Tower 团队 48 个月远程实践

徐峥

彩程设计软件工程师

> *这是古灵 2017 年 03 月 18 日在成都中生代技术大会上的分享*

我们的历史

彩程设计是 2008 年在成都成立的，CEO 叫沈学良，大家叫他老沈。我 2008 年的时候还在成都电子科大读研究生，名义上虽然是在上学，但实际上从 2006 年开始就一直在跟老沈做事情了，所以 08 年老沈回成都创业的时候，我想反正年轻，而且创业是一件多么自由的事情啊，那就试试呗，谁知道这一试就一直试到了今天。

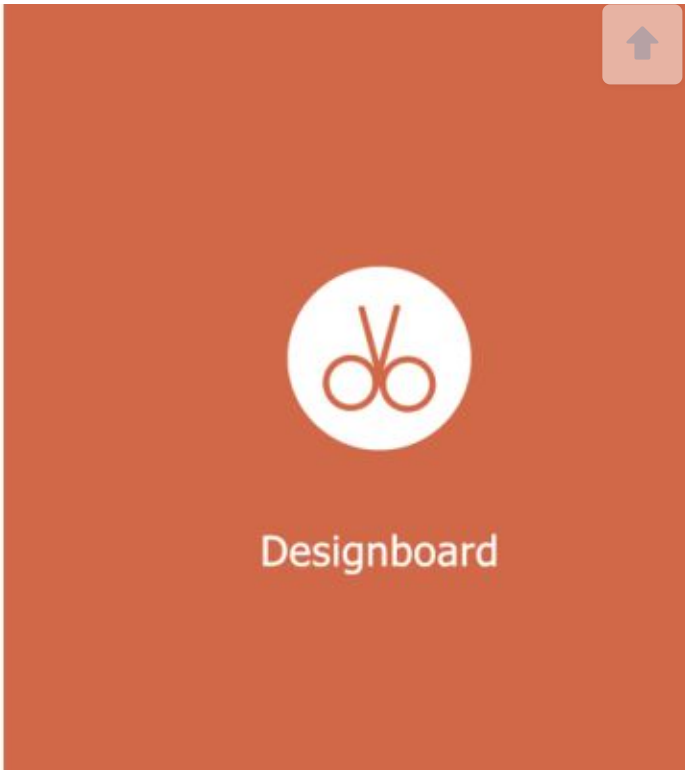
我们 08 年最开始做的事情是用户体验设计外包，那个时候国内还很少有公司知道「用户体验」或者「UCD」这样的概念。最开始我们做的比较多的，是帮一些电信运营商，像亚信联创去做他们业务系统的体验优化设计。传统的电信运营商公司的业务系统多半是跑了很多年的，在底层架构上非常稳定，但是因为那个时候大多数业务系统并不太关注用户的使用体验，这些产品的设计往往都是极其复杂的。所以从 08 年开始我们帮着这些基础服务运营商优化他们的系统，亚信当时的四川、

辽宁、北京 10086 网上营业厅，还有客服系统、网管系统、亚信海外的一个计费系统，都是我们帮助他们重新设计的。在这个过程中我们自己团队也积累了很多产品上的经验，特别是怎么把复杂的东西简单化的思考方式，以及在前端技术方面积累了很多。

后来随着移动互联网的兴起，我们团队从 2010 年开始也为一些移动 APP 做设计外包，比较著名的有成都本地的咕咚运动，还有易到用车，这些 APP 的早期版本的设计都是我们和客户合作的结果。但是随着做外包的时间越来越久，我们发现如果继续做下去，会面临一些问题。

首先是整个行业会逐渐成熟起来，互联网企业会慢慢的掌握设计的技巧，并且因为随时都能了解用户实际使用的感受，所以互联网企业用自己的产品经理和设计师，从长远来看，一定会比外包给设计公司做设计更合适。其次，如果我们要把这艘小船往更开阔的海域行驶，只做设计外包的话肯定是不行的。所以我们团队从 2011 年开始，就试着做自己的产品。

