B端业务RD该如何发展

随着餐饮商家生态的业务发展，我们的团队也在迅速地壮大，在过去的几个月我听到了很多同学关于个人发展方面的诉求和问题，针对这些问题9月份我们组织了一次圆桌会议来答疑解惑，但时间有限很多问题并没有在这次会议上得到解答，做为一个延续，我希望能用这篇文章来继续分享一些我对业务RD个人发展的思考。

先列出一些常见的问题：

> 1. 我是做B端系统开发的，数据和并发的挑战好像不大；业务需求很多，我就像是个搬砖的，没觉得有很多技术提升...如何发展技术能力？
> 2. B端系统开发积累的技能换行业是不是就没用了？
> 3. 我对xyz技术很感兴趣，周围没有人是大拿或者志同道合的同学怎么办？
> 4. 我的个人发展好像没以前快了，如何突破？
> 5. 我想提升软素质(比如面试、沟通、业务理解力)，但是不知道怎么入手该怎么办
> 6. 我看到abc团队的工作好像很有意思，我得到了xxx公司的offer，我要不要去？
> 7. ...

要回答这些问题，我想先分享一下我对RD个人发展的一个整体的思考，在这之后我们再看这些问题，答案就呼之欲出了。

我们需要修炼哪些能力

关于个人发展，我有这么一个职业发展公式:

> $ = T * M * B

其中：

- $可以代表我们从工作中获得的报酬，可以代表我们的职级，可以代表任何我们在个人发展上的期望
- T代表一个RD的安身立命之本 – 技术能力，这个也是每个RD最关注的，所以只要努力一般都成长得比较快
- M代表一个RD发展到一定阶段自然产生的、管理方面需要的一些软技能的诉求
  (如带人、面试、项目管理...)，这个方面大多数RD都有诉求，但工作中又很少被给予像培养技术能力一样多的时间和重要性，所以在这方面的成长往往相对缓慢
- B代表一个RD对业务的理解，因为B相对于T往往显得不是很有"技术含量"，所以往往被绝大多数RD所忽视，所以这方面很多人工作多年依然没有太多成长
- * 代表T、M、B这三个因素是乘积的关系，也就是说这三者任何一个方面的短板都将极大地制约一个人的整体职业发展的高度

那么为什么 $ = T * M * B这个公式是成立的？我有这么一个基本逻辑：

> 我们的工作是一个不断解决问题的过程，只有能解决越来越大、越来越复杂的问题，我们的发展才能不断地向上突破

- 在我们的职业发展初阶，只要我们的技术能力不错，我们往往就能把很多问题解决的不错，此时主要是T这个能力在发挥作用
- 我们因为很好地解决了一些问题，从而被委以重任来解决一些更大更复杂的问题，此时往往单兵作战已经力不从心的，我们需要带领一个小分队来一起解决问题，此时管理技能的重要性就凸现出来了，俗话说的“兵熊熊一个，将熊熊一窝”就是这个道理；此时T * M组合拳的威力开始显现
- 等我们能带一只能征善战的队伍之后，我们要解决的问题的难度进一步上升，此时仅仅靠团队的战斗力(do things right)已经不能解决问题了，我们基于对业务的深刻理解，做出正确的选择(do right things)变得至关重要。有些同学可能会认为，理解业务并做出正确的选择是PM团队的职责，这个观点正确但也不正确。如果我们把自己放在一个将军的角度来看问题，那么技术Leader就必须要能深刻地理解业务，就像一个善战的将军不会带队伍去送死一样，一个好的技术leader不会把自己的团队往不靠谱的业务方向上带，这也是各个公司都对高阶技术Leader有业务理解力的要求的原因。此时T * M * B三位一体的能力体系开始充分发挥作用，进而推动我们的职业发展不断突破

T M B落地到B端业务有什么不同

讲完上面这个通用的个人发展公式，我想再聊聊我对B端和C端业务RD工作差异的理解。首先在M方面，B/C两端RD的能力要求是一样的，所以这里主要讲一下T和B这两个方面(尤其是结合商家生态的业务来讲)：

