

T O P

T E C H

T E A M

中国顶尖 CHINA 顶尖

2016年／第五季

技术团队

|访|谈|录|



扫一扫，了解更多

Broadview®
www.broadview.com.cn

滴滴出行李贤辉：滴滴iOS客户端的架构演变之路

InfoQ记者采访了滴滴出行平台产品中心iOS技术负责人李贤辉，了解了滴滴iOS客户端的架构演变之路。



刘译璟：创业公司是否是技术天才的理想国

InfoQ采访了刘译璟博士，借此了解他的个人经历，以及探询百分点的技术和文化。

前Google全球研发总监： 好的架构需要精心设计

为了了解Google的早期成员所看到的Google成长背后的故事，InfoQ记者采访了小红书CTO 郑小虎。

鸟哥：技术人员如何保持进步

去年“鸟哥”惠新宸离开微博加盟链家一度为业界盛事。藉此机会InfoQ再访鸟哥，听他讲讲链家技术团队的文化、思想与经验分享。

陈科：在创业公司的架构师需要解决哪些问题

本文作者分享了他在创业公司遇到的一些挑战以及解决方案。

中国顶尖技术团队访谈录 第五季

本期主编 郭 蕾

流程编辑 丁晓昀

发行人 霍泰稳

联系我们

提供反馈 feedback@cn.infoq.com

商务合作 sales@cn.infoq.com

内容合作 editors@cn.infoq.com

Geekbang
极客邦科技



FACILITATING THE STUDY AND COMMUNICATION OF TECHNICIANS

让技术人员学习和交流更简单

EGO EXTRA GEEKS' ORGANIZATION
NETWORKS

高端技术人员
学习型社交网络



InfoQ ueue

专注中高端技术
人员的社区媒体



StuQ ueue

实践驱动的
IT职业学习和服务平台



GIT GEEKBANG
INTERNATIONAL
TRAINING
极客邦培训

一线专家驱动的
企业培训服务



地址：北京市朝阳区洛娃大厦C座6层1607室

网址：www.geekbang.org



卷首语

七年，沧海桑田

作者 **QCon 大会主编 岐秀涛**

春光明媚，花开正好。

当您拿到这本书的时候，应该正是 QCon 北京 2016 的现场。

既然来到了现场，想必您对 QCon 已不陌生。

QCon 于 2007 年首次在伦敦举行，之后便遍地开花，影响着越来越多的技术人。

2009 年 4 月，QCon 来到北京。当时 Spring 之父 Rod Johnson、ThoughtWorks 首席科学家 Martin Fowler、eBay 杰出架构师 Randy Shoup、SitePen 的 CEO&Dojo Toolkit 联合创始人 Dylan Schiemann 等国际

知名专家带来了分享。程立（花名“鲁肃”）、洪强宁、冯大辉、岳旭强、周爱民等国内专家担任出品人或演讲嘉宾。

七年过去了。

在历史长河中，七年转瞬即逝。

但是在互联网行业，七年，沧海桑田。

七年过去，当时的架构师，现在很多已经成为行业的领头人或知名企业的 CTO。比如程立现为阿里巴巴集团的合伙人之一，蚂蚁金服 CTO；冯大辉现为丁香园 CTO；岳旭强现为蘑菇街联合创始人，等等。

七年过去，云计算已经落地，越来越多的新兴企业选择在云上搭建自己的服务；大数据不再是抽象的概念，Spark、Storm、Druid，各种框架涌现，各大公司也纷纷研发自己的大数据处理基础设施，从海量数据中挖掘价值；机器学习、人工智能也不再是高高在上，Google 的 AlphaGo 完胜围棋冠军李世石，也进一步向世人显示了人工智能的魅力和潜力；移动互联网已经深入人们的日常生活，2015 年的天猫双十一，移动端交易已经占比 68%，2015 年第 4 季度，移动端收入已经占到 Facebook 的 80%。而这一切的背后，都是各种软件开发技术、架构在支撑。

今年已经是 QCon 第 8 次在北京举行。参加过第一次 QCon 北京的洪强宁老师将再次归来，担任联席主席和云平台专题的出品人；Dylan Schiemann 也将归来，分享 TypeScript 和 ECMAScript 6 方面的最新经验。我们也根据当前的行业趋势，设置了业务

核心架构、大数据应用、工程效率提升、技术创业、移动开发与测试等 20 余个热点技术专题。

除了参会者可以现场聆听一线技术专家的实践经验，我们还会在会后逐步发布 QCon 的演讲视频，让更多的 InfoQ 读者学习。

Instagram 的联合创始人兼 CTO Mike Krieger 曾说过，在他架构 Instagram 的过程中，QCon 的演讲视频给他很多启发和帮助。

除了邀请专家来大会上分享，InfoQ 还策划了很多专访和深度技术文章。

您手中的这本《中国顶尖技术团队访谈录》，会聚了我们对滴滴出行、小红书、链家网等 IT 团队的专访，希望展示他们的文化、思想与经验，对您有所启发。

不忘初心，砥砺前行。“促进软件开发领域知识与创新的传播”，是我们不变的追求。



我们的使命：

引导高端技术人学习和成长

我们的愿景：

全球最具影响力的高端技术人社交网络

关于EGO：

EGO是极客邦科技旗下高端技术人聚集和交流的组织，旨在打造全球最具影响力的高端技术人学习和成长平台，线上线下相结合，为会员提供专享服务。

EGO采用实名付费会员制，每一位申请加入的会员都必须经过EGO组织的严格审核，保证会员信息的真实性，让每一位会员在平等和相互信任的环境中分享交流，大家共同学习，共同成长。



微信公众号
EGO



即刻加入EGO

为什么加入EGO？

学习交流

独一无二的学习和交流体验，会员间平等分享各自的实战经验，从工作到生活

拓展人脉

认识国内外志同道合的技术同行，分享交流，拓展自己的人脉圈子

获取信息

获取最新最重要的行业信息及资讯，了解最新技术动态、新闻、社会热点，独家原创的专家观点及评论

解决问题

身边的智囊团，遇到问题时能够第一时间通过EGO平台找到答案

工作之外

在日益激烈的竞争中平衡工作与生活之间的关系，提高软技能

入会资格

1. 认同EGO理念，遵守EGO规则；
2. 热爱学习，乐于分享，追求成长，相信技术的力量
3. 公司/团队中的技术决策者，如CTO、首席架构师、技术总监等；
4. 在某一技术领域具有超过5年的工作经验。



► About StuQ /关于StuQ

StuQ 是极客邦科技旗下 IT 职业教育业务，通过实践驱动的系统课程，帮助 IT 从业人员提高职业技能，获得更好的工作机会。

有干货，有情趣，随手拯救技术宅，轻松报名微课堂，免费获取技术大会 PPT。

► Online Course /在线课程

StuQ 不仅邀请业内顶尖的技术专家录制视频课程，还会为用户在学习环节提供项目实战演练，技能测试，工作机会推荐 3 个课程服务。



StuQ微信公众号: StuQ

► Micro Class /微课堂

讲应用，重实战，强互动，打开微信收获一堂干货课。

► OUR CHARACTERISTICS /我们的特色

一线专家



远离教书先生,我们邀请行业里的技术专家，将当下最需要技术讲解给大家。

技能评估



课程中，讲师将会以实际项目，对学员的学习成果进行评测与考核。

项目辅导



通过参加我们的课程项目，在实践中进行学习，提升解决项目中具体问题的能力。

专家认证



通过参加StuQ专家顾问团的面试，获得来自专家的技能认证推荐信。



一线专家团队驱动的企业培训服务



Enterprise Training

—
同国内外企业深度合作，为其设置专属定制化学习课程，紧密贴合用户个性化需求的内容与形式，提供top级的学习体验。

Deep Workshop

—
携手全球知名一线技术专家团队分享软件研发技术管理实践，提供研发团队提升必修精选课程。

Open Class

-
- 工作坊学习体验、案例干货尽收
 - 大时段分享
 - 全程30个精选课程研修
 - 纯净绿色学习环境
 - 情景教学、沙盘演练
 - 以本土化选题适应城市风向需求

专访滴滴出行李贤辉： 滴滴iOS客户端的架构演变之路



作者 徐川

李贤辉，滴滴出行平台产品中心 iOS 技术负责人。多年客户端开发经验，2010 年加入百度日文输入法部门，2011 年从 C/C++ 的 PC 端开发转到百度音乐移动端的 iOS 开发，2013 年加入滴滴。

滴滴出行作为国内互联网出行平台的代表，短短几年时间就从小的创业公司成为一线互联网公司，发展速度令人瞩目，在如此快速的发展之下，滴滴 App 在架构上又经历了哪些演变？InfoQ 记者采访了滴滴出行平台产品中心 iOS 技术负责人李贤

辉，了解了滴滴 iOS 客户端的架构演变之路。

InfoQ：请向大家介绍一下您自己，您是何时加入滴滴的？

李贤辉：大家好，我是李贤辉，业余时间喜欢踢踢足球和旅行。我在 2013 年 5 月加入滴滴，目前是平台产

品中心的 iOS 负责人，经历了滴滴的 2.0 到现在的 4.2.5 版本，并且随着滴滴的快速发展和业务增加，团队规模也不断扩大，业务上经历了拆分整合的过程。很幸运能进入一线互联网公司，选择了热门的技术方向，并且跟随快速增长的创业公司一起成长。

InfoQ：在滴滴高速发展的过程中，iOS 客户端架构发生了哪些大的改变？

李贤辉：2013 年 5 月，我们启动了滴滴 2.0，主要是 UI 交互方式的改变，采用了 MVC 的方式，代码结构上做到了更清晰。

2014 年 5 月，新增了专车业务线，需要支持出租车、专车 2 个业务线，启动了滴滴 3.0 的开发；借鉴了游戏里的状态机，把订单中的阶段，例如：出租车的等待抢单、出租车的等待接驾、专车的等待抢单、专车的等待接驾，都当成一种独立的状态，每个状态机只需要知道可能要导向的状态机，从而做到了相对独立，状态机满足了出租车、专车双业务线的需求。

2015 年 1 月，滴滴开启了多元化，需要快速上线顺风车、代驾、试驾、巴士等多项业务，代码耦合严重，功能开发和协同发版成为痛点；滴滴的

技术团队分散在北京、杭州、上海，不在同一个城市，而需要在同一个 App 里进行发布，客观上增加了架构的难度。

2015 年 6 月到年底，滴滴进行代号为 The One 的组件化框架开发，为了实现“代码治理”，可以“分而治之”的目标，在各个业务线各自开发的情况下防止代码大面积腐化，方便未来再做更加细致的架构重构，采用 CocoaPods 的方式进行拆分；组件化之后，公共部分拆分为技术组件、公用业务组件，每个业务线是单独组件，如出租车组件、专车组件、快车组件；通过把不同功能的组件代码拆分到不同的 pod 里，实现了业务线仅依赖于公共就可以迭代开发，改善了功能开发和协同发版。组件化只是做了治理的第一步，后面的架构优化还任重而道远。

InfoQ：目前滴滴 iOS 采用的是哪种架构模式？在 controller 瘦身上有哪些实践？

李贤辉：目前滴滴的 iOS 架构采用 MVCS 和 MVVM 的架构方式。MVCS 的 S 是 Store 的意思，包括服务器访问接口控制、数据共享、数据缓存的能力，每个 Store 处理一类情况，