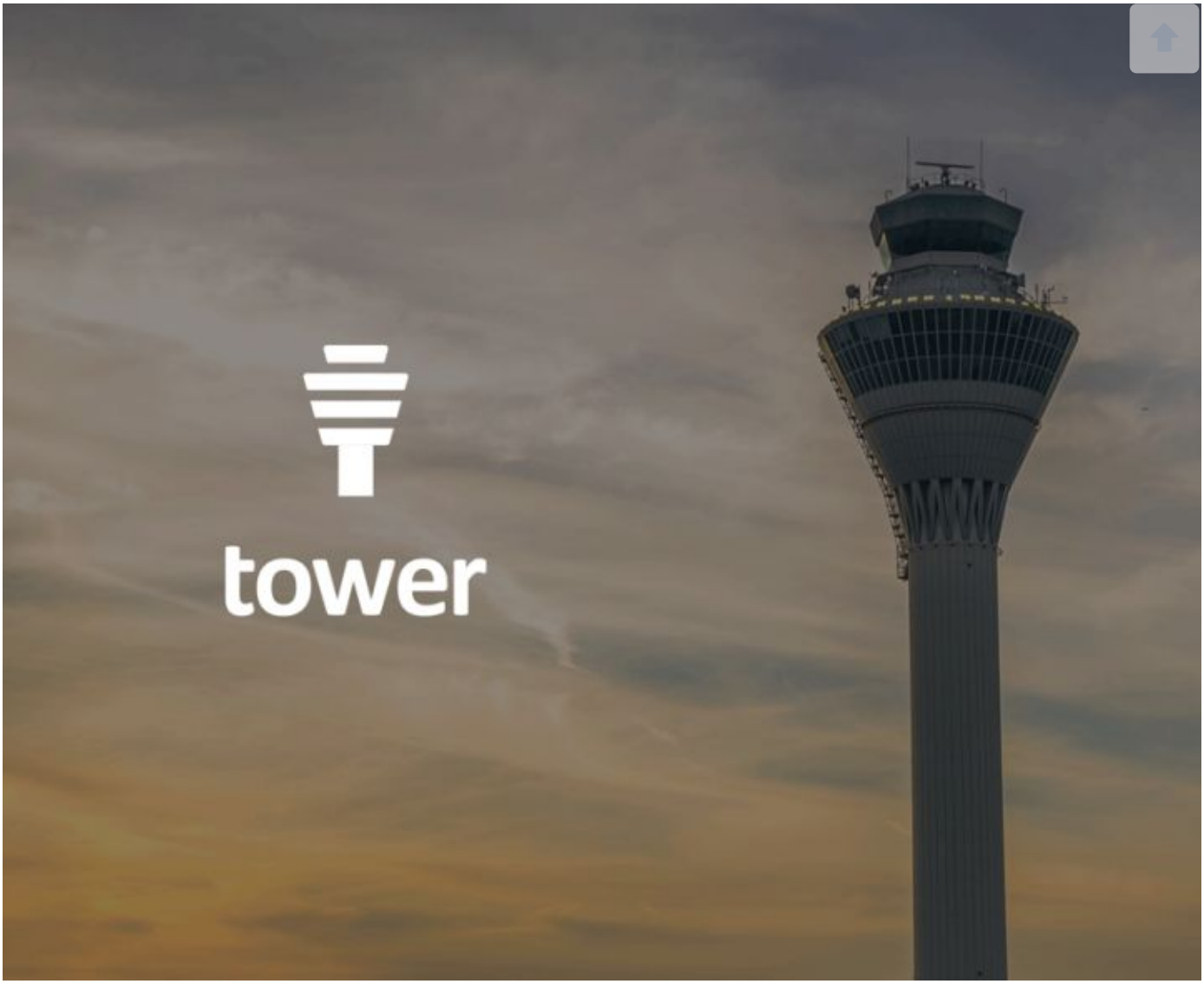
Teamcola



在推出 Tower 之前，我们做过另外两款给团队使用的工具，Teamcola 和 Designboard。前者是一个用于记录每天工作时间的工具，在类似一个日历的界面上拖动，然后记录每天的工作内容，并且团队成员彼此可以共享。这个产品最初只是为了解决我们自己做设计外包的问题，就是我们的一些客户是按照工时付费的，我们要记录设计师每天的工作时间，好最后汇总成工单找用户结算，做了 Teamcola 以后，发现还挺好用，于是干脆就把它开放出来，试了试水。

另外一款叫做 Designboard 的产品，和 Teamcola 诞生的背景也很类似，也是我们自己在做外包的过程中，发现怎么交付设计稿，怎么围绕设计稿进行讨论这件事情可以交给工具来处理，于是我们自己打造了这么一个产品，它可以把设计图上传，组织自己的设计板，然后针对设计图进行评论。

这两款小试牛刀的产品让我们团队积累了做产品的原始经验，我们以前只是帮别人生孩子，现在终于开始学着自己生了。



到了 2012 年下半年，我们推出了 Tower 这款协作软件，这款产品上线第一天的注册用户数，就远远超过了 Teamcola 和 Designboard 的总和，我们也在随后一个月就拿到了红杉的 A 轮投资，于是几个创始人商量了一下，决定停止所有外包业务，转向浩瀚的企业市场领域。

除了从一个设计外包公司转型成了产品公司，2013 年开始远程办公也是我们团队的一个非常大胆的举动，后面我会介绍关于远程工作的原因和经验。先来看看，现在 4 年多时间过去了，我们团队从当时的 10 来个人，成长到现在将近 100 人，我们取得了哪些成果。

首先是 Tower 这个产品，现在是中国市场排名第一的协作软件，有超过 50 万的注册团队，带来了 700 万的注册用户，这些用户每天在 Tower 上产生大量的任务、讨论、文档和文件，如果把 Tower 比作写字楼的话，这应该是全国最大的一栋写字楼了。

另外，我们从 2015 年开始打造的另外一款人力资源类软件「知人」，去年下半年正式发布，这款产品相比 Tower，面对的业务场景更加复杂，为了能够做到自动计算工资，我们做了考勤、请假、公出、加班等出勤类产品的功能，还做了和工作流产品类似的审批流功能，还做了电子合同签署功能，更不用说基础的人事、社保公积金、期权功能，以及对接的滴滴打车、企业微信、摩卡招聘等等第三方服务。

从产品的复杂程度上来讲，知人是一款远超 Tower 的产品，从推广到市场的结果来看，知人也是目前市面上产品形态最完备的人力资源 SaaS 产品，能够支撑从几十人的公司到几千人的公司的人力资源日常事务。而这个结果，也是整个团队在远程工作的模式下做到的。

为什么要远程工作

有三个驱动我们远程工作的原因，一是产品原因，二是不愿意再花时间在一些麻烦事上，三是最重要的一点，就是因为彼此信任。

2013 年我们决定要开始远程工作，当时最大的一个初衷是想要通过这种方式，把产品打造得更好。这么说挺奇怪的，为什么不是大家聚在一起没日没夜的加班，而是干脆要分开来做事情呢？这是因为 Tower 这款产品本身的特点决定的，我们在 2012 年的时候冒出来了一个大胆的想法，就是如果一款协作类型的产品，能够完全支持一个远程团队的日常协作的话，那么它对于那些天天聚在一起的团队来说应该更不在话下了。

种瓜得瓜，后来的事实证明，虽然我们没法证明 Tower 对所有类型的团队都适用，但是它确实是对于远程工作者最为友好的工具，我们是最早支持 Email 回复、微信公众号、视频会议的协作工具。

另外我们选择远程，是因为创业那么多年，我们换过好几个办公场所，每次换个地方就要折腾网络、桌椅、吃饭等各种麻烦的问题，更别提花在路上奔波的时间。我们在 2013 年初租的最后一个

办公场地快要到期的时候，突然发现要是继续待下去，大家都觉得有些审美疲劳，但是如果要想找新的地方，又会是一件劳心劳力的事情，加之我们团队最为欣赏和喜欢的一只外国团队 37signals 也是远程办公的模式，所以我们就开始琢磨这事儿的可行性了。

最后，前面跟大家啰嗦了半天公司的发展历程，因为我们从 2008 年到 2013 年，核心团队都在一起工作，5 年时间让彼此变得非常默契与信任，我们知道各自的长处和短板，核心团队在一起走了那么长的时间也没有人选择离开，说明大家的长远目标是高度一致的，加之当时的主要核心成员都在同一个城市，也觉得随时可以找个地方聚在一起，所以就更加觉得远程这个事情没什么可担心的了。

