为什么未来是全栈工程师的世界？

> 谨以此文献给每一个为成为优秀全栈工程师奋斗的人。

技术在过去的几十年里进步很快，也将在未来的几十年里发展得更快。今天技术的门槛下降得越来越快，原本需要一个团队做出来的Web应用，现在只需要一两个人就可以了。

同时，由于公司组织结构的变迁，以及到变化的适应度，也决定了赋予每个人的职责将会越来越多。尽管我们看到工厂化生产带来的优势，但是我们也看到了**精益思想**带来的变革。正是这种变革让越来越多的专家走向全栈，让组织内部有更好的交流。

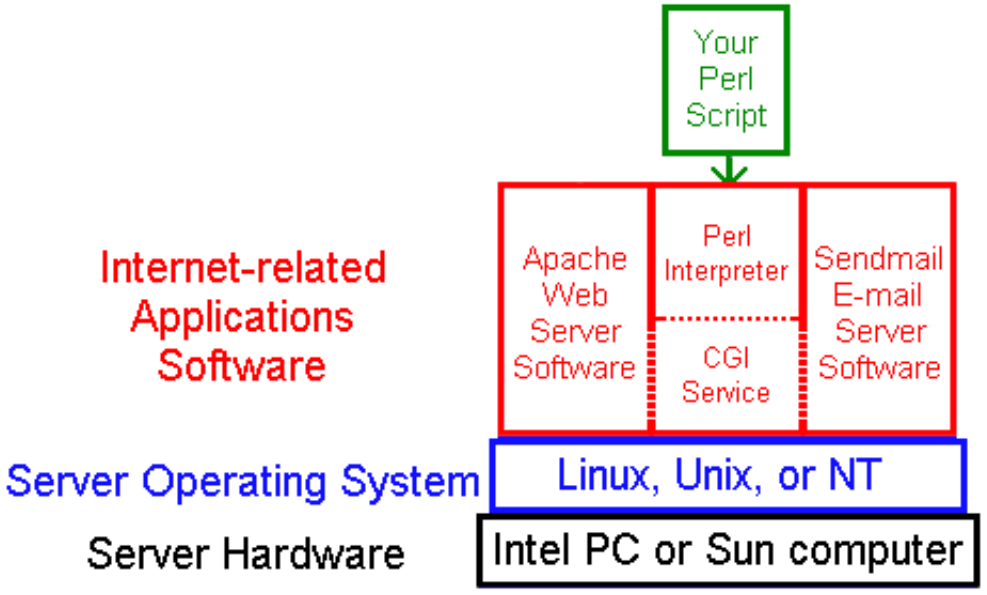
你还将看到专家和全栈的两种不同的学习模式，以及全栈工程师的未来。

技术的革新史

从开始的CGI到MVC模式，再到前后端分离的架构模式，都在不断地降低技术的门槛。而这些门槛的降低，已经足以让一两个人来完成大部分的工作了。

CGI

二十年前的网站以静态的形式出现，这样的网站并不需要太多的人去维护、管理。接着，人们发明了CGI(通用网关接口，英语：Common Gateway Interface)来实现动态的网站。下图是一个早期网站的架构图：



当时这种网站的URL类似于： https://www.phodal.com/cgi-bin/getblog

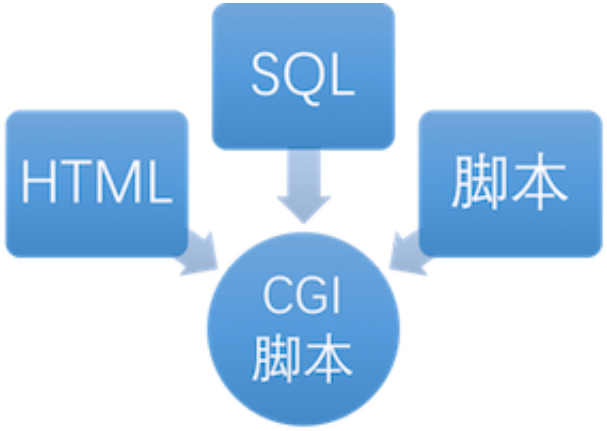
(PS：这个链接是为了讲解而存在的，并没有真实存在。)