例如：订单 Store 会包括发单、取消订单、结束订单、评价订单等，常用地址 Store 会包含编辑常用地址、删除常用地址、拉取常用地址等。MVCS 这种方式的思考逻辑是希望做到组件无状态，完全依赖于 Store 所返回的各种状态信息，这种思想是非常类似于 React Native + Redux 的思路。

项目启动时，原计划用 Objective-C / Java 这种原生代码做一个类似于 React Native 的框架，但是工作量过于巨大且不成熟，不适合在 The One 这种量级项目中使用，后来没有开发该框架。采用 MVCS 的同时也采用 MVVM，用 VM 为 Controller 减肥，但 VM 要通过 Store 处理数据层逻辑，达到了为 Controller 瘦身的目的。

InfoQ：滴滴 iOS 客户端的组件化是如何划分的，采用什么技术实现？

李贤辉：我们在 15 年后半年实施了组件化，组件包括技术组件和业务组件，技术组件是可以跨 App 使用的，例如：网络组件（长连接和短连接）、界面导航管理组件（统一界面转场方式，模块间界面转场，通过 openURL 方式解耦）；根据业务功能，已经实现了支付、登录、消息、定位、广告 SDK、数据统计、分享等组件，每个组

件是独立的 CocoaPods。

组件采用私有 CocoaPods 来实现，并采用了 Local Pods 的方式，可以在本地不提交代码的情况下，组件与调用方实现调试。组件间的页面间跳转支持 openURL 的方式，由 ONERoute 模块进行管理，页面在 +(void)load 方法中完成注册，ONERoute 内部保存一份 URL 与 Class 的对应表，当调用 openURL 时，会查找到对应的类，然后生成对应的实例对象。这种方式可以通过 URL 解耦具体的类名称，方便从 H5 拉起 Native 页面，未来还可以实现流程的可配置化。在设置页面里，还是直接依赖类的方式，避免过度使用 openURL。为了增加安全性，每个页面会设置是否允许外部打开，仅有允许外部打开的页面才可以通过系统的 openURL 方式打开。

InfoQ：从客户端展现来看，滴滴需要同时获取快车、的士、专车等的实时数据，在数据的传输、展示上有哪些挑战？滴滴是如何应对的？

李贤辉：挑战如下：

1. 消息不及时：如乘客发单后，有司机抢单了，需要尽快知道。后来采用了 socket 长

- 连接，使服务端具备主动通知客户端的能力。
2. 流量过大：我们和服务器交互较多，流量消耗较大。我们现在在用 protobuf 作为数据载体，有效的减少了流量消耗。
 3. 数据易被修改：共享业务数据多，最初没有限制修改数据的权限，出现一些莫名其妙的问题；后来一方面减少共享的业务数据，并且共享数据对外，尽量是只读，如果某个状态来源于服务器，则只能在模块内部，在收到服务器结果时，修改接口数据，即 MVCS 的 Store 内部可以修改数据。
 4. 共享地图是另外一个挑战，为减少内存占用，业务线之间切换流畅，在滴滴首页只能有一份地图实例。首页作为主页面，包括导航栏、地图、多个业务线子页面。为了降低对业务线代码的侵入性，业务线首页从系统的 UIViewController 派生，由首页来设置业务线首页的背景色为透明色；业务线首页的 UI 元素，会在业务线内部完成处理；而在地图模块的操作，首页会收到手势，通过把手势传递给地图模块，使地图得到响应；此外，为了使业务线之间的地图元素互不干扰，每个业务线的都包括一块独立画布，所有地图元素都在独立画布里处理，当切换业务线时，移除原画布上的所有元素，切换回来时，再恢复画布内容。
- InfoQ：你们是否使用了热修复技术，目前实践情况如何？**
- 李贤辉：**滴滴采用了 Lua 和 JSPatch 两种热修复技术，由于 JSPatch 更加被苹果官方认可，目前在线上主要使用 JSPatch。我们还在开发一套 JSPatch 发布平台，内部使用者可以通过后台可以下发和管理脚本，并且处理传输安全等部署工作，提供了脚本后台管理、版本管理，并保证传输安全等功能。
- InfoQ：滴滴 iOS 团队和 Android 团队的代码共同情况如何？**
- 李贤辉：**iOS 和 Android 团队在 socket 长连接库使用了同一份代码，

iOS 和 Android 底层都是 Unix 系列的内核，对于网络操作是一致的，可以使用同一套系统接口，其次，这套代码的稳定性要求很高，功能与系统无关，一份代码可以降低人力消耗，并保证一致性。对于和系统相关的功能，都采用设计思路一致，iOS 和 Android 分别开发的方式。

InfoQ：在对新技术的应用上，滴滴目前是否有用过 Swift？或者准

备何时采用？

李贤辉： 滴滴目前还没有用 Swift，因为在 iOS 8 及以下的平台上，使用 Swift 需要将 Swift 运行时打包到 App，会增大 App 体积。滴滴的乘客端由于受限于即将超过 100M 的包体积，目前还没有采用 Swift 的计划。滴滴的出租车司机端预计在下一季度会小范围尝试 Swift。

扫码关注回复“架构”

了解互联网公司 App 架构实践



前Google全球研发总监： 好的架构需要精心设计，千万不要把问题留给进化



作者 郭蕾

郄小虎 (Tiger)，于 2003 年从普林斯顿 CS 辍学加入 Google AdWords 核心团队，负责广告系统核心技术和产品的开发，因为对广告系统的突出贡献，两次获得 Google 创始人奖。去年年底，Tiger 加入小红书担任 CTO。去年年底，Tiger 加入小红书担任 CTO。

小红书是一家专注于海外购物分享的移动平台，通过 UGC 社区形式，帮助用户找到全世界的好东西。去年年底，时任 Google 全球研发总监郄小虎 (Tiger) 宣布加入小红书，并出任 CTO。从全球顶级的互联网公司，再到

一家成立不到 3 年的创业公司，Tiger 的选择令人诧异。为了了解 Tiger 的人生经历，特别是作为 Google 的早期成员所看到的 Google 成长背后的故事，InfoQ 记者采访了小红书 CTO Tiger。

InfoQ：2003年的时候，您就从PhD辍学，并加入Google，而当时Google还只是家创业公司。能介绍下这段经历吗？

Tiger：在今天看来，很多人会说当初我辍学，放弃PhD进入Google是个非常明智的选择。然而，2003的Google还只是一家很小的公司。

当时很多人说拿了PhD学位安安稳稳的进研究院多好，不是每个辍学的人都能成为扎克伯格或者比尔盖茨。但我很庆幸自己的人生第一次抉择选对了。

知道谷歌是因为那时候，有人在偶尔提起时说，“你搜一个什么东西可以试一下，那个名字很难记的G开头的网站。”

那时候市面上还没有什么真正好用的搜索引擎。经学长推荐用了Google，一旦你用了，就像吸了鸦片一样，根本停不下来。

于是当我想要做点事情的时候，首先想到了这家公司。

InfoQ：那当时，我想你应该还有很多选择吧，比如IBM、微软之类的大公司，选择Google是为了梦想？为了爱好？为了改变世界？你认为这是冒险吗？我想现在很多人也面临和

你当你一样的选择，能给点建议吗？

Tiger：同一时间发给我入职offer的，还有微软研究院、IBM研究所等成名已久的大公司。但我却选择加入Google，当时看来就像是放弃去住五星级酒店而选择去住地下车库一样。

其实做出这个选择很简单，因为微软、IBM面试时，面对的都是一群头发花白，60多岁的老爷爷，中规中矩却有些死气沉沉。

但在谷歌面试我的是副总裁艾伦·尤斯塔斯(Alan Eustace)，我记得两年前全世界媒体还报道了他从4万米高的平流层完成了个人高空跳伞，打破三项世界纪录。

这就是Google带给我的感觉，年轻、活力，充满了可能。稳定的背后是不变，不确定的背后，是无限种可能。

我相信，在重要的时间节点和选择上，抛开更多的“应该”和“最好”，跟着自己的感觉走，至少不会给自己留有遗憾。

直到我入职那天，才知道我要去做广告后端系统，现在是Google商业化的核心，但当时我们团队就只有四个人。

然而今天，Google全球有超过

五万名员工。短短十几年里，公司的规模至少增长了五十倍。

InfoQ：Google 总是能在技术上先别人一步，你想过里面的原因吗？如果生活三个最关键的点，您认为是什么？

Tiger：如果回头来看的话，我认为可以总结三个点：

第一个是需求导向。产品需求、业务需求、用户需求给谷歌创造了一个前所未有的机会。

谷歌当时刚好就站在了那个风口浪尖上，因为他做搜索这样的产品，互联网正好处于这种爆炸增长的阶段！所以对于搜索这个产品的使用率和产生的数据都是前所未有的，之前没有人见过这样大的数据。这个需求就会要求谷歌不能去从现成的这些解决方案中去找一个答案，没办法找一个现成的产品把它拼凑起来。所以我觉得这一点谷歌从一开始就看得很清楚，就是市面上没有任何一家现成的产品能够解决我们所需要处理问题的这种规模。所以我们必须要把这个推倒重来，重新自己从无到有搭建一个合适的系统。因此，日后很多的技术领先，都是基于这个前提下的。

第二个是，技术的思考高度，和对于基础架构的重视。

当时谷歌是有一些很具体的问题需要去解决。但是它并不是真是对于这个问题，它还是尽量的把这个东西做的非常的通用。比如说其实当时谷歌最早的一批人，他们站的高度还是挺高的，他们并不局限于这个东西只是为了要解决谷歌今天面对的问题，而是要搭建一个非常强大的，之前没有任何人做出来的一个大规模的数据中心。从数据中心到集群，就是服务器的集群它都是从头做起的。这样其实，就已经走在了很多其他公司的前面，其实谷歌的这一步是走的非常超前的。在有了这么一个硬件的基础之上，有一批非常厉害的人，在上面开发出了这样一些非常有深远影响的基础架构。

第三个是人，选用最好的人，即便做最普通的工作。

不管在哪里，人的问题我觉得这一点还是挺重要的。谷歌一开始招的那些人，就把招聘的标准放的非常高。就是可能招进来的人做的工作并不是特别高级的工作，但是他一直招聘的理念就是说我把最好的人招进来让他做最平凡的工作，他也是能做出闪光点的。这样才能保持公司持续的创造力。

在谷歌，就是每天都有很多事情要做，还有很多新东西要学。我觉得 Google 做的很好的一件事情就是招聘非常严格，非常注重少而精的模式。

比如说我们做 AdWords 系统，一共四个人，如果配合得好，每个人都非常的 productive。如果一百人搞这件事情，速度肯定会慢很多。还有 GFS，就是三个人写出来的。

InfoQ：你怎么看业务驱动和技术驱动？在 Google 内部，是技术驱动还是业务驱动？

Tiger：我觉得业务驱动、技术驱动，其实不如理解为商业模式的创新，和技术创新。但在实际工作中，我觉得没有固定的模式，不同的产品会有不同的需求。

例如，搜索显然是一个技术跑在前面的产品，可能因为搜索本身并没有什么商业模式。它就是一个完全免费的给用户使用的工具，这完全依靠技术驱动。正是因为谷歌发明了全新 PageRank 的算法，能够比当时所有的搜索引擎的结果要好，于是，技术驱动产生了这么个效果。