T – 技术能力要求的不同

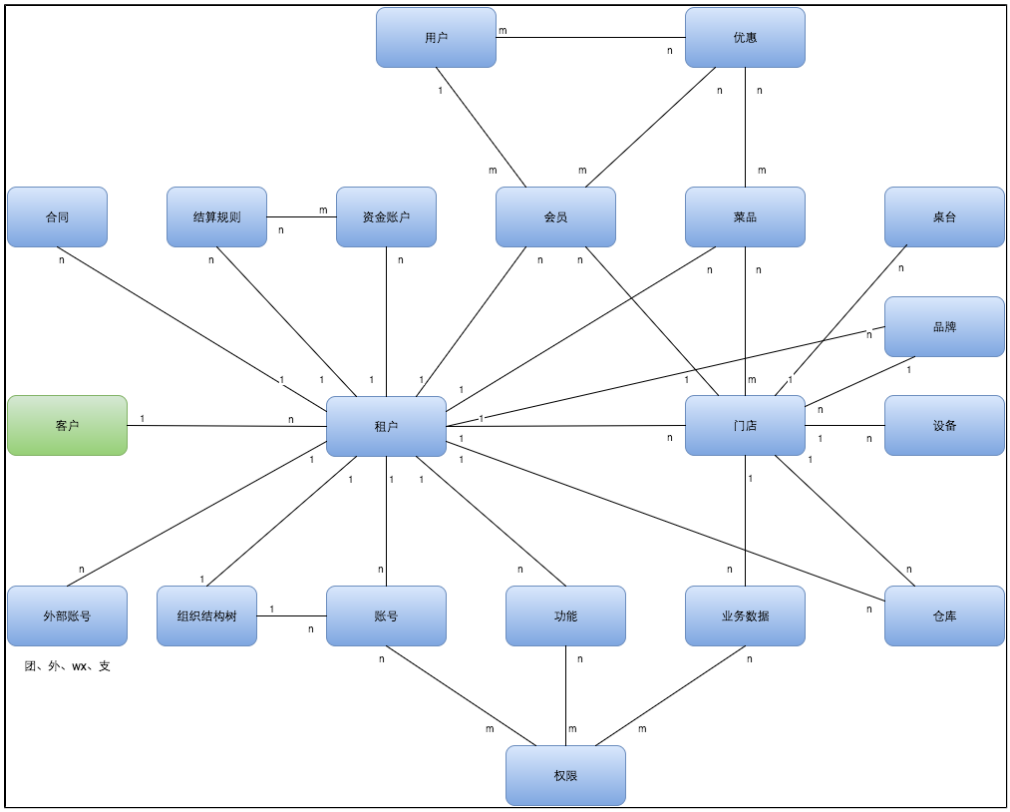
- B端核心业务系统对可用性的要求更高：在C端系统暂时crash了,用户骂两句然后可能暂时先用别家系统了，但只要系统不是频繁崩溃，一般多数用户还是会回来的。在B端，因为商家的核心业务跑在我们的系统上，系统崩溃了他们是没有Plan B的，他们不可能因为系统奔溃而暂停营业，系统崩溃了对他们就是直接的生意损失，只要出一次故障，他们可能就再也不用我们的系统了。所以如果C端系统的可用性要求是99.99%，那么B端系统的可用性就是99.999%

- B端核心业务系统对易用性的要求更高：在商家核心业务的核心环节上，易用性实际是可用性的一个延伸，比如在我们的店内收银系统中，在桌台和点菜这两个环节，如果易用性不做到极致就会极大影响操作效率从而降低营业高峰期的效率，在这方面，商家的容忍度几乎为0