用户访问上面的网页的时候就会访问，cgi-bin的路径下对应的getblog脚本。你可以用Shell返回这

个网页：

```
#!/bin/sh
echo Content-type: text/plain
echo hello,world
```

Blabla，各种代码混乱地夹杂在一起。不得不说一句：这样的代码在2012年，我也看了有一些。简单地来说，这个时代的代码结构就是这样的：

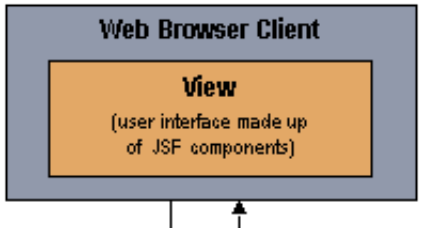


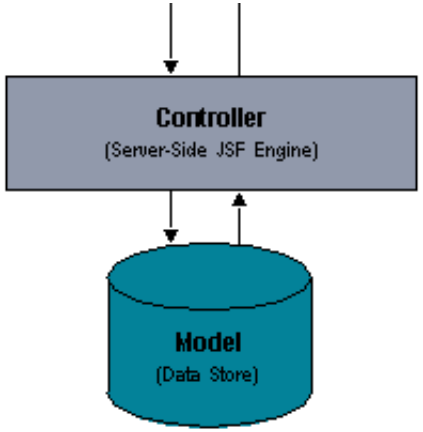
这简直就是一场恶梦。不过，在今天好似那些PHP新手也是这样写代码的。

好了，这时候我们就可以讨论讨论MVC模式了。

MVC架构

我有理由相信Martin Fowler的《企业应用架构模式》在当时一定非常受欢迎。代码从上面的耦合状态变成了：

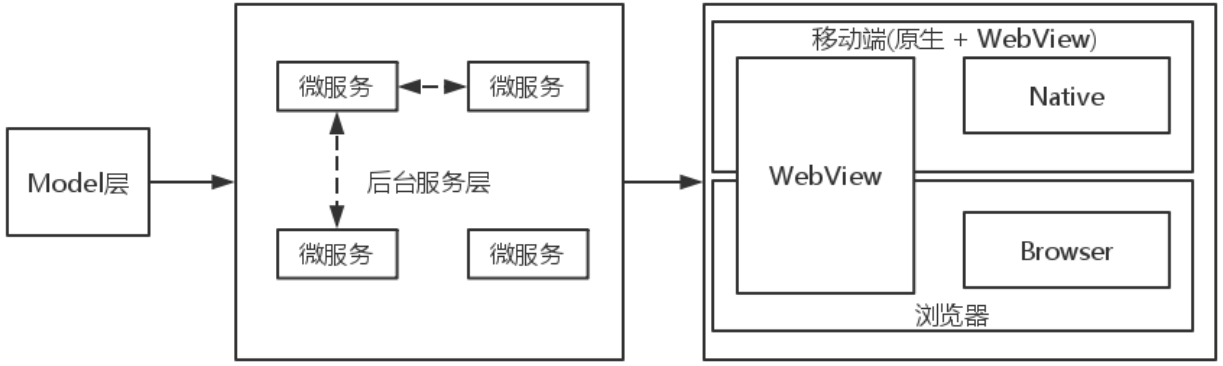




相似大家也已经对这样的架构很熟悉了，我们就不多解释了。如果你还不是非常了解的话，可以看看这本书后面的部分。

后台服务化与前端一致化架构