但谷歌的广告系统，就变成了一个商业模式创新驱动的产品。技术创新可以解决实际的问题，但如果连问题都没有找到，那技术本身没有实际意义。

最早我们开始搭建谷歌广告系统的时候，首先在想的是，这应该是一

个怎样的商业模式，在大家经过讨论和商量后找到了大致的方向，然后在这种前提下，去想技术可以如何实现。当我们希望将广告的相关性做的和搜索一样，在此之前没有任何一个系统能做到这点，于是意识到需要一个强大的信息学习系统，这个信息学习系统所需要的处理的数据量的规模是之前从来没有人看到过的。在这个思路下，我们先打造这样的一个基础架构，然后再把这个广告系统当成一个应用在上面来解决。

将业务和技术分开来看，可能是因为目前国内的大多数互联网公司，采用产品和技术分离（至少是两个团队）的方式来工作。而在谷歌早先，我们并没有那么明确的划分。Google AdWords 系统，当时也就是我们十个人不到的团队负责，从思路、商业模式搭建、技术实现，都是在共同讨论实现的。

因此具体是哪个优先，还是要以产品的属性和需求来看。

InfoQ：业内某位同行曾经说过，好的架构是进化来的，不是设计来的。你认同这句话吗？你怎么看架构的演化？

Tiger：我不认同，我认为好的架构，都需要经过这么几个过程：设计

- 进化 - 进化 被推翻 - 再设计，是这样循环往复的过程。最开始的架构，肯定是从无到有，根据产品的需求和当时的业务的需求设计出来的。

我觉得，一个系统的演化，一般会经过这样的阶段：第一个系统肯定是 under-engineer 的，从无到有被设计出来后，肯定是不完善的；第二个版本，一般是 over-engineer 的，因为随着之前那个版本的使用，积累了一定量需求后，会发现想要增加很多内容在上面；到第三个版本，应该是最恰当的，减去了一些没必要的设计之后，更合适。但当到了某一个时间点，发现现有的系统架构已经没有办法再维持下去，跟不上需求增长时，就需要推到再重新设计。

拿谷歌的广告系统来说，我 03 年加入了 Google，当时的架构还是比较简单的，只分为两层，Web Serving 层和存储层。所有的数据都存在 MySInfoQL 里，前端的 Web Server 会把用户搜索的关键词转化成一个数据库的 query，然后把所有的查询结果做聚合和排序。

但很快就遇到了问题。我们有两个 customer，一个是 eBay、一个是 Amazon，他们什么关键词都买，所以

他们一家就要占用一个独立的数据库，他们的量还不断往上涨 当时的解决方案呢，就是多做一层分离，把存储数据和需要响应在线搜索的数据通过分开，增加一层 cache server。后来很快用 customer 做分片也不足够了。我们就转为了用 keyword 的 fingerprint 做 shard key，从增长最快的数据着手解决。其他还做了的包括异地容灾，多套 primary 数据中心同时运行等等，都是后来的升级了。架构的演进，一般是这样一个周而复始的循环过程。

InfoQ：Google AdWords 从最初到现在的稳定，经历了哪些重大的调整和优化？有哪些是国内公司可以借鉴的？

Tiger：除了我刚才说的，还有过一次升级，大概是在 04 年到 05 年之间。这一次在结构上有更加根本的改变，主要是考虑了 geographical 的 redundancy。因为那时候赚的钱已经很多了，服务一旦 down 掉后果是非常严重的。虽然我们是从数据存储上有很多 replica，但是 primary 还是很容易 down 掉的。我们后来就建了两套 primary，在 primary 的更新会被 push 到其他所有的 replica datacenter 去。

一旦某一个 primary DC 出了问题，比如地震之类的，我们可以很快 switch 到另一个 DC 去，正常情况下两套系统会同时运行。我们的每个 replica datacenter 也会有两套不同的 stream，data 也分别来自于不同的 primary。这两套 stream 是完全独立的，一旦一个 stream 出现问题，可以很快的 switch 上另一个 stream 上去。

当时也没有成熟的 Auto failover 的机制。当时能买到的就是 Oracle，但是没办法 scale 到那种程度的。理论好像有了很多年了，但是真正在实践中一个在很多 Master 中投票，选出 winner 的算法也是没有的。这个也是 Google 当时才实现出来的。其实还有很多底层问题，也有一些比较有意思的事情。比如说当你有五十个 datacenter 的时候，你怎么把 data push 过去，也是一个挺大的问题，因为数据量很大。当时我们也没有用现成的 solution，也没有任何的现成的 solution 可以用，我们就自己研究了一套。说白了它就是一个 pub-sub 或者说是一个 multi-cast 的问题吧。一开始的时候，data push 的 latency 是用小时来计算的，就有人利用了这个延时长的缺陷刷了很多广告的 impressions。因为我们的 replica DC

需要几个小时才能知道某个 customer 的预算已经用完了。后来经过了一系列的优化，我们才把跨 DC 的 latency 做到了分钟级别，如果是同一个 DC，那就是秒级别了。

InfoQ：可否分享下您这些年在架构方面积累的经验？

Tiger：首先，我觉得没有一种 general purpose solution 是可以拿过来就能用的。尤其在 Google 我们当时很清楚地意识到我们将会面临的挑战是其他人都没有遇到过的，而且用那样的方式一定是行不通的。但是，Google 的经验确实可以帮助把握好大的技术方向。

比如说，从一个 rule-based 的推荐系统切换成一个 model-based 的系统，一定会有一个比较大的提升。之后的算法还可以优化，也会带来提升，但可能很难再超越之前的提升幅度。具体的算法实现、优化就得靠团队里的这群年轻人了。

InfoQ：能不能谈谈您这一路走来，对人生、行业的一些认识和感触？

Tiger：去年年底，我加入小红书，从无到有的过程对我来说，就像当年在 Google 重塑广告系统一样，令人兴奋。

每一次转型，都是逼着自己跳出

原先的舒适圈，在面对和探索全新领域的过程中，也在不断的学习和试探自己的潜力边界。如果给二十几岁的我一些只言片语，或许我会在四十一岁的档口，对自己说：

- Small things follow your head, Big things follow your heart
- 值得去的地方，没有捷径；难走的路，才更值得开始
- 坚持执念，不负初心
- 适时清零，挑战自我升级

InfoQ：你后续的职业规划是怎样的？

Tiger：2014 年，为了把 Google 重新带入中国市场，我开始做了一系列针对国内互联网环境的调研。

这个调研的目的，是为了向 Google 高层建议，应该在中国开展一些什么业务，不过在调研过程当中，我越来越深地感受到：未来几年里，在社交、移动电子商务、智能硬件、移动支付领域，全球的 leader 会从中国出现，而不是美国。

我在谷歌 13 年，接下来，就想在中国市场，找下一个 Google——未来很长一段时间，我相信小红书会成为我接下來的全部。

InfoQ：就您这些年的从业经验，能否给中国程序员一些建议？

Tiger：有人曾经问我，该如何定义一个好的技术人才？

我觉得这没办法给出一个统一标准，如果用 Google 的文化来衡量，那就是能否把一个以前从来没有接触过、或是完全不会的东西，在很短的时间内学会，并且做得很好，这是一个很重要的因素。和学习能力相比，个人经验就显得并没有那么重要。

这一点我相信放在哪里都同样适用，但同时，随着回国组建团队，开始慢慢发现中国工程师和美国工程师的一些差异。

中国的工程师有一些明显的长处，比如说，很擅长做一些理论基础很强的事情，算法、基本功、计算机能力都很强，能一件事情做得很工整、很好。

但不一样的地方在于，中国工程师的思维发散程度明显不如美国的工程师们。我们可能特别习惯于被分配一个工作，把这件事情做好。然而那些硅谷的工程师，他们每天中午吃饭、或是喝咖啡的时候，随便聊天都是在说，我应该做一件什么样的事情去改变世界。他们对这些事情，特别感兴趣。

鸟哥：技术人员如何保持进步



作者 魏星

“**鸟哥**”本名惠新宸，是国内最有影响力的 PHP 技术专家，PHP 开发组核心成员，PECL 开发者。曾供先后职于雅虎、百度、新浪微博，以及链家。惠新宸是 PHP 7 核心开发者，PHP 5.4、5.5 的主要开发者。作为 PECL 开发者贡献了 Yaf，Yar 以及 Yac、Taint 等多个优秀开源作品，同时也是 APC、Opcache、Msgpack 等项目的维护者。

在任何一个时代，一款优秀的产品背后必定有一只优秀的团队做支撑。有目共睹，链家在过去的一年里成绩斐然。我不禁好奇，这背后是怎样的一个技术团队？恰好去年“鸟哥”惠新宸离开微博加盟链家一度为业界盛事。藉此机会 InfoQ 再访鸟哥，听他

讲讲链家技术团队的文化、思想与经验分享。

InfoQ：鸟哥，很高兴您再次接受 InfoQ 的采访。首先想请您介绍一下刚加入链家时，链家的产品与研发现状？您加入之后主要做了哪些工作？

鸟哥：时间过的很快，我是去年九月份加入链家网，到现在已经快半年了，目前我负责链家网整体的基础技术方面的团队，包括外网、前端、移动端、测试、运维、DBA、基础平台等团队。在我来之前，其实链家网已经建成了成熟的研发团队，很多同事都是来自BAT等互联网公司，各种产品、研发、测试、运维流程也都比较成熟。

虽然看起来去年一年链家网看起来成果斐然，线上的流量、口碑、美誉度都还不错；但其实链家网其实才成立一年多，很多事情我们其实只是刚刚起步。今年我们会在品质上投入更大精力，让研发团队的效率变得更高。所以现阶段我主要的工作就是提升研发效率、组建团队、寻求人才——。这也是我这半年来主要做的的工作。

InfoQ：链家研发团队有将近200人。经过这段时间的融合，在您看来，链家作为传统行业（多数人眼里）的企业文化与您之前所在的新浪微博等互联网公司的企业文化有哪些不同？

鸟哥：我们研发团队已经朝着

300人的规模快速前进了。确实很多人眼中链家是一个传统行业公司，这也是很多来面试的朋友会提问的问题，其实链家网是一个纯粹的技术型的互联网公司。从我们自己的感觉来说我们的文化是一个轻松、自由、简单的文化氛围，并没有和我之前待过的公司有什么差别。可能唯一的差别就是，我们目前还小，人与人之间、部门与部门之间，还是相对比较简单，公司政治会少很多。

我想原因可能是，一方面是我们团队的人员绝大多数都是来自互联网公司，大家价值观都比较相近，自身也习惯了互联网的文化，不自觉的就复制了这样的文化和氛围；另外一个方面就是因为链家网是链家集团的子公司，集团给予了很大的期望，也赋予了很大的自由度。

InfoQ：除了企业文化方面的差异，在具体业务形态之下，链家现在的产品与系统架构是怎样的？这与您之前所做的迥异之处有哪些？

鸟哥：我们目前的产品，主要分为面向用户和面向经纪人的俩大部分，从业务上又分为二手房、新房、租房、金融等多个方向。大家可以看到的都

是面向用户的产品，比如链家网、掌上链家，还有一些大家看不到的比如 Link，这个是给经纪人的作业系统。

系统架构这个我不确定从什么角度描述。简单来说，以链家网为例，我们使用混合云，有自己的数据中心，也使用云服务。最基础的楼盘词典，这个是链家网 100% 真房源的基础，它的目标是记录全国所有的楼盘信息，房屋信息，类似美国的 MLS。然后有搜索、推荐、图床、数据挖掘、日志统计等支持系统，还有话务系统、商机系统等通用业务系统。