- 生态B端系统的数据量更大：在O2O这个领域，目前线下和线上的交易比例是9:1的比例，在餐饮生态的体系中我们不但能拿到更全的交易数据，同时我们还能拿到类型更丰富的数据，比如像翻台率、人流量、线下支付、原材料、员工...等很多数据都是C端系统无法采集到的，一旦我们的产品在可用性、易用性方面打磨成熟，随着我们的业务扩展，我们的数据量将会爆炸式增长

- SaaS对系统设计提出了更高的要求：C端的系统只需要为一个客户 – 美团服务；而生态的SaaS架构则在软件设计上带来了更多的挑战，相当于从为一个客户设计变成了为多个客户设计，我们如何处理个性化的问题？我们如何在保持通用的同时又做清晰的数据、业务、安全的隔离？我们如何证客户数据资产的安全？SaaS的概念在业界提出了很多年了，这个领域有很多专有的设计挑战等待我们解决

- B端系统对业务建模能力的要求更强：我们拿最基本的菜品管理为例，就会涉及菜品、规格、套餐、分类、价格、库存、营销、门店、渠道、品牌、区域等多个实体，这些实体之间应该是什么关系？哪些关系有一致性的要求？哪些实体应该聚合在一个子Domain中？实体之上应该提供哪些服务？服务之间的依赖应该如何处理？哪些数据应该是只读的？哪些是需要有唯一标识的？这样一个模型如何保证是可扩展的？如何适应餐饮具体细分的11个业态？要回答这些问题就需要有强大的系统建模能力，这也是我们大力推荐DDD的学习的一个原因，业务建模能力是一个RD的核心竞争力　　这张图就是说明复杂性的一个例子:



B – 业务能力要求的不同：

在业务理解上，B/C两端在很多方面都是很类似的，都需要对所做的行业有深刻的理解，但在下面两点，我觉得有很大的不同：

- 唯快不破 vs. 唯稳不胜：以《Lean Startup》中的思想为代表，C端业务做起来往往都是小步快跑，MVP不断迭代。但同样的打法在B端就行不通了，因为商家在核心业务系统上对缺陷的容忍度极低，而且核心业务系统的MVP往往和一个全功能的产品没有太多差别。在我们收银系统的1.1、1.2版本也曾经尝试了C端的打法，但实际情况就是这种简化版的产品在B端完全无法落地，所以我们及时调整打法，从1.3开始做全功能的产品，虽然周期长但是能落地，能真正地让我们开始打磨产品。这个过程是一个血泪的教训

- 体验至上、爆款突破 vs. 流程再造、改变行业：用户体验对C端产品是至关重要的，在这方面我们也有如AB Test等很多工具来辅助我们；在B端比体验更关键的是业务流程，在核心业务系统中，往往会涉及多个角色、实体、流程以及行业中约定俗成的各种惯例，这些领域知识是无法(业务也不允许)用数据统计分析来得到的，我们必须要比商家还要了解如何做好他们的生意，我们才能

真正对他们的业务流程进行优化和改造；在C端一个爆款产品(比如当年的抢车位游戏)往往就能迅速打开突破口，占领市场；在B端占领市场我们要一个商户一个商户地拿下，我们要一个一个业务流程地改造，日积月累下来就能改变餐饮这个几千年都没太多变化的行业；这个改变的过程需要时间也需要我们有耐心，但是一旦这个改造完成，它将对一个行业产生10年甚至几十年的影响(相比C端，谁现在还在玩抢车位)

如何修炼这些能力

说完了我们需要哪些能力，接下来最重要的就是如何练就这些能力了，关于能力的锻炼我有如下几个基本逻辑：

> - 互联网公司的绝大多数业务(研究院除外)都是在应用这个层面做工作，在这个层面基本上只要花时间，人人都可以掌握相关技能
> - 管理是艺术但首先是科学，其科学部分和技术一样都可以被掌握
> - 修炼业务能力、管理能力和技术能力的方法没有本质区别