于是 2013 年春节后，我们就各自解散回家了。哦，我没有回家，而是买了张机票飞到英国去待了 3 个月，彻底远程了。

远程工作的好处

那么，这么多年的远程工作，给我们带来的好处是什么呢？我们团队的远程工作应该划分为前三年和 2016 年这两段来看。

从 2013 年到 2015 年这三年，我们团队的人数增长是缓慢的，因为协作类型的产品相对来说并不复杂，我们也不想用搭建销售团队的方式来获取收入，因此总是想保持小规模的状态来打造产品。这个时候远程工作带给我们最大的好处主要有两点。

首先不得不承认的是，远程工作会让个人的生活质量得到极大的提升，我在远程工作的前几年，基本上每天的生活可以规律得像机器一样。每天早上 7 点左右起床，走路 5 分钟去家对面的健身房游泳一个钟头，然后回家吃早餐，8点半左右开始工作，中午 12 点在家做午饭，然后小睡一觉到下午 2 点左右开始继续工作，工作到晚上 7 点，走路 10 分钟去家附近的一个社区图书馆看书到 8 点半，或者在家看部电影，结束以后处理工作到晚上 11 点，上床睡觉。

每天早上从健身房回家的路上，看到匆忙上班的人群，想着自己再也不用去挤公交地铁的时候，幸福极了。

其次就是工作上的，「专注」是远程办公能够获得的极大的好处。如果是一帮人聚在一起，难免时不时会被打扰，周围的环境也容易把你的注意力吸引走。远程以后，团队在怎么提高效率方面做了很多努力和尝试，这些努力里面就包括怎么样让每个成员都能更加专注的完成任务。远程所提倡的并不是「在家工作」，更不是「旅行办公」，而是「在任何你认为自己效率最高的地方工作」，所以我们有很多成员会去自己喜欢的咖啡店，或者干脆跑到偏远的小镇上去工作。

2014 年我们曾经开源过的一个文本编辑器 Simditor，就是一个前端工程师跑到丽江去，独自开发了 2 个多月的产物。

远程工作后，我们超过 4 个人参与的会议数量明显减少，以前聚在一起的时候，不管是随便找个同伴，还是同伴可能自己无意识的加入，都会让会议变得很冗长。远程以后因为人不在一起，「说话」的成本变高，导致说话之前做的准备工作就会相对充足，比如会在 Tower 上形成具体的文字，或者能够忽略的不重要的事情，就直接被忽略掉了。这是远程带来的好处。

后来到了 2015 年以后，我们开始转做知人，这个产品如前所述，比 Tower 复杂很多，因此我们开始扩张团队，包括工程师，这个时候，远程的工作性质能从三个方面帮助我们招募到自己喜欢的人才。

首先，在薪资肯定比不过 BAT 的前提下，远程可以加持我们这种创业团队的吸引力，让候选人至少能够时不时的惦记你一下。

其次，那些敢于选择远程工作的成员，一般都是艺高人胆大的，普通的工程师可能更愿意找一个能上班的地方，让自己可以融入到群体里面，但是作为创业者，我们更希望的是招募那些更独立的，不畏惧「与众不同」的伙伴，所以远程也能变相帮助我们筛选候选人，这样招募进来的小伙伴普遍水平都比较高。