其实从技术的角度来说，和我之前所做的没什么差别。只不过以前我更关注性能，我们现在更关注系统的稳定、流程的合理。

InfoQ：在上一次我们对您的采访中提到，这次跳槽意味着您从研发岗位转到管理岗位。这段时间您在管理岗位上有哪些心得可以分享一下吗？

鸟哥：恩，是的。我现在在工作中基本上都是一些协调、管理、推动的工作。

我感觉，管理岗位，管的事情相比理的事情，我还是认为理的事情更重要一些。想起那句话，“21 世纪什么最重要？人才！”，我们努力的寻找人才、招募人才，而这些人才进来以后，工作的开心不开心、个人的最大能力有没有得到发挥、环境满意不满意、有没有感受到提升，他们想要什么、期望什么，这都是我认为都是我的管理工作中最重要部分。

另外一个就是，很多技术人员会有一个习惯，做事的时候喜欢：你做的太慢，我来。或者：你做的不好，我来。但是在转了管理岗以后，更多的应该想：你做的太慢，我要想办法帮助你快起来。或者：你做的不好，怎么样才能让你（做的）好起来。这样我们的团队才会更好。

InfoQ：接下来想跟您聊聊技术社区的话题。PHP 7.0 发布之后，在未来这段时间里，您在社区的主要工作主要有哪些？

鸟哥：恩，PHP 7.0 已经发布了一段时间了。目前整体的应用情况比起前几个版本来说，速度快很多，很

多云服务商也提供了 PHP 7.0 的支持。最大的原因还是性能提升，我们得到了很多正向的反馈。

PHP 7.1 也开始开发了，我们引入了类型推断，基于类型推断也尝试一些新的优化 Pattern，但不是 JIT。老实来说，7.1 工作量相比 7.0 来说小了很多，毕竟 7.0 是一个重构型的项目。所以我现在很多时候都是修修 BUG，邮件中 Review 下 Patch，沟通意见等。这些工作对我来说，都相当于是一种休息和放松吧……我还是很享受，一个人静静待着，攻克一个个疑难 BUG 以后的那种喜悦。

InfoQ：目前国内与很多比较火的语言类技术社区，比如 Ruby China、Erlang、Python 等等。您怎么看待技术社区的未来发展？您对技术社群的理解和期待是什么？

鸟哥：做技术社群是个辛苦活，赢利非常难。所以首先很敬佩这些致力于技术社群建设、推动技术进步的同学。

一个语言、一个工具，他之所以能流行，能长久不衰，就是因为背后

有一个强大的社群。当我们在学习一门语言的时候，遇到的任何错误信息、遇到任何不解的问题，如果都能在网上找到解决方案，那学习的门槛就会降低很多。而这些都是要靠社群去做，所以技术社群对于一个项目来说，至关重要。

另一方面，现在技术开始综合性发展了。一个人不可能单单只了解一方面的技术就够了，包括还有一些系统性的技术知识，而这些单纯的靠原来的社区是无法满足技术人员的成长需求的。所以我们也看到了比如 <高可用架构> 这样的微信群的产生，这种以实际需求出发，倒推技术能力需求的社群，就很好的弥补了原来的单纯社区的缺失。

InfoQ：在您看来，技术社群现在正发展到怎样的阶段？主要存在哪些方面问题？您对此有什么看法和建议？

鸟哥：恩，对。首先要强调是在我看来，毕竟我认知有限，我觉得技术社群现在是在快速发展的初期，很多东西还不成熟，这个不是指社群本

身，而是指我们的参与者。

不少人热衷于追新，比如新模式、新框架、新概念，但是对于基础重视的不够。我也曾经有过这样的迷茫，我觉得当时我可能是武侠小说看多了，会期望我学会一门绝学，就可以独步武林。对于少林苦修几十年才能有小成的做法不屑一顾——总觉得有捷径可以走，只是自己没找到，所以痴迷于求仙问药。但现在回头来看，我并没有找到什么捷径。

我觉得技术社群应该负责起让大家回归理智的职责。技术本无优劣，应该引领技术人员专注于基础，关注基本功的修养。

另外一方面不成熟，就是目前语言之争盛行。这个不仅仅是国内，国外也一样，可能就是文人相轻吧。但是作为受灾比较严重的 PHP 社区的一员来说，我对这样的风气很是不屑。更可惜的是，如果是一些初学者有这样的想法也就罢了，但是我看到不少从业已经很久了，在领域也有所建树的人也整天参与互相攻击、讥讽、贬低别人，让我有点沮丧。如果这样的人，也能引领一个社群，那这个社群会发展成什么样，也就可想而知了。

所以我还觉得，我们技术社群也是要弘扬一些正能量，拒绝以炒作语言之争或框架之争来搏眼球的做法。更多的倡导大家博学众长、综合应用，学会根据情况不同，选择最优工具。

InfoQ：最后一个问题是，技术人员该怎么通过社群提升自己的技术和影响力？

鸟哥：想起来之前看到的一个笑话，大意是说，一个人跟老板说要涨薪，说自己已经十年经验了，但是他老板说，你不是十年经验，你只是一年的经验重复了十年。恩，对于技术人员来说，要多和业界接触，多参与业界的一些活动，认识不同的人，了解一些新的技术，一些新的模式，并选择合适的应用于自己的项目中，从而让自己保持进步是很重要的。

至于说影响力，我觉得我们中国人传统的一个观念就是“酒香不怕巷子深”，但我并不认可这个理念。酒香首先要被认可、被检验，不能你自己认为自己酒香，就孤芳自赏，不被人认可就怨天尤人。你就需要多发声，努力让你的声音让更多人听到，这样才能让你知道你到底是不是真的酒香，你才能影响别人，才能打造你自己的影响力。

另外一个就是很多人认为“言多必失”，所以他们不愿意过多的在人多的地方表达自己的观点，害怕自己犯错。而我也不认同这个观念，如果你不表达你的认知，你又怎么知道你是错的呢？另外也不要害怕错误。拿我自己来说，我个性比较张扬，从小就特别喜欢分享，经常会去研究一些稀奇古怪的东西给我的朋友、同学们分享。工作以后也是这样，我08年开始写博客，把我的一些心得分享出来，这个过程中会有自己认为错误的地方，

别人指正出来，我就虚心接受，立即改正，这样让我自己的进步也很大。

总得来说，要提升影响力，首先你自己需要有能力，有实力，这就需要我们不断提升自己。其次你要多发声，现在比以前好多了，以前最多也就是写博客，混邮件列表；现在呢有微博，有微信等，传播起来方便多了。

每年都会有那么几个朋友在社区崭露头脚。我相信未来还会有很多，下一个新星是谁呢？也许就是现在在努力提升自己的你。

扫码关注回复“鸟哥”

鸟哥详情全知道



专访百分点刘译璟： 创业公司是否是技术天才的理想国



作者 杜小芳

刘译璟，博士从北大毕业后，在百分点创业开始便加入了该公司，成为公司第一个技术人员，一路携手，互相成就，如今百分点已然是行业翘楚，而刘译璟博士也成为公司的技术副总裁 / 首席架构师。

百分点作为国内大数据技术与应用服务商，希望每一个普通用户都能拥有使用大数据的能力，这种“赋能”也让其成功为 2000 家互联网企业和实体企业提供服务。InfoQ 采访了刘译璟博士，借此了解他的个人经历，以及探询百分点的技术和文化。

InfoQ：请讲讲您的背景和个人经历，您如何和大数据结缘？

刘译璟：我是 85 年出生，山西人。2001 年上本科，就读于南京邮电大学应用数理系。2005 年到 2011 年间在北京大学数学学院硕博连读，跟随裘宗燕老师做形式化方法、程序语义和

验证方面的研究。

读博期间，我主要的工作集中在如何用逻辑（我自己发明的 00 Separation Logic）的方法描述 Java 程序的需求（Specification），以及用一系列数学证明过程验证程序的编写是否符合需求。这是一个非常数学和非常理论的研究领域，在这个领域中接受的训练和研究工作让我受益匪浅，让我具备了严谨的思维，学会了高度抽象的思考方式，以及刨根问底的追究精神，我认为它们是我工作和生活中最好的伙伴。

另外，在北大数学院的六年中，我还学习了许多计算理论、编译原理、人工智能、模式识别以及科学哲学方面的知识，这些知识对我后续的工作起到了巨大的作用，让我看清楚了很多大数据方面的问题和发展方向。

接触大数据是我到百分点以后的事。我是在 2009 年 7 月 2 日加入百分点的，是公司成立的第二天，加入百分点完全是机缘巧合。在研究生阶段，每到假期我都会找一些兼职赚赚外快，做过一些的 Web 和桌面类型的应用，也有自然语言和机器学习方面的项目。2009 年暑假，我在北大未名上看到了我们董事长发的百分点的招聘帖，简

单电话面试后我就上船了，成为了百分点第一个技术人员，负责了第一代推荐引擎的研发工作，算是正式从事大数据方面的工作。后来，随着公司业务的扩展和深入，我对大数据的思考越来越深入，逐渐形成了自己的一套理念和方法论，包括什么是大数据、大数据的核心技术、大数据在信息化体系中的位置、如何运用大数据解决业务问题，等等。

在我看来，大数据最核心观点的是要利用数据化的理念和技术对现实世界建模，构建一个数据世界，基于这个数据世界再去构建相应的业务系统，从而实现智能化的应用。这也是 IT 时代与 DT 时代最大的差异，如图 1 所示。

简单来说，将一切转化为数据和数学模型，就是大数据要做的事。这是我对我对大数据的认识，更多这方面的对话题我们后续再谈。

InfoQ：能介绍下百分点科技产品线发展历程是怎么样的吗？各不同产品之间是否有使用共同的技术？

刘译璟：2009 年公司刚成立时，我们唯一的产品是基于 SaaS 的个性化推荐引擎，主要面向的客户是电商网站。据我所知，百分点是国内最早做

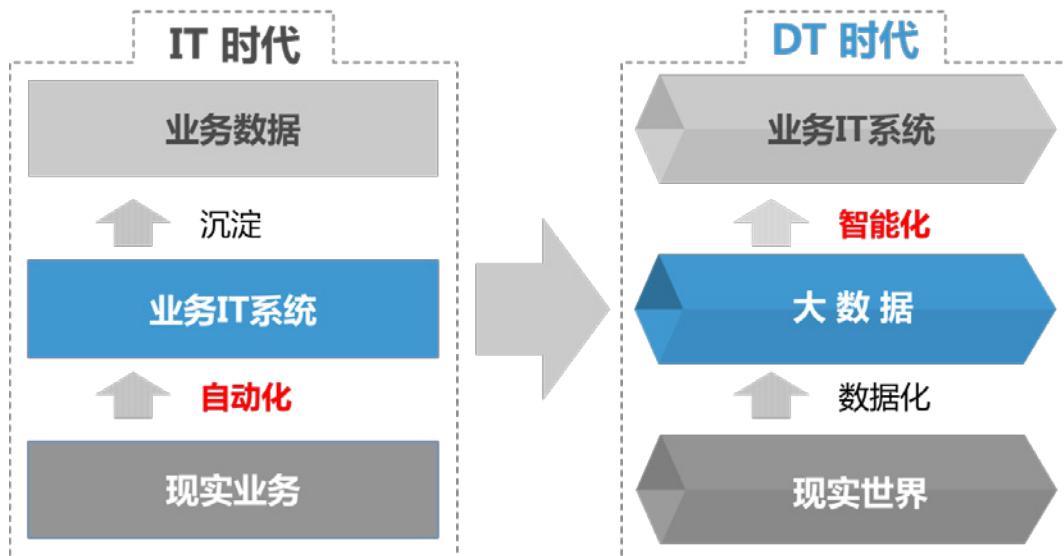


图 1

第三方推荐引擎的企业，并且我们的产品从一开始就强调实时性。实时性体现在两个方面：第一，任何一个网站接入我们的引擎后，几秒钟后就会有推荐产生，而不需要提前准备数据让我们分析；第二，网站上的用户行为会被实时处理，反应到后端的模型和下一次推荐结果中。实时性让我们的推荐引擎能更及时准确的把握消费者偏好，作出令人满意的推荐。能做到实时的原因是我们从一开始就采取了流式计算和批量计算混合的 Lambda 架构（这个词是 2013 年 LinkedIn 提出的，意思是“人字形”的架构），并且我们用把大多数的推荐算法改造

