一只眼闭一只眼，只要别做的太出格就行。

## 3.13 个人能力的提升

我在开始做管理的时候，经常会定期组织一些技术分享，通过技术分享。让大家每个人的技术能力都得到提升。

但是时间长了以后我就发现，这种技术分享会，一点价值都没有。对于那些想学的人，即使你不组织，人家也会私下里学习进步。对于那些不想学的人，即使你组织了，也是左耳听右耳冒，没有一点价值。

所以现在，我一般不会去组织什么技术分享了。只是对于那些想进步的人，我会和他多交流交流，把我自己的一些经验分享给他，帮助他进步。

# 四、质量与进度的保证

## 4.1 事前风险评估

在项目开始前，要充分的评估出项目中可能遇到的风险，并做好应急措施。对于自己无法处理的风险，一定要在项目开始前向上抛出，让大家共同商讨应急措施。

## 4.2 事中提前发现风险

项目开发中，可以定期以站会的方式，过一下大家的进度，以及是否遇到问题。这样可以提前发现风险，提前处理。

## 4.3 突发事件导致影响进度

有很多初做管理岗的人，经常会遇到这样的情况：

在项目开发中，突然有个人有急事必须要请假，导致任务无法完成，也无法按期交付。

有很多人说，预防这种情况，就是要提前做好备岗。但是根据我多年的管理经验来看，备岗这种事情，太理想化了，实际上很难实现。

对于一些重复性的工作，当一个人接收另一个人工作时，不需要交接，或者只需要少量的交接，那这种工作是适合备岗的。

但是程序员每天的工作，并不是重复性的，一个人接收另一个人工作时，必须要做好严格的交接，别人才能很快的接手。我们不能说每天下班的时候，两个人都把自己的工作给对方交接下，以防第二天有事来不了。这谁也做不到，我也做不到。

所以备岗并不是解决这种问题的最佳方案。

我来解决这个问题的方案，就是在评估工作量时，多评估出一些来。比如一个任务三天就可以完成，我可以评估五天完成。但是我会要求开发人员必要在前三天完成，这样即使你后面有急事需要请假，也不影响进度。

## 4.4 与测试的沟通

在测试阶段，经常与测试沟通，关注测试的进度，bug的多少，是否会因为bug太多影响按期交付。

## 4.5 与产品和用户的沟通

在交付前，要为产品经理或者用户做一下系统演示，看看是否符合他们的预期。如果有哪里不符合预期，要在交付前即使调整。

## 4.6 代码检查

制定代码规范，定期组织代码检查，来保证代码的质量。检查的内容包括：代码是否符合规范、是否存在一些隐藏的、测试难以发现的bug、某些功能是否有更好的实现方案等等。

# 五、向上汇报

作为一个管理人员，除了要向下管理以外，还要学会向上汇报。目的是让上级领导了解你目前的工作状态与工作进度。你也能这这个过程中，了解到领导对你和对你的团队的看法，以及领导的一些诉求，这样才能更好的开展工作。

## 5.1 汇报方式

可以每周以邮件的方式向领导汇报。

也可以在平时，找个领导和自己都不忙的时间，以聊天的方式汇报。

## 5.2 汇报内容

1、目前的工作进度、工作状态、以及团队的目前状态等等。

2、自己对于工作的一些想法。

3、自己无法解决的事情、需要与领导协商或者领导协调的事情。比如你发现团队某个成员工作态度不积极，工作都是拖后腿。你想换掉他。但是换人的成本是很高的。这时你可以找领导一起商量下，从工作与成本之间找到一个平衡点，寻找出最终的解决方案。

# 六、前瞻性

前面说到了，作为管理者，要学会放权。那么放权以后，要把更多的精力投入到对项目的近期和远期规划上面。也就是作为管理者，你需要具有前瞻性。

## 6.1 业务的前瞻性

经常与产品精力或者用户沟通，看看他们对系统的反馈如何，还有哪里需要改进，以及他们还有哪些需求。了解了这些以后，在制定出一份近期的与远期的规划。然后与上级领导、产品经理或者用户，一起分析计划的合理性，制定出最终的规划。

## 6.2 技术的前瞻性

经常的回顾一下自己的系统，看看系统中，在技术层面有哪些可以优化之处。比如性能方面、安全方面、扩展性方面等等。也时刻关注着市场里最新的技术的动向，看看有哪些好的技术可以应用到系统里面来。然后制定出一份近期的与远期的规划。然后与上级领导、开发同事，一起分析计划的合理性，制定出最终的规划。