具体地关于能力的锻炼我有如下几个建议：

- 思考总结：在业务开发中，有些人会发现自己的技能进步逐渐变得缓慢，我们会做很多的需求，但却只是在不断地重复已经掌握的东西。此时我们会面临一个选择：
  1. 去换一个新的领域做做
  2. 突破平台期，把该领域的技能提高到一个新的高度

  这里我的建议是选择#2，原因很简单：从0开始总是提升很快，但真正区分一个人能力高低的却恰恰是哪些需要长期积累后的技能突破。我们短时间就能学会的技能别人也能学会，如果不停地变换方向，虽然看起来我们学了很多东西，但实际竞争力却没有实质的提升。那么我们如何在一个领域能够有突破而不是重复搬砖呢？我觉得要点就在于总结思考，总结思考可以在很多个环节发生
  - 一个需求/项目结束时：这往往是总结思考的最佳时机。乘着我们对项目还有鲜活的印象，总结一下我们在哪些地方做得好，哪些地方做得不好；归纳下掉了哪些坑，为什么会掉这些坑，这些坑反映了我们技能、思考等方面的什么问题；设想一下如果项目重来一遍我们在技术上、业务上、团队上哪些地方可以做的更好；review一下自己的代码，看看哪些地方写的精彩，哪些地方应该重构；这个思考列表几乎可以无限延展下去...俗话说高手都是坑出来的，这些思考就是让我们成为高手的关键
  - 在项目正式启动之前：这一般是最有产出的时机。在动手之前，多琢磨下业务需求之外的非功能性需求；多考虑下如何在自己的项目中落地一些最佳工程实践；多研究一下行业/竞对的解决方案；多思考一下团队每个人能力特点和项目分工如何做最佳匹配；多设想一些可能的技术/业务/项目风险和应对预案...时间不够永远都不是不思考/少思考的理由，磨刀不误砍柴工，动手前多一分钟思考，项目中可能就少加一个小时的班
  - 在项目进行中：这常常是最容易有灵感的时机。在项目中我们会碰到大大小小的很多问题，每个问题点都可能会让我们对某个知识点产生更深的认识，或者会和我们的一些其它知识点产生关联并逐步形成体系，或者发现自己在认知或者知识上的一个盲点，或者产生一些新的启发(比如有一天我突然认识到递归、多态、管理授权在解决问题的思路上无比相似)。我们要真正地把知识内化为自己的一部分，最最需要的就是这些实际项目种大大小小各种问题的刺激
  - 其他任何时候：其实思考可以在任何时候发生，持续的思考带来持续的成长，学而不思则罔说的就是这个道理

- 教会别人：这是判断自己是否真正地掌握了一项技能的一个有效办法。我们只有真正理解一件事，我们才能抓住其最核心的点，也才能以此为中心用大白话给不懂的人说明白。如果我们讲解一个问题不得不用大量的术语、复杂的概念，这往往说明我们实际并没有真正掌握这个问题。我工作中有时说"把我当成一个小学没毕业的人把...讲明白"，"把...讲的让猪都能听懂"，"一句话说不明白就是没想明白"之类的都是这个道理，因此我也鼓励大家抓住一切机会去分享，技术的、业务的、管理的都可以，多多益善

- 多交流不要闭门造车：这个世界上，基本上任何一个问题总是同时有很多人在学习研究。交流的圈子不必被自己的所在的组织结构所限制，如果自己的小组没有人在和自己做相同的研究，那么把扫描范围放到部门看看，再不行放到公司范围看看，甚至放到业界看看，我相信一定能找到很多志同道合的同学；不认识可以请自己的leader帮忙引荐，也可以直接联系通过请教/探讨问题而结识对方，也可以通过关注他的博客、微信等方式来学和交流，甚至还可以自己发起一些学习的圈子/社群来邀请有兴趣的同学加入...而在广泛的交流中我们实际就在创造一个个教会别人或者让别人教会我们某个知识点的机会，在这种教与被教的过程中，我们的学习效率远高于一个人单独学习奋战