成了增量计算的方式。即使 7 年后的今天，很多企业的推荐引擎还没能做到实时性。

到了 2011 年，我们的推荐引擎不只是服务电商，而是拓展到了媒体、团购和导航等业务和行业。我们发现，仅仅依靠一系列的推荐算法很难进一步提升推荐效果，也很难做到多元化的推荐。因此，我们对推荐引擎做了一次大升级，改造成了图 2 的四位一体架构。

在这个架构里：场景引擎用来计算消费者所处的业务场景，我们要分析出消费者处于哪个阶段，好有针对性的进行推荐，它就像是部队中的侦



图 2

察兵；算法引擎可以比喻为部队中的后勤，它的主要任务是不停歇的处理各种商品、用户和行为数据，计算出用户的偏好标签、用户间的相关性、产品间的相关性、用户对产品的评分、各种类型的排行榜等等；规则引擎是业务的核心，它的任务是根据网站的业务目标和用户场景，选择合适的算法结果，并拼装成最终的推荐结果；展示引擎可以看作是进攻部队，它以合适的方式将推荐结果展示在消费者面前。

在这个阶段，我们开始引入用户

画像、商品画像、抓取系统等技术，这些技术后来都转化成了我们的产品。

到了 2012 年，我们推出了百分点分析引擎，它是一款 BI 类型的产品，可以电商分析产品、类目、品牌、订单和用户等核心运营主题，帮助电商提升数据化运营能力。

2013 年底，随着互联网的冲击和 O2O 的崛起，越来越多传统企业开始重视经营用户和全渠道经营，一部分走在前面的企业急切的期望具备大数据的能力，例如王府井百货、长虹等等，于是他们找到了百分点。在这样的背

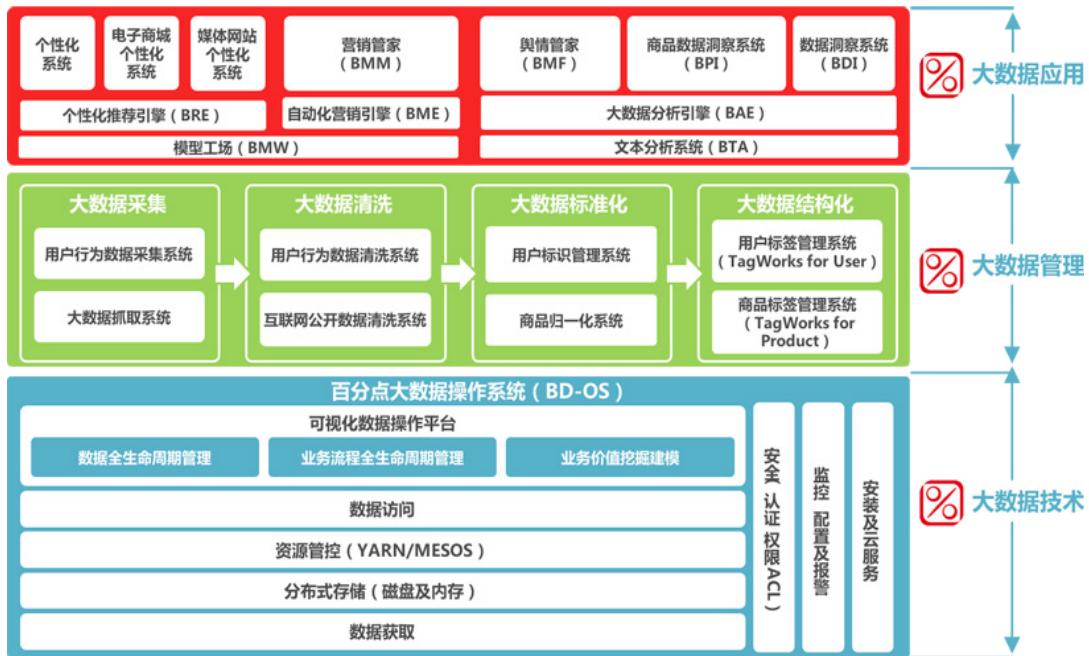


图3

景下，我们将自己内部用到的方法论和技术一点点封装成产品，提供给企业，目前我们的产品体系如图3。

可以看出，我们将大数据能力分成了三层：

- 技术层。它负责提供数据的采集、存储、整合、加工和对外接口等基础能力，就好比工厂中的厂房、水电、设备一样。我们将自己用到的所有的大数据底层技术整合到了百分点大数据操作系统（BD-OS）这款产品中；
- 管理层。它类似于工厂中的流水线、

管理制度。在管理层，我们为企业提供了生产用户画像和商品画像的完整流程和工具，让企业很容易做到用数据来描述自身业务的方方面面；

- 应用层。这是最终的数据产品。我们有三大类应用供业务人员使用，分别是服务型的推荐系统及其衍生品，营销类的营销管家，以及分析类的舆情管家、数据洞察系统。

与推荐引擎一样，所有这些产品都基于经典的 Lambda 架构，它实际上也是 BD-OS 的核心架构，如图 4 所示。

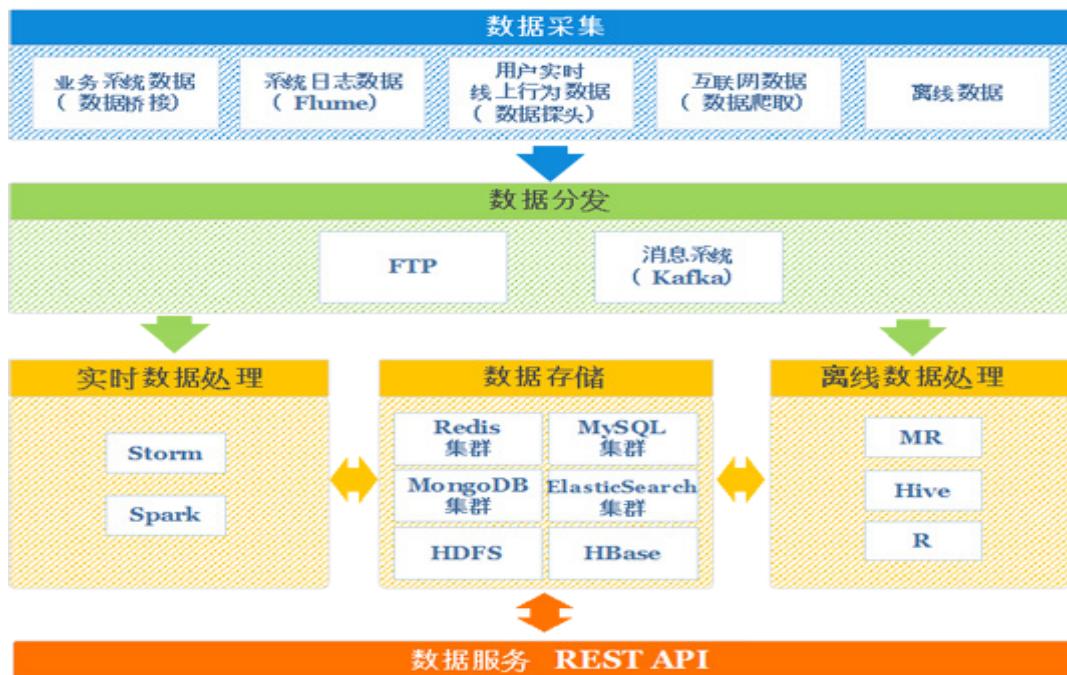


图 4

另外，在这些产品中我们大量使用了机器学习和深度学习、分布式高可用、负载均衡等技术。

InfoQ：百分点的数据来源应该含有第三方数据和自有数据，这两者现在所占数据源的比例如何？对于第三方数据，比起自有数据，总会有点受限，在数据分析时是否带来影响？百分点如何减少两者差异？

刘译璟：我们一般从数据所有权角度出发，将数据分成第一二三方，大致的定义如下：

- 第一方数据，是指企业完全拥有权

限的数据，即数据的产生和使用全部由企业决定。例如企业业务系统中的数据，就是企业的第一方数据。

- 第二方数据，是指企业和合作伙伴共同拥有的数据。例如我在天猫开个店，这些订单数据到底是我的还是天猫的？按我们的理解，这就是典型的双方共有的数据，所有权如何分配是商谈出来的。在使用这些数据时，也要遵循双方共同的利益；
- 第三方数据，就是所有权不归企业，但是可以通过某些途径获取到的数据。例如互联网上公开的网页数据。

它们的分析和应用一般遵循互联网的公开协议或版权。

按照这个定义，百分点大部分原始数据是第二方的，是合作伙伴为了使用我们的服务主动提交给我们的。并且，在我们与合作伙伴的协议中会明确规定好数据的使用方式和范围。

我们在业务中确实需要很多第三方数据，比如我们需要国家行政区数据和天气数据，以便支持各类业务规则。

百分点的第一方数据主要是我们挖掘出各类语义信息、商品分类、商品画像和用户画像等等，它们是提供各类数据服务的基础，也是我们数据集中规模最大的部分。

总的来说，不同所有权的数据获取和使用方式都有差异，我们要做的就是在权限规定的范围内尽可能的挖掘数据价值并使用它们。

InfoQ：如何评断推荐引擎效果，百分点科技采用的是什么检验方法，百分点还会从哪些方面提升效果？

刘译璟：这是一个非常好的问题，也是一个非常复杂的问题。很多朋友都会问类似的问题：“你们的推荐准不准啊？效果好不好啊？”，感觉上应该很简单吧，但是大家想没想过，

什么叫做“准”，什么叫“好”呢？

我先讲一些通用的概念。如前所述，推荐引擎其实是对现实中推荐业务的数学建模。对于一个数据模型，我们评估它的好坏，本质是看它是否与现实相吻合。这个吻合包含两个方面：第一是模型是否反映了现实，就是模型做出的判断在现实中是否真的存在，一般我们称之为准确度；第二是现实是不是都在模型中有反应，就是说现实中发生的事是不是都被模型捕捉到了，一般我们称之为覆盖度。准确度和覆盖度在不同的领域中有不同的叫法，例如在逻辑中一般称为可靠性和完全性，在搜索中叫查准率和查全率，在机器学习中叫准确率和召回率，但本质上它们是一个意思。这样，我们就能看出，“准”和“好”是两个不同的概念，“准”一般是指准确度，而“好”是结合准确度和完全度的一个综合评价。