最后，因为是远程，所以可以完全的跨地区全国招募，网子撒的大，鱼儿才会多。

如何保证工作质量和效率

远程工作也是工作，我们会不惜一切代价的提高工作的质量和效率，这么多年下来，我们发现对于远程工作的团队来说，有几件至关重要的事情需要做好。

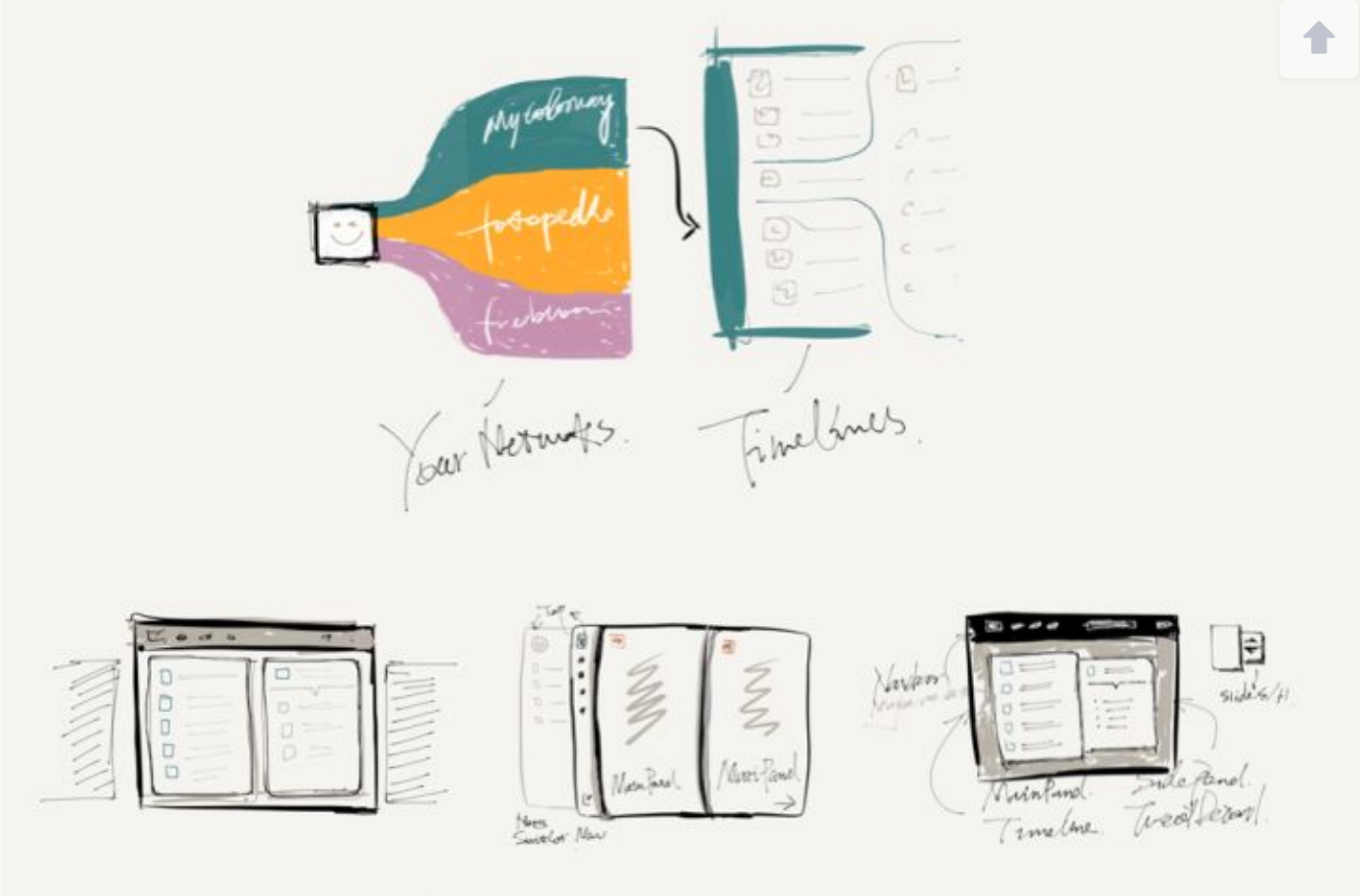
找到对的人

毋庸置疑，找人是创业公司的重中之重，在创业的不同时期，找到优秀的小伙伴都有不同的方法和渠道。

在 2008 年刚回成都开始创业的时候，我们还叫「成都彩程数字科技有限公司」，一个所谓的科技公司，但是能写代码的只有我一个人，所以当务之急就是赶紧扩充团队。我们没有足够的资金和成本去打招聘广告，而且可预见的是这个阶段就算打广告也不可能有什么好效果，这个时候最好的办法就是从身边的熟人「下手」。彩程当时一半的员工都曾经是电子科大一个叫做「栋力无限」的学生社团的成员，老沈甚至还是这个社团的发起人，所以当时我们很快找到「栋力无限」的小同学，请他推荐一些愿意出来实习锻炼的学生给我们。

现在我们团队里的前端技术负责人就是在那个时候加入彩程的。当时这位小伙子才读大四，而且自己曾经休学一年，跑到香港大学去游荡了一年以后才回来的。就像我在前面说的，有这样经历的小伙子一般来说，思想都非常独立，也不害怕去走一条看似风险重重的路，结果也证明了这一点，他现在同样是公司的合伙人，我们产品里所有交互上比较复杂的组件，都是由他带领完成的。

随着公司渐渐发展，在 2010 年左右曾有一段时间，我们在成都本地举办过一个叫做「UCD书友会」的活动，每个月固定的一个周末，邀请一些朋友，以及我们公司的小伙伴，用一个下午的时间，跟一些对设计和产品感兴趣的同行进行交流。这样的活动帮助我们扩大了成都本土的人脉资源，也让很多对当时还不那么火爆的「用户体验设计」感兴趣的人能够进入到我们的视线里。这期间我们认识了后来做 Tower 的交互设计师，那个时候他也只是一个西南民族学院的大四辍学小孩，曾经一个人骑车去过西藏，现在的兴趣爱好是开卡丁车，据说现在已经是四川省卡丁车前三强了。当时看到他手绘的原型设计图我们就知道，这就是我们要找的小伙伴。



让这样的小伙伴加入团队的好处是，他们周围基本上会有一些和他们同样出色的朋友。比如我们的前端技术负责人，带来了另外一个后来的合伙人，我们现在的数据专家，而那位做 Tower 的交互设计师则推荐了一位优秀的视觉设计师，我们在 Tower 里看到的那些漂亮的原画，都出自这位伙伴之手。

后来公司做的产品有了一些口碑之后，我们逐渐能收到一些全国各地的简历了，这个时候我们就需要提炼出对于团队来说，怎么才能从这些简历中找到优秀人才的方法了。

简单地说，我们在整个面试和笔试流程中，对人的考察的核心因素有两点，第一是他是不是足够聪明，第二是他对于我们走做的事情的认同感。

聪明的同伴有两个非常重要的判断标准，一是对产品的理解力，当你在跟一个聪明的小伙伴讨论产品需求的时候，他甚至能先于你想到很多潜藏的问题，以及这些问题的解决方案。第二是他的协作能力，他有没有在 Tower 上拆解自己的任务，能不能够用流畅的语言描述遇到的问题。

而小伙伴对我们团队和做的产品的认同感，是我们判断小伙伴是否合适的第二个标准。有些小伙伴是非常聪明的，但是在和他们合作的过程中，你却会觉得比较别扭。比如你跟他交流的时候，虽然他都知道产品的需求是什么，要做的改进是什么，但是就是投入不够，他会跟你抱怨说，产品的文档没有写好，有些代码写得很烂，但是自己却从不愿意主动去做改进。这其实本质上是对团队做的事情的认同感不够。如果认同感不够，那么这样的小伙伴在一起是走不长远的，他对于产品遇到的短期问题，也不会有足够的耐心。

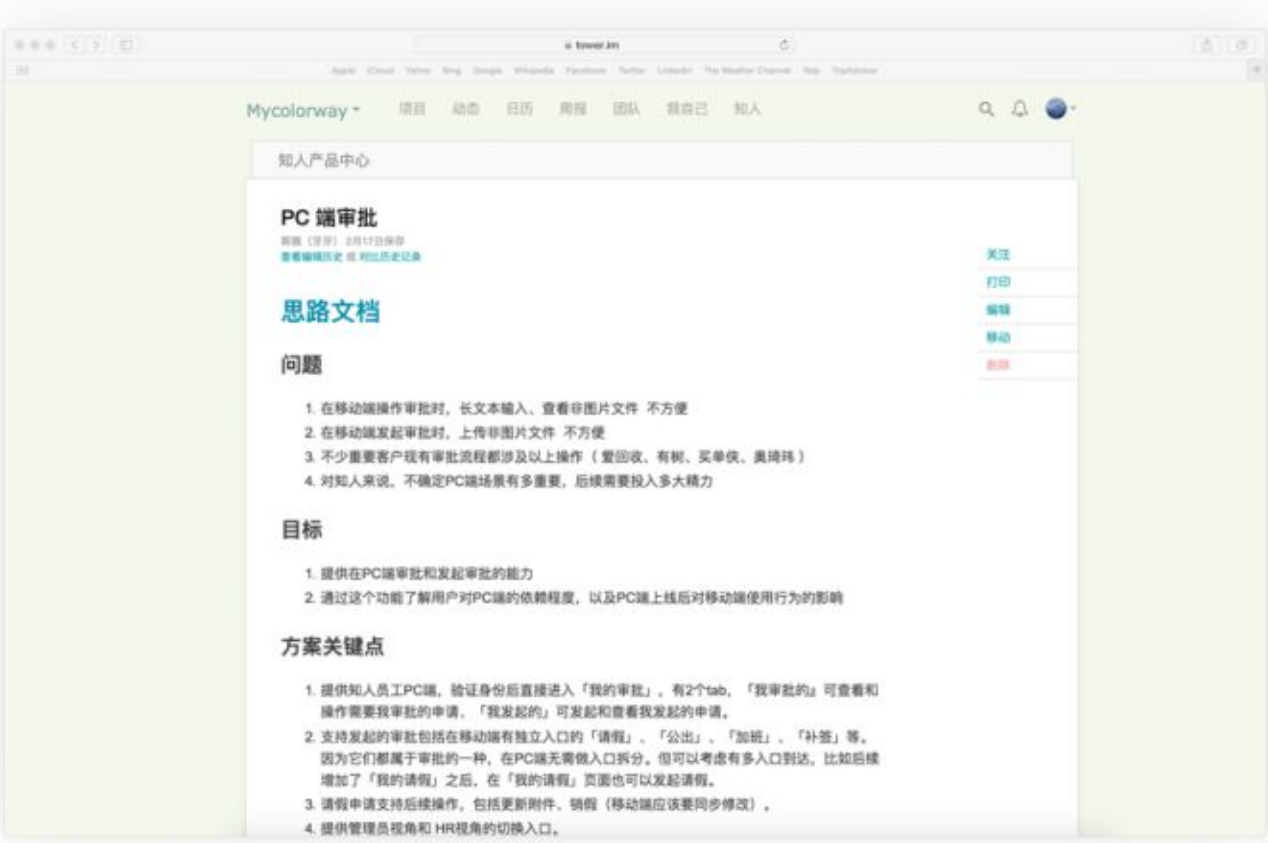
我们并不能总是招到最好的成员，但是结果却一再证明，最后能真正在团队里发挥重大价值的，都是那些最好的 A Player。并且由于我们远程办公的特点，所以对于人的要求更是没法松懈，如果招