- 变化是无尽的，所以关注那些相对不变的：这句话实际是盗版贝索斯对业务的看法，但应用到技能学习方面也是一样的。所有的技术，其存在都是为了解决某个/某类问题的，技术的变化是极其快速的，但技术背后的问题往往是相对不变的。分布式环境下数据一致性的问题问题已经存在了好几十年，这是相对不变的，而这个问题的解法每过一段时间就会出现新的思路/技术框架，这是相对易变的；教会机器认知世界也是一个存在了几十年的问题，这是相对不变的；人工智能领域的技术则在迅速演化，这是相对易变的；我们学习总是从一些具体的技术入手，这部分是相对易变的，但我们不能被这些具体的技术所局限，我们要透过技术看问题，当我们理解了技术背后的问题，我们就能对技术有更深的理解，也能举一反三，快速理解这个问题的不同解法，融会贯通。从这个角度出发：
  - 具体技术栈的选择不是最关键的，拿语言为例，任何编程语言都会消亡(多数还很快)，与其执着是选PHP还是Java，更关键的是要理解语言的特性为什么存在、有哪些优劣、适合解决什么问题、语言和语言如何在特性上做横向对比、语言相关的上下游技术栈为何会存在如何做对比等等
  - 相对技术知识，领域知识相对是不易变的，因为领域知识往往是领域问题的直接映射。拿餐饮领域业务知识为例，吃饭和开饭店这个问题几千年都没有太多变化，我们今天做得只是用不同的技术来重新解决这个问题而已
  - 相对具体的知识，掌握新知识的能力是不易变的。有些同学担心换一个行业我们现在业务知识就白学了，实际上不论做C端还是B端，要真正做好都需要对背后的领域有深刻理解，换行业带来的知识错位的挑战是绕不过去，这个挑战对领域知识如此，对技术能力也是如此，换一个技术方向也会面临知识错位的挑战，我觉得面对这个挑战最好的办法就是元知识，比如如何对一个复杂问题进行建模，比如如何快速熟悉一个陌生的领域，比如如何全局性地进行思考和总结，比如如何系统性地做出一个好的设计...

- 持续学习：按一万小时定律，基本上要在任何一个领域小有所成，10年的时间是需要的，因此持续学习就成了一个必要条件：
  - 有些同学可能会说我其实也不想学习的那么深入，差不多就行了，但事实上没有深度的知识除了有一定装潢作用之外基本都是浮云，而且正因为有如此多的人是这么想的，所以这恰恰成为了我们提升核心竞争力的机会，无论是哪个领域，只要我们能持续10年的学习，我们就会发现业余选手是如此得多而竞争对手竟然是如此得少
  - 有些同学可能会说这个10年长跑太辛苦，我的建议是找一个自己真正热爱的赛道来跑这场长跑。无论是技术、业务、管理，找到一个我们真正热爱的方向，那么在外人看来再大的苦在我们看来实际却是乐在其中，就像一个爱打游戏的人从来不会觉得熬夜打比

赛幸苦一样，一个热爱自己学习领域的人也从来不会觉得10年太长

- 又有些同学可能会说我热爱xyz，但是我却没有机会，我的建议首先是看看自己是真的热爱还是仅仅是不了解所以距离产生美，如果我们分析完发现xyz确实是我们发自内心的兴趣所在，那么我建议及早地开始对xyz的学习并寻找相关的机会，至少在学习这件事上是没有什么限制的。现实情况更常见的却是上面说的距离产生美这种情况，我的建议是对手里做的事情要走心，每天前进一点，然后把这个进步显性化，让自己能不断地看到自己的成长，形成一个正反馈的循环，这样我们就会发现手里正在做的事可能恰好就是自己的兴趣所在，所谓“干一行爱一行”其实说的就是这个，这种效应其实在各行各业也都屡见不爽

1. _____
    - _____
    - _____
    - _____
    - _____
2. _____
3. _____
4. _____
5. _____
    -
    -

•

!