再来看推荐系统，从学术上来讲，我们可以把推荐引擎简化为输出“用户喜欢的产品列表”的模型，这时：

- “准确”就是“推荐的产品列表”是不是用户真的喜欢。这一点一般是可以评估的，实践中我们可以利用点击率或购买率来代表它；

- “覆盖”是指用户喜欢的产品是不是真的出现在“推荐的产品列表”中。这一点实践中基本上无法衡量，因为我们根本不知道用户喜欢哪些东西！

所以，在学术上，准确度是来衡量推荐系统好坏的主要指标。另外，从用户体验角度出发，学者们来提出了“多样性”（推荐列表中产品类型是否多样）和“新颖性”（推荐列表中的产品是否远离用户的偏好）等指标，用来衡量推荐系统的好坏。

但是！学术归学术，实践中要复杂很多。推荐引擎最终服务的是企业客户，而不是终端用户。推荐是为了帮助企业达成业务目标，而不（只）是服务好消费者！虽然企业的业务目标和用户的目标有很大关联，但这两者之间并不是相等的。例如，利润高的产品往往并不是用户喜欢的。从这个角度来说，学术上的“准确性”、“多样性”和“新颖性”等很难指导推荐引擎的实践，因为这些指标完全没有考虑企业的需求。

在百分点的推荐引擎中，我们有若干种业务指标，例如“点击率”、“下单率”、“客单价”、“购买件数”、“购买种类数”、“转化率”等等，这些

指标都是直接反映企业业务目标的。推荐引擎的目标就是优化这些指标。

如何优化这些指标呢？我们的做法是不断尝试新的模型，并进行 A/B Test。简单来说就是不断的试错，尝试符合业务目标的方向。实际上，这种方法就是卡尔波普证伪主义 的体现。

当然，尝试是要有策略的，不是随意尝试。我们的尝试包括两方法：

- 第一是全局层面的，改变我们对推荐业务的建模方法。例如我们从最早只有推荐算法到搭建成场景引擎、规则引擎、算法引擎和展示引擎，又例如引入用户画像的概念，这些都是根本上的改变和尝试；
- 第二是局部层面的，例如采用新的推荐算法，又例如调整用户画像的计算参数等等，这些都是在细节上进行调整。

所有的尝试都会放到线上进行分流测试，测试效果好的将说明我们在这个方向上的尝试是有道理的，值得进一步思考；效果不好则说明这个方向的尝试不太符合实际情况，先放弃再说。

以上过程我们每时每刻都在做，好让推荐引擎不断接近现实业务需求。

InfoQ：请问您对推荐系统及技术未来发展趋势如何看法？

刘译璟：我更愿意用“个性化技术”而不是“推荐技术”，“推荐”是“个性化”的一个具体应用，而“个性化”是更本质的要求。实际上，“个性化”和“推荐”一直伴随着人类，甚至比搜索一类的需求更加基础。

百分点之所以在 2009 年开始做推荐引擎，就是看到随着信息化的发展，每个人都将深度参与到信息社会中，面对近似于无穷的信息，如何将信息和人有效的匹配将是全社会的刚需和难题，解决不好这个难题我们将深陷在信息暗海中，社会运作的效率也将极其低下。

回过头来看，我们在 2009 年的思考是非常准确的，并且这种信息爆炸的趋势越来越明显。现在很多人都有这样的感受：“满屏都是垃圾信息，反倒是重要的信息都被刷了！”另外，各大网站和企业纷纷将推荐系统作为标配，这也是旁证。可以看出，个性化技术以及基于个性化的服务将会迎来更大的发展。

“个性化技术”本质上是让机器理解人的方方面面，我认为我们现在做的还非常初级。随着互联网和移动互

联网、物联网、可见光通信、可穿戴设备、VR 等技术的发展，人会越来越容易被“感知”和“数据化”；随着脑科学、人工智能甚至量子计算机等技术的发展，机器会越来越强大。届时，真有可能让人感叹“机器更加懂我”。

InfoQ：能介绍下您的团队以及团队理念吗？

刘意璟：百分点现在有员工 600 多人，其中技术人员有 300 多，大致分为基础平台、算法建模、Web 应用几个方面。整体来讲，我们公司的技术人员是很年轻的，富有朝气而激情。同时，我们的学术能力和学习氛围非常好，这也跟核心 Leader 大多是名校的博士有关。

团队理念方面，我们比较推崇小团队、敏捷式的工作方式，鼓励大家主动参与到项目中，成为项目的主人，分享项目的利益。公司也鼓励大家参与到团队管理中，积极提出出自己的想法，只要是合理的 idea 都会被采纳并推广。

配合上述理念，我们还设计了一套激励手段，包含奖牌、奖金、晋升、培训等等，帮助每个团队成员发展。

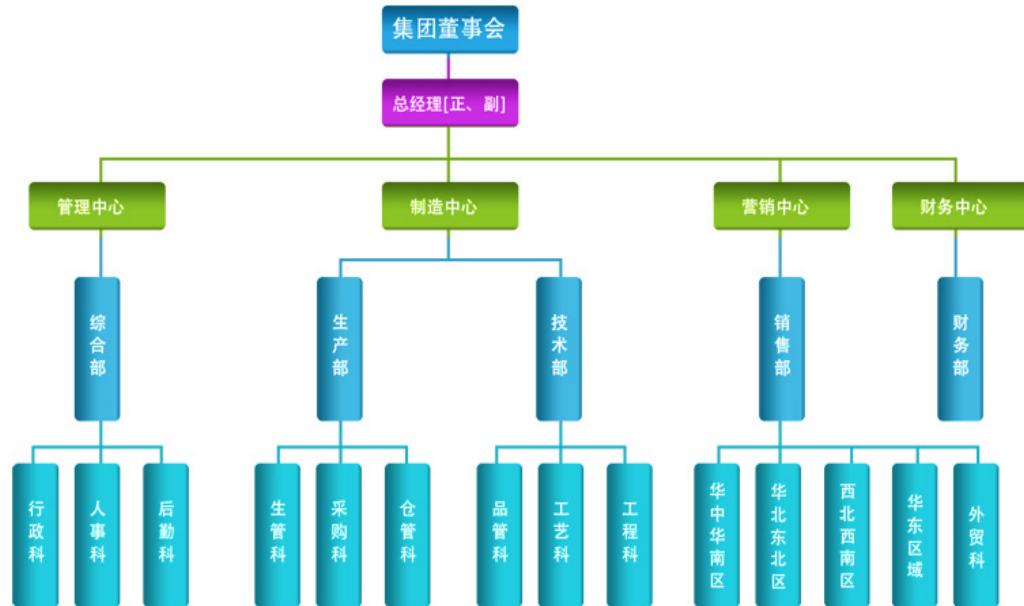


图 5

InfoQ：作为首席架构师，请问有什么感悟或经验可以和大家分享？

刘译璟：在这里我谈谈自己对架构师的看法吧，因为我发现很多人对“架构师”和“技术管理者”纠缠不清。

我们先来看看什么是“架构”。日常生活中，我们处处都在谈论“架构”，例如建筑的架构、世界观的架构、组织架构、甚至我这段文字也有架构。那么这些“架构”中最本质的是什么呢？我认为是有两方面：“成分”和“联系”，并且“联系”是更重要的。以“组织架构”为例，一般它会表现为森林形状，如图 5 所示。

图中每一个方块就是一个“成分”，是架构中的一个功能实体；每条边是“成分”之间的一种“联系”。例子中这个图中的“联系”更多是从属关系或者汇报关系。实际上，组织架构中还有很多“联系”没有体现在这幅图中，例如业务上下游关系、财务关系、人事关系等等。这些“成分”和“联系”的总和，构成了一个企业的组织架构。

从这点出发，我认为“架构”就是某些“成分”及其“联系”的总和，如图 6。

那么，我们在设计一个架构时，无论是软件方面还是其它方面，就需

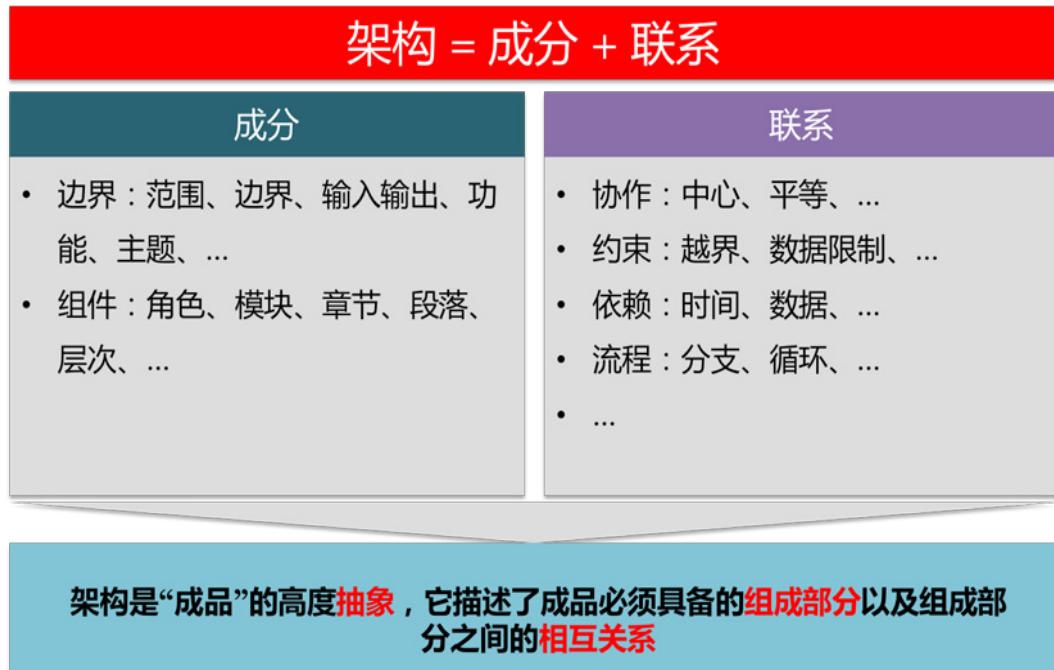


图 6

要区分哪些是我要考虑的“成分”，以及这些“成分”间有哪些需要考虑的“联系”。而且只要考虑这两方面就足够了。

我们再回到组织架构这个话题上，思考一个组织架构是怎么得来的。组织架构是为企业目标服务的，因此组织架构的设计一定从企业的业务目标出发的。在设计过程中，我们按照企业的业务流程，区分出一级部门，规定每个一级部门的功能和边界；而后继续向下拆解每个一级部门到二级部

门，直到规定到每一个员工的功能和边界。这是一个自顶向下的过程。这个过程，可以抽象为图 7，我称它为“关于架构设计的元架构”，因为它也是一种“成分”和“联系”的总和。