募到一个资质平平的工程师，对于团队来说甚至是会带来极大的副作用，因为优秀的人对于那些不怎么优秀的人，总是没什么耐性的。

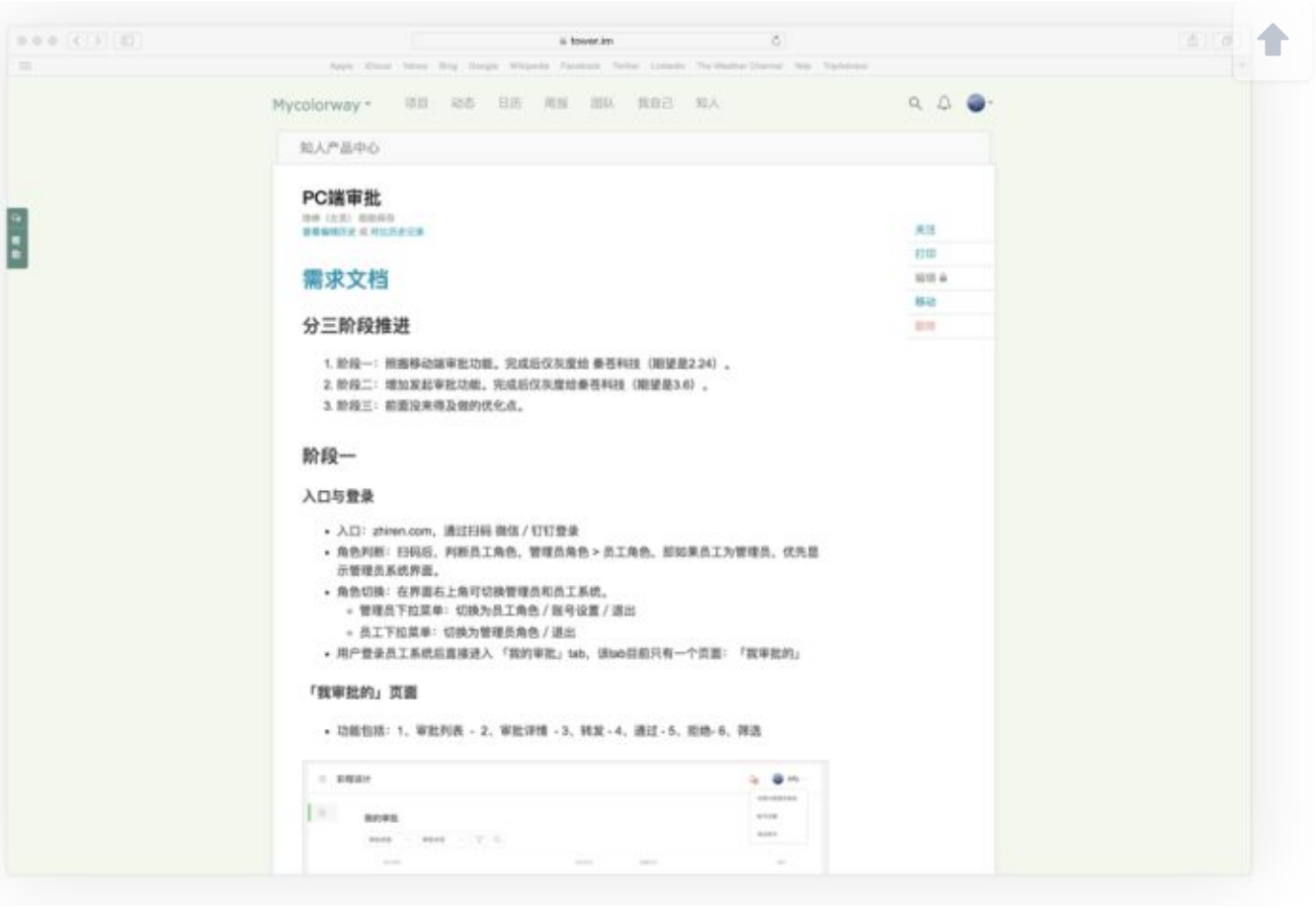
设计设计设计

招到对的人只是第一步，要提高远程团队的协作效率，尤其需要重视产品的前期设计，因为本来远程团队沟通成本是最高的，所以如果前期不围绕功能设计讨论透彻，开发了一段时间以后发现有问题，成本就太高了，这不是像大家坐在办公室里，随时一扭头就可以说，「诶，我怎么觉得这个地方有点问题」然后可以聚在一起解决掉的。