从图中可以看出，架构设计是从某一个目标出发的，终结在符合目标的结果上，设计过程分为几个阶段：

- 抽象和设计，得出顶层设计架构。
例如组织架构中一级部门的设计；
- 分治，将顶层架构拆解为更小的子架构。例如进一步设计某个一级部

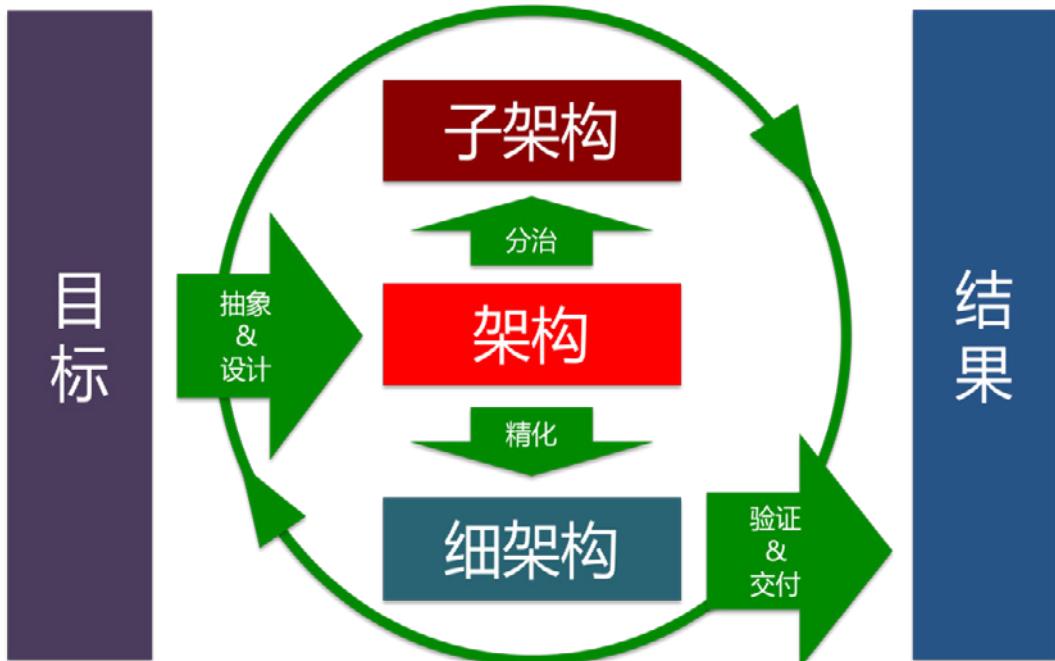


图 7

门的组织架构；

- 精化，规定顶层架构中的某些细节，去除不确定性，成为“细架构”。例如规定部门负责人；
- 验证和交付，检查设计出的架构是否与原始目标相吻合，如吻合则得出设计结果，否则进一步设计。

当一切结束，我们便得到了最终的成果。

“架构师”，便是执行上述设计过程的人，如图 8。

这样，架构师的职责便很清晰了。他关注目标和结果，推导从目标到结

果的方案和途径。而“技术管理者”，更多是职责是找到并明确团队的目标，并且带领团队完成目标，为此他需要架构师的辅助。

最后，我想谈谈架构师的主要工作内容和必备的素养。对于一个架构师来讲，日常的工作应该集中在以下几个方面：

- 分析。首先是分析目标，利用归纳法将目标转换为某一种已知的问题；其次是找出思路，利用演绎法找到解决这中问题的手段和条件。

要想做好这个环节，架构师必须具

**图 8**

- 备丰富的经验，熟悉相关领域的套路和定式，并且能随时举一反三；
- 设计。找出解决问题的方案和实现路径。这一环节中，架构师需要有严密的逻辑思考能力，并且能把握问题的重点和主干，懂得取舍；
 - 选型。将设计方案落地，落实到具体的操作步骤上。这一环节，架构师需要关注细节，关注成本，关注团队成员的能力；
 - 验证。检验产出物是否与目标相吻合。这一环节要求架构师具备极强

的目标意识，时刻思考如何量化目标，如何对标目标，如何检验设计成果。

以上，就是我对架构师的理解，希望对读者有益，也希望与大家进行更多的探讨。

河狸家陈科： 在创业公司做架构师，你需要解决哪些问题



作者 陈科

作河狸家的架构师陈科是千万技术创业者中的一位，在此之前，他曾在浙江电信、阿里巴巴、华为、58 同城任开发工程师及架构师等职位，本文中，作者将会分享他在创业公司遇到的一些挑战以及解决方案，主要内容包括以下几个方面：

- 项目管理的问题
- 业务代码的问题
- 产品需求的问题
- 组织协调的问题
- 技术选型的问题

• 运维方面的问题

• 人的问题

突然想到一句话，人生若只如初见，找对象如此，在公司干活也是一样。

在你加盟一家初创公司的时候，总是豪情万丈，自信心满满，但是问题的出现总是那么的突然，没有一丝丝防备，创业公司甚至没有大公司的蜜月期，你就会面临很多问题。

首当其冲就是项目管理的问题。

创业团队为了追求小快灵的模式，很多时候牺牲了项目本身的科学管理

部分，例如项目计划倒排，任意变更需求，随意封闭开发加班加点，甚至压缩测试工期等。

当然，我们并不是来抱怨问题的，而是想如何更好的解决它，我觉得作为一个技术人员合格素养的一条就是敢 say no。很多问题都是因为畏惧权威，过度承诺导致的。

另外，也需要尊重科学，重视项目里程碑，杜绝人月神话这类事情的发生。

我觉得，不管加班也好，砍需求也好，一定要遵守一个原则，就是不能伤害客户的利益，很多时候我们一味追求糙快猛的工作方式，看起来做了很多事情，其实结果却是漏洞百出，应付不过来，最终伤害的还是客户，倒霉的还是你自己！

推荐大家都好好读读 Bob 大叔的《程序员的职业修养》这本书，特别是前面几章。在这里介绍下 Bob 大叔。Robert C. Martin，世界级软件开发大师，设计模式和敏捷开发先驱，敏捷联盟首任主席，C++ Report 前主编，被后辈程序员尊称为“Bob 大叔”。20 世纪 70 年代初成为职业程序员，后创办 Object Mentor 公司并任总裁。Martin 还是一名多产的作家，至今已

发表数百篇文章、论文和博客，除本书外，还著有《代码整洁之道》、《敏捷软件开发：原则、模式和实践》、《UML：Java 程序员指南》等。他最近创办了 cleancoders.com 网站，专为软件开发人员提供教育视频。

《程序员的职业修养》是一本讲“方法论”的书，目标是让大家成为“专业人士”，在这里推荐一个写的不错的读后感，没时间的同学可以先浏览下这个读后感，再去决定是否看。

接下来就是深入业务代码的问题。

由于创业公司发展迅速，所以往往忽略代码构建的科学性，甚至牺牲和忽略设计的过程，完全需求片段导向，系统可以说是功能点的不停叠加，到最后就算是经验丰富的超级救火队员都没法解救了。

这时候，很多人就会说，这还不简单吗？赶紧重构啊，拆分模块啊。说实话，画个几个框，搞几个箭头标注一下数据流向谁都会，但是具体业务如何建模？

任何互联网的业务平台无非逃不过：会员，账户，订单，支付，营销，计费等几大模块。

但是重构不是说画几个框框搞定的，而是在不脱离你业务显示情况下



的合理设计，例如做营销，你不能只是简单构建一个营销的规则引擎就完事了，也不是做一个抽奖或者优惠券的系统就是把营销的事情做完了，而是需要去分析：

从上帝视角（系统整体视角）你的营销系统摆在什么样的位置。

从整体上走通营销的规则和流程。

套用目前和未来一段时间内有可能会出现的需求，是否可以在这套模型下可以走通的。

设计营销系统本身，站在营销系统的视角进行建模。

在设计完营销系统核心部分后，安排开发工程师进入营销系统的细节功能点开发，以及周边系统的接入。

再例如会员系统，有些有主子账号的概念，这样又会影响订单系统是否需要体现主子账号的关系，甚至又得影响营销，我买一送一这样的活动如何搞，账单如何体现等等？

所以，建模和重构没那么简单，也不是画几个框框就能解决你的问题的，你得深入业务，围绕业务，并且建模绝对不是技术人员的职责，而是运营，产品，市场，技术等大家都需要达成一致甚至深入理解的。

推荐大家阅读下，Eric Evans 写的《领域驱动设计：软件核心复杂性应对之道》一书。

接着是产品需求上遇到的问题。

你会发现产品经理有时候也不能

很好的从产品的整体角度去产品，仅仅只是把业务方的需求过滤和理解了一遍就交给了开发，这样，你就会遇到很多自相矛盾的业务逻辑，甚至影响到用户体验。那么，这个时候你会怎么办呢？是选择做鸵鸟把头埋到沙子吗？还是选择帮助产品经理一起来分析呢？

很多时候，产品经理会抱怨，自己只是个传声筒，老板找他，运营找他，市场找他，那么多事情，根本没法系统性的去梳理产品需求，结果导致需求总是零星的提给开发，结果上线之后漏洞百出。最后背黑锅的不但是产品经理本人，还有开发和测试，客户会说，这是什么垃圾技术开发的。另外，老板不了解情况，也会说，花那么多钱，养一个技术团队，结果开发出了什么鬼？

我想，在创业公司产品技术部工作，最难的就是产品经理和架构师，架构师需要从产品的角度去审视，去把住命脉，因为这已经是最后一道工序了，不系统性的梳理，那么进入开发就会导致前面的悲剧。

产品经理也得好好思考，虽然前期会很累，确实事情很多，但是累一次总比你无穷无尽受虐最后导致团队

和老板还有客户信心丧尽的好。

另外，难道产品设计真的很难吗？除了整个业务模式的玩法你跟别人不一样外，整个产品的形态，其实大多数同类的厂商设计的都是可以互相对标的，比如抽奖怎么玩，你大可以对标一下京东，天猫等产品的玩法，然后结合你自己的特色进行改善。其实我的意思不是叫你去抄袭，而是你要去梳理思路，形成体系化的结构，当你不知道自己怎么弄的时候，可以去借鉴别人，就像读书，读的多了自然就会明白些。我想，大多数重运营的APP，其实还都是相似的。

再接着是组织协调的问题。

大公司的好处就是各司其职，只要有合理的制度和流程，只要你按照规范操作，傻子都能把任务完成（当然这不是说大公司的员工都是傻子。我这里是说流程和规范的重要性，当然傻子虽然能完成工作，但却不能成为大牛，同样，大牛不管在大公司还是小公司，他一样能成为大牛）。但是，在创业公司却不是这样，流程和制度的缺失，导致你会觉得什么都很乱，什么都会有问题，但是你得反过来思考一下，假如什么都好了，还需要你干什么？另外，你加盟创业团队



的初心是什么呢？为什么开头我要说，人生若只如初见呢？

其实，乱就是体现你的价值，把你的经验和优势发挥出来，梳理清楚流程，该自动化就自动化，不合理的就及早提出改善。其实，管理管理，管理的不仅仅是你的员工，也不只是你自己，你还得需要如何向上管理你的老板，如何跨部门管理其他同事等等。拥有一颗大心脏吧，为什么麦迪成为不了科比这样的人呢？

当然，我觉得在协调合作的时候，千万不能把自己的成就感建立在别人的痛苦之上，一个创业团队，大家为什么跟你合作，为什么大家要在这个

公司，都是和你一样心存梦想，奉献自己，不是只有你一个人是最伟大的。是人总会犯错，你要理解别人，严于律己，宽以待人。我觉得华为价值观里有一条非常好，赢则举杯相庆，败则拼死相救。而不是你赢了还黑队友，你败了在责怪队友。时刻心存感恩，感恩你的同事，感恩你的下属，感恩你的伙伴。只有这样，你的事业才有可能长久，否则就算事业成功了，你失去了一群帮助过你的人，你觉得这真的是你想要的吗？

再谈谈技术选型约束的问题。

很多创业项目为了图快和省事，在开发过程中甚至没想清楚为什么，

就随意从网上下载别人的代码和组件，然后写出来的代码五花八门，比如一个 web 工程，有 VO、DO、DAO、DTO、AO、BO、POJO…这么多的概念不把开发给搞成脑残才怪呢。

再比如 JSON 输出，有手工拼装的，有 fastjson 的，有 Jackson 的，总之市面上有什么同类技术，就会在你的代码里一一出现。

当然，我这里不是讲你要选择什么技术，选择什么框架，选择什么软件，而是想说，这种规范和约束性的工作，越早做越好，否则后面重构代价非常大。比如说我们之前的 action 居然都是 servlet 手工输入的，连自

动注入都没法做，后面新人进来就说，你们公司怎么那么垃圾等等，为了解决这个问题，我跟一个哥们合作，光光迁移 1000 多个 servlet，并且完成测试，就花了 2 个礼拜，另外，主干代码人家还得提交，后面还得发布前增量 merge 一次。代价非常大。

架构，还是需要持续的控制，持续的优化，而不只是等到重构的那个点，否则你会很累，团队会很累，老板也不满意，他们会觉得，不是刚刚重构过吗？为什么你们的系统又不行了？为什么又要停下来让业务需求停止呢？

当然，这个事情不是技术驱动那么简单就可以完善的，还是需要得到



你的老板，产品业务方的理解和支持才行。其实架构不是牙膏，挤挤也不一定会有，技术和业务好比公司前进的两条腿，缺一条都不行。其实再怎么业务导向的公司，技术其实还是很重要的，比如很多重运营的公司，假如没有牛逼的运营支持系统来提高运营生产效率的话，人力成本是很庞大的。而且都是完成手工点击装备的低价值工作，还看不到运营的效果是什么样子的。

架构，其实有时候还是和权利有那么点关系的，你既要让架构师干活，而且要出规范，让系统工作的更好，却给架构师明确的权利和约束的职责，有些事情屁股决定脑袋，业务，需求，老板，随时都可以否决你的建议，屁股决定脑袋，很多时候真的是一条铁律。

再接着是运维方面的问题。

现在很多公司都上了云，比如阿里云，腾讯云，所以，很多同学都会觉得运维省事了，没什么挑战了。但是恰恰相反，这是墨菲定律的反定律，越是你觉得没问题的地方，越会出现问题。因为人家可以帮助你维护硬件，维护机器，甚至维护中间件，数据库。但是没法帮助你维护你的业务，也没

法帮你优化性能的瓶颈。

另外，现在有很多第三方的公司和服务，做了很多运维支撑监控，错误排查等服务和工具，假如你的业务发展速度飞快，运维跟不上，你也可以先期使用这些服务来快速定位线上问题，等后期业务发展壮大了，再慢慢发展你的运维团队，招人永远是个大问题，但是我们不能因为这个问题就面露难色，天助自助者呀！

同时，一定要提倡人人都是运维的理念。试想，你自己写的代码，你居然都不知道哪里出了问题，线上到底是什么情况引发的问题，这是多么可怕的事情呀。

不是说上了云，你的系统就不出问题高枕无忧了！很多中间件，甚至一些基础组件，因为贴合你的业务，还是你自己开发的，它们的运维成本并不比维护机器更低。

最后，再谈谈人的问题。

其实，前面讲的这些最终都可以归结为人的问题，或者说不管大公司还是小公司，人的问题才是最本质的问题，天下熙熙皆为利来，天下攘攘皆为利往！

首先你得解决内部沟通的问题，如何沟通，如何平等沟通，特别是创



**心态 比能力重要，
能力 比知识重要，
知识 比学历重要。**

业团队，大家为什么会选择加盟你，为什么又会选择离开你？这是很科学严肃的问题。很多创业团队就会犯类似错误，把人当做工具使，不行就换，再不行继续招聘。

他们从来不想如何培养内部人员，如何让内部人员的效能最大化。天助自助者，假如你自己都没法救你，还指望别人来救你？

其次是招聘的问题，很多公司都缺人，都缺牛人，但是他们缺从来没有认真科学的理性分析，我为什么要招人？我要找什么样的人。我可以负责人的说，很多公司都是在凭感觉招人，盯着简历凭感觉撞大运！

最近遇到好几个创业公司的CEO，都说缺人，需要架构师，但是他们自己却说不清楚招聘进来架构师能解决

他们什么问题，是技术问题？还是业务问题？太可怕了。其实这不仅是因为自己不负责，也是对别人不负责。人进来不仅仅是短期能否解决你的问题，还得长期关注别人在你这里获得什么，找到合适的人是双赢，否则就是双输。

你真的得想清楚，你是想要找的这个人能做成什么事情，还是看重这个人看起来曾经做过什么事情，这可是有本质的区别，因为这恰恰决定了你招聘的态度，这是个严肃的话题，而不是你撞大运的行为！

另外，很多时候，你第一个招进来的技术负责人会决定后面人员的素质，屁股决定脑袋，牛人跳槽也不希望找个二逼管着自己，就像蔡振华管中国足球，就是个笑话，所以，当你和一个大牛把酒言欢，畅谈一番觉得

这个人板上钉钉能加盟你的时候，却在关键时刻放你鸽子，你就该反省，为什么你的团队没法让大牛加盟了。

也许大公司招人只是为了垄断人才市场，形成人才资源的行业壁垒，但是创业团队不能这么干，否则，迟早你会身败名裂，找不到任何一个人。

接下来就是人才培养的问题了，一个人进来之后，人家为什么选择在你创业的阶段加入你？你是否对这个员工具有感恩的心，人家帮你成长了，你是否愿意帮助人家呢？每个人都会在一家公司遇到阶段性的瓶颈，在这个时候，你是一脚把人踢了，还是乐意抱着感恩的心，来给人家成长的机会呢？因为任何一个人，在一段时间内，总有被榨干的时候。感恩的心，不只是对别人，也是对你自己！

最后，就是成就感的问题了，假

如工作天天都是上面指派，每天都是机械的处理指令，我想，任何有想法的人干不上几个月就得离职。当然这个也得看你的团队是否足够精英，假如是组建精英团队的思路，就不会出现这样的情况，但是假如你的团队只是局部精英，你就得考虑如何保证精英人才的成就感问题，而不只是机械的向他们传递指令，否则……

讲了那么多，我是想说，当你豪情万丈打算加盟一家创业公司的时候，今日长缨在手，何时缚住苍龙？却在进入一家创业公司之后，发现这样那样的问题，请你耐心，细心，爱心的去解决。而不只是抱怨。

创业不简单，加盟一家创业公司也不简单，请你认真对待你的人生和你的家人。并不是每一个人都适合在创业公司工作的！

扫这里

和百位架构师共聊架构



ArchSummit

技术峰会让学习交流畅通无阻!

ArchSummit传承经典案例，引领未来技术！

2016年7月15-16日/深圳南山区华侨城洲际酒店
中国·深圳



传承经典

云服务架构探索、发展中的移动架构技术、社交网络等专题带您领略经典行业
的技术变化



跟进前沿

大数据和个性化及高可用
架构专题与您一同探索热
门技术方向的更迭起伏



引领趋势

智能硬件、机器学习、虚拟
现实于您携手观看最新技
术前沿的波澜壮阔

5月15日前 **8** 折购票
团购优惠更多

sz2016.archsummit.com
购票电话:010-89880682



线上专家零距离沟通
想有更多的思想碰撞吗？
请扫描上方二维码
“ArchSummit技术关注”
为您提供足够的专家交流平台！

你应该加入InfoQ 全职编辑团队的 3大理由



可刷脸
蹭饭蹭会



分享就是
最好的广告



跟大牛们混的多了
想不牛都难

InfoQ社区志愿者永久招募中！

6大招聘职位：

1

强力的
技术翻译

2

喜欢四处组织参与
技术活动的
形象大使

3

在任意IT技术领域
信息灵通的
线索发现者

4

深入了解任意IT
技术领域的
专业内容把关人

5

擅长记笔记的
新闻撰写者

6

知道如何
问好问题的
采访者

“

我们是InfoQ编辑，我们是信息的罗宾汉。
现在就发邮件给editors@cn.infoq.com，告诉我们你的专长和意向，我们会将你培养成为一名好编辑：）

”



扫一扫关注InfoQ

EXPERTS RECOMMEND

专家推荐

▶ 程立 / CHENG LI

持续关注InfoQ好多年了。由于工作繁忙没有很多时间泡技术社区，我一直选择坚持精品与原创路线的InfoQ作为获得业界信息的主要来源。当遇到难题时也会到InfoQ上寻找灵感并常常有所收获，可以说InfoQ是我的老师、智囊和朋友，借此机会向InfoQ说声谢谢！

▶ 冯大辉 / FENG DAHUI

InfoQ，技术人都喜欢。几年下来，通过InfoQ网站获得了许多有价值的资讯，通过InfoQ的电子杂志借鉴到很多技术思路，而通过InfoQ举办的数次QCon大会，又结识了不少业界朋友。期待InfoQ坚持自己的特色，期待越办越好！

▶ 洪强宁 / HONG QIANGNING

InfoQ是我获取业内最先进的技术和理念的重要渠道。在InfoQ的帮助下，我也得以与国内外众多技术高手交流切磋，获益匪浅。感谢InfoQ！

▶ 卢旭东 / LU XUDONG

我很早就是InfoQ的注册用户了（哈哈，有好几年了吧，持续保持潜水状态），它一直是我们了解业界研发趋势，学习先进技术和方法的最好平台！在这里还能认识很多志同道合的朋友，InfoQ有潜质成为国内最专业、最大、最有影响力的研发社区！InfoQ的电子杂志更是必看，深浅结合，对实践很有指导性。

▶ 王文彬 / WANG WENBIN

InfoQ办的QCon大会是一个高质量的盛宴，对于最新的互联网技术和最佳实践一直在做探讨。除了邀请国内的牛人，也会有国外的大牛来做分享，对技术人员是一个不可错过的大会。

▶ 杨卫华 / YANG WEIHUA

InfoQ每年遍布全球的QCon大会是技术界的盛会，给业界很多研发方向上的启发，新浪微博的技术架构也从往届QCon大会演讲中获取了不少宝贵经验。

▶ 吴永强 / WU YONGQIANG

接触InfoQ，包括QCon，已经有好几年了，我非常喜欢它的风格，灵动、快速、实用，Moq网站、QCon、《架构师》杂志都能够紧贴互联网技术的发展前沿，带来大量的最佳实践，对我们这样发展中的公司的帮助非常大。希望InfoQ能够越做越好！

▶ 毛新生 / MAO XINSHENG

InfoQ社区是架构师的一流资讯来源，也是大家交流的桥梁。