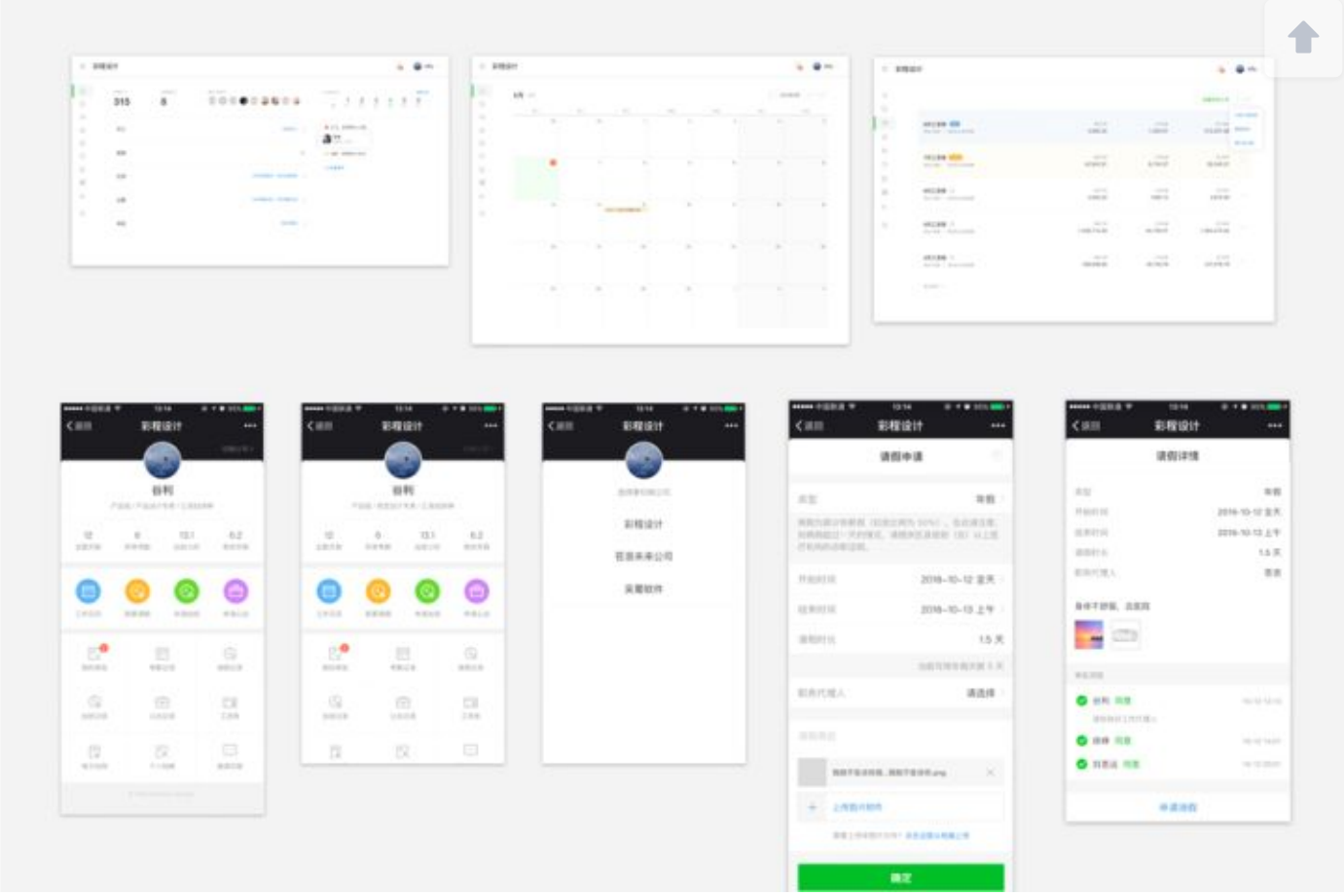
我们团队在开发一个功能点的时候，会要求产品经理至少写两份文档，一份叫做「思路笔记」，这份笔记主要是产品负责人个人对于功能点的思考，要解决的问题是什么，解决的思路是什么，有没有什么想得到的坑，怎么测试，上线以后用什么数据衡量功能是否成功。产品经理会把这份文档发给负责开发的工程师先看一阵子，然后大家会约定好一个时间进行远程视频会议，在视频会议上，每个人提出他们的问题，供产品经理参考。



接下来产品经理会去完成他的第二份文档，叫做「需求说明」，这份文档主要是明确具体的解决方案，根据需要，我们会去做一些具体的设计。如果功能比较简单，工程师根据现有的系统组件就能搭建出来，那么产品经理一般做一个草图即可，但是如果涉及的功能比较复杂，产品经理就需要在草图的基础上，寻求视觉设计师的帮助了。当这份需求说明文档完成后，我们会提前发给工程师再看一次，然后约时间再做一次视频会议。



这次视频会议因为有具体的设计图，所以工程师能更好的理解需求的意图，并且可以提出更具体的问题。像我前面说的，一旦试图省略掉前面的这些过程，那么必然会导致最后开发出来的产品功能考虑不完善，因为前面需要思索的时间，你偷懒了，那些自以为清楚的事情，往往根本就没有想清楚。所以最好的办法，就是用这些能够积累的的东西，把思想过滤出来。



设计这个事情随着做的产品越来越复杂，会变得越来越重要。在做像知人这样的产品的时候，有很多牵一发动全身的功能。比如就算是做一个简单的删除部门的功能，你都要考虑如果以前曾经在这个部门的员工重新入职，要怎么处理，于是重新入职的流程里就需要引入部门设置，然后你就会考虑除了部门以外，还有哪些属性是同样需要重置的，于是本来是在思考提供部门删除的功能，就会延伸为，先规范重新入职流程。问题一开始如果没有想清楚，很容易出现工程师吭哧吭哧做了半天，结果上线以后漏洞百出的情况。

所以，对于远程团队来说，产品迭代里的设计阶段会变得尤为重要，千万不要忽视它。

控制节奏

如果有游泳或者长跑习惯的朋友应该知道，在这些体育运动里，最重要的并不是瞬时的爆发力，而是始终如一的节奏。像长跑 10 公里，你如果能保持一个比较固定的速率的话，跑下来的时候人的感觉是很舒爽的。

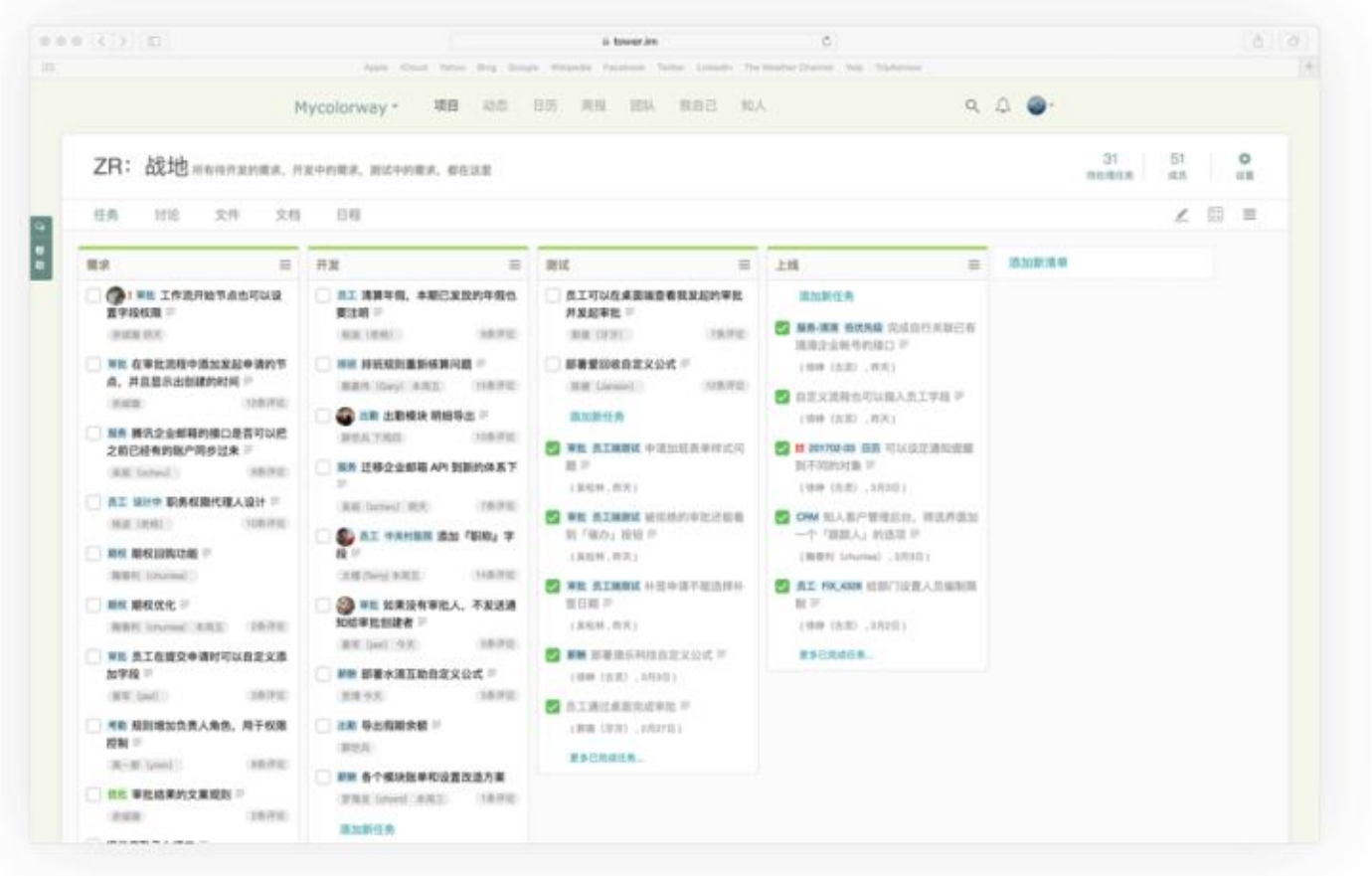
我们在开发产品的时候，也需要做一些工作来保证产品的节奏感。主要是分组和迭代。

在做知人这款产品的时候，我们按照功能进行分组，比如电子合同这个具体的功能，就是两个工程师组成的一个小组，然后多个小组又会负责一个大的功能模块，比如电子合同，是属于人事模块下面的子功能，所以就会有好几个小组会被放在人事模块里面。属于人事模块的这些工程师一般不会做轮岗调整，因为他们做的越久，对人事业务就会越熟悉，也就越容易对产品功能做判断。但是人事模块里面的小组是会轮替的。比如做电子合同的两个工程师，一个季度以后会被调整去做组织架

构，因为这些大的业务模块下的功能是彼此相通的，通过这种方式，工程师可以熟悉彼此所做的业务。

分好小组以后，我们会设置一个迭代周期，比如对于知人，就是两周一个迭代周期。我们的小组工程师会分为 A / B 两个 Team，在一个迭代周期里，如果 A 小组的工程师开发新功能的话，那么 B 小组的工程师就会负责处理 Bug。

在 Tower 里，我们会用两个项目把事情分开，一个项目我们叫做「战地」，这个项目会用看板的形式罗列所有待处理的需求，每个需求会关联到前面说的产品文档。



每个迭代周期开始的时候，开发新功能的工程师，按照需求的优先级认领任务，并把任务从需求池里拖动到「开发」清单里面。我们之所以用两周来做迭代，是因为一般来说，功能开发本身不会太久，因为需求一般都会拆分的比较细，但是开发完了以后，要上线去做测试，测试以后的改进时间也要预留进来，所以两周一次迭代是一个比较合适的长度。

在功能开发迭代之外的工程师，主要是处理另一个叫做「机械师」的项目里面的任务，里面主要是各种 Bug。当然，Bug 不会总是那么多，所以我们并不严格要求维护组的工程师在维护周期里只能做维护，我们希望产品功能能够尽快的推进，所以在维护期的工程师如果需要处理的 Bug 不多，他是会被要求提前去处理战地里的需求的。

如果把产品开发类比为体育运动的话，这种模式的训练强度明显是最高的。

善用工具

彩程设计从创建的第一天起，就在用各种提高我们工作效率和生产效率的工具。

对于远程团队来说，解决沟通是第一位的，我们用 Tower 解决工作计划的问题，在 Tower 里围绕项目、任务、文档和文件来沟通工作的计划，以及推进工作计划。但是也不可能把所有沟通都放在 Tower 上，所以我们团队平时还会重度使用企业微信这个 IM 产品。

对于 IM 工具的选择团队也是经过了很多尝试，从 Slack 还是 Beta 版本的时候，我们就开始尝试使用，到后来转到 Telegram，然后是钉钉，最后现在使用企业微信，对于工具的选择我们发现有两个重要的因素，一是工具要是所有人都能很快获得的。如果一直用 Slack，那么可能只是对工程团队友好，因为销售团队的同事不一定人人都会翻墙，而且产品的全英文也会导致有些同事使用起来产生障碍。所以如果是决定整个团队都长期使用的话，最好还是选择国内的产品。

其次是，这个工具要在产品设计上打动我们，让每个使用的成员觉得产品设计得很赞。而正是因为这个原因让我们最终选择企业微信，因为企业微信的设计更加简洁、舒服。

对于工程团队来说，最重要的几个工具是代码托管、集成测试、代码质量检查、持续发布和服务监控。我们曾经自己搭建过 gitlab 服务，3 年前的一次安全事故让我们明白其实自己搭建服务，并不比使用第三方靠谱平台安全，所以后来我们就把代码管理全部放到 github 了。

集成测试我们使用过 CircleCI 和 TravisCI，这两个工具都可以集成到 Github 里面，后者对前端组件的测试更加友好，前者在产品设计上我个人觉得优于后者，所以这两个平台都是很推荐大家使用的。

代码质量检查，我们使用一个叫做 CodeClimate 的服务，这个服务也是和 Github 的集成非常完美，在做 code review 的时候，可以直接在 Github 上看到一些代码质量的基本问题，工程师可以在 merge pull request 之前对提交的代码做基本的质量判断。

持续发布是我们自己写的一个定时发布工具，会在每天的三个固定的时间把 master 分支的代码发布到生产环境的服务器上面，这个定时发布工具在做代码发布的时候，会检查集成测试环境有没有跑过，只有当测试用例都跑过的情况下，才会做自动发布。如果发布失败，这个发布工具会给固定的一个微信公众号发告警通知，这样运维的同学就能去及时处理。

服务上线以后，我们会用各种工具来监控服务的质量，对于工程团队来说，首先就是各种报错的监控。我们使用 sentry.io 做程序运行错误的告警平台，这个 SaaS 产品允许设置一些 webhook 的回调地址，于是我们会根据告警的代码所属模块，自动在 Tower 的机械师项目里产生分配给对应工程师的任务，非常方便。

另外我们用 Newrelic 做性能监控，目前主要是对 Tower 的监控居多，像去年和钉钉的合作，有一阵子会从钉钉那边带来巨大的注册用户量，Newrelic 在分析到底是哪一层的服务出现了性能问题上是非常方便的。

我们还会用一个叫做 Mixpanel 的工具做用户行为的采集和分析，这主要是对产品经理的帮助更大，使用 Mixpanel 可以记录用户的行为轨迹，帮助产品经理对用户行为进行分析。

代码托管

GitHub

告警监控

Sentry

集成测试

circleci

Travis CI

性能监控

New Relic.

代码质量检查

CODE CLIMATE

用户行为记录

mixpanel

这些好的工具里的很多优秀的产品设计思想，正是我们自己打造自己产品时候的设计灵感的来源，深度使用好工具，能极大的提升整个团队的设计品味，这是除了提高团队工作效率之外的一个隐形的好处。

选择合适的技术

最后一点，也是一个很敏感的话题，虽然我们都知道「PHP 是世界上最好的语言」，但是我们团队还是选择了各种主流技术之外的一个冷门的技术栈，Ruby On Rails。

使用 Rails 的原因是，不仅仅是因为它是一个成熟度很高的框架，而且还因为这个框架背后的「价值观」。

一个互联网工程团队，一般都有两种属性，一种是技能属性，一种是产品属性，我们团队的工程师特别强调工程师的产品属性，而这种团队基因，注定了我们会选择 Ruby On Rails 作为当前的技术框架。我特别喜欢 Rails 的创始人 DHH 写的这篇文章：「The Rails Doctrine」。

Rails Doctrine

http://rubyonrails.org/doctrine/

Ruby 这门语言的出发点是为它的用户的「幸福」考量，而不是从纯粹的计算机科学的视角去发明的，这是 DHH 之所以会选择 Ruby 的初衷。Rails 框架继承了同样的价值观，相比于当前各种纷繁复杂的前后端分离框架，它在技术层面是否优劣，并不是它最重要的考虑因素，Rails 框架重点考虑的是，对于从 0 到 1 的 Web 产品开发，究竟什么样的框架能够让开发者尽快的输出产品功能，让用户去体验和使用，让用户在使用的过程中感到幸福。而 Rails 的天生的好处在于，同样作为一款产品，框架本身是为开发者的「幸福」考量的，这对于一个产品型团队来说，有助于工程师把所有注意力都放在「产品特性」本身上面。

让技术团队保持幸福感，把幸福感通过产品传递给用户，也是维持高效工作的一个基础。

远程工作适合什么样的团队

最后，可能大家会在心里开始默默评估远程工作是不是适合自己了。在我看来，你的团队如果特别小，或者特别久，这两者只要具备一条，那么你现在就可以考虑让团队远程工作了。

所谓特别小，一是团队规模小，二是做的事情小。在个人价值越来越凸显的今天，实际上我们的商业形式也会慢慢的转变，比如个人知识经济，那么可能两三个在一起就能做一些事情出来了，这种非常小的团队是很适合远程的，可以在全国，甚至全球范围内寻找志同道合、配合默契的伙伴。

另一种是核心团队在一起特别久的团队，像我们这样，公司的价值观已经形成，大家彼此呆久了也都足够信任（审美疲劳）了，做的事情也不会是一蹴而就，明天就收获成果的，那么也许可以试试远程。

不过我们也并不认为远程工作就是完美的，远程工作对于一些年轻的小伙伴来说，可能会造成巨大的孤独感，因为人毕竟是感情动物，需要交流，所以我们团队在招募工程师的时候，其实也会尽量在已有工程师的地区招募，这样我们每周五，会让同一个城市的工程师挑选自己喜欢的地方，大家带上电脑出门聚在一起工作。

另外，随着团队规模的扩张，我们也开始每年定期集中，比如 2016 年，整个工程团队一共集中过 3 次，大家在一起的时间总共差不多有 2 - 3 个月的样子，这样可以让新人更好的融入团队，也能集中在一起解决一些产品上重要的问题。

远程工作适合耐心的团队，适合愿意独自面对自己的团队，适合对目标有清醒认识的团队，适合单兵作战强悍的团队，如果你的团队是这样的，不妨一试。

编辑于 2017-03-27

远程工作 Ruby on Rails 彩程设计