在今天看来，我们可以看到如下图所示的架构：

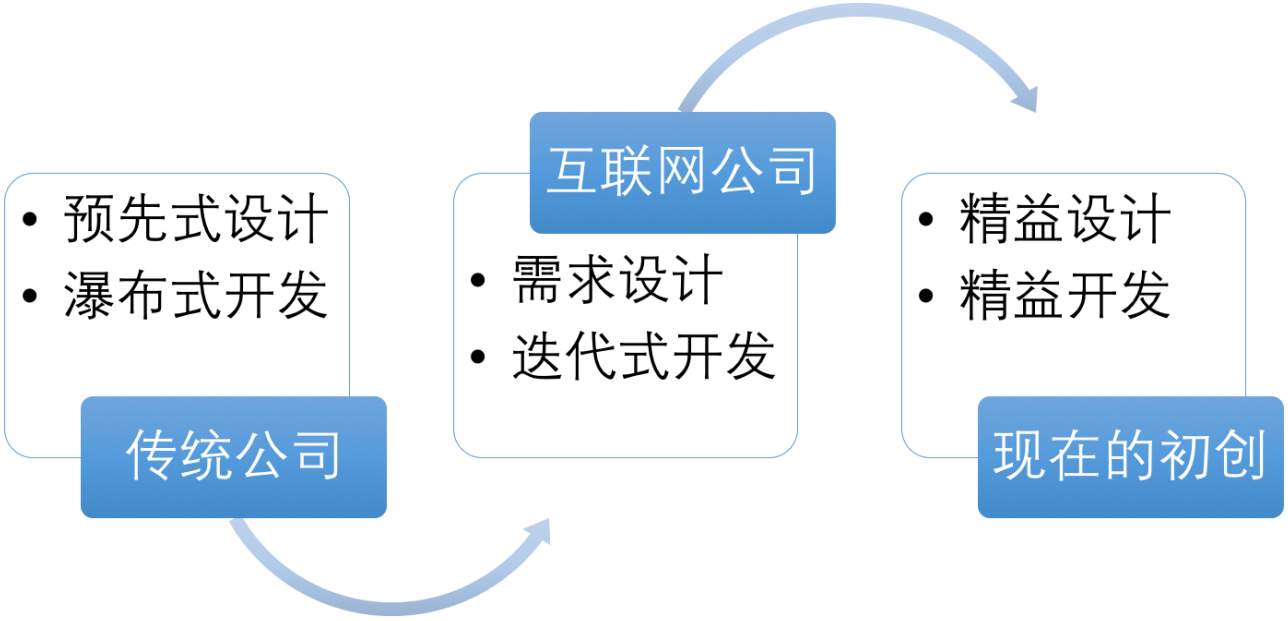


后台在不知不觉中已经被服务化了，即只提供API接口和服务。前端在这时已经尽量地和APP端在结合，使得他们可以保持一致。

软件开发的核心难题：沟通

软件开发在过去的几十年里都是大公司的专利，小公司根本没有足够的能力去做这样的事。在计算机发明后的几十年里，开发软件是大公司才能做得起的。一般的非技术公司无法定制自己的软件系统，

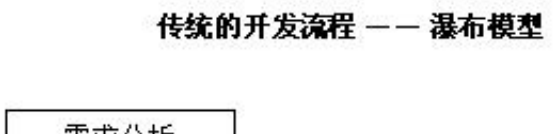
只能去购买现有的软件。而随着技术成本的下降，到了今天一般的小公司也可以雇佣一两个人来做同样的事。这样的演进过程还真是有意思：

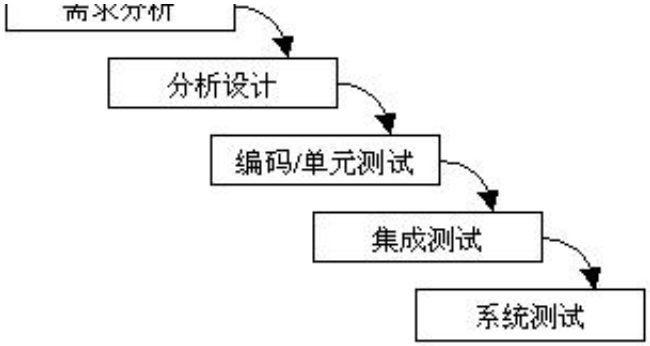


在这其中的每一个过程实质上都是为了解决沟通的问题。从瀑布到敏捷是为了解决组织内沟通的问题，从敏捷到精益不仅仅优化了组织内的沟通问题，还强化了与外部的关系。换句话说，精益结合了一部分的互联网思维。

瀑布式

在最开始的时候，我们预先设计好我们的功能，然后编码，在适当的时候发布我们的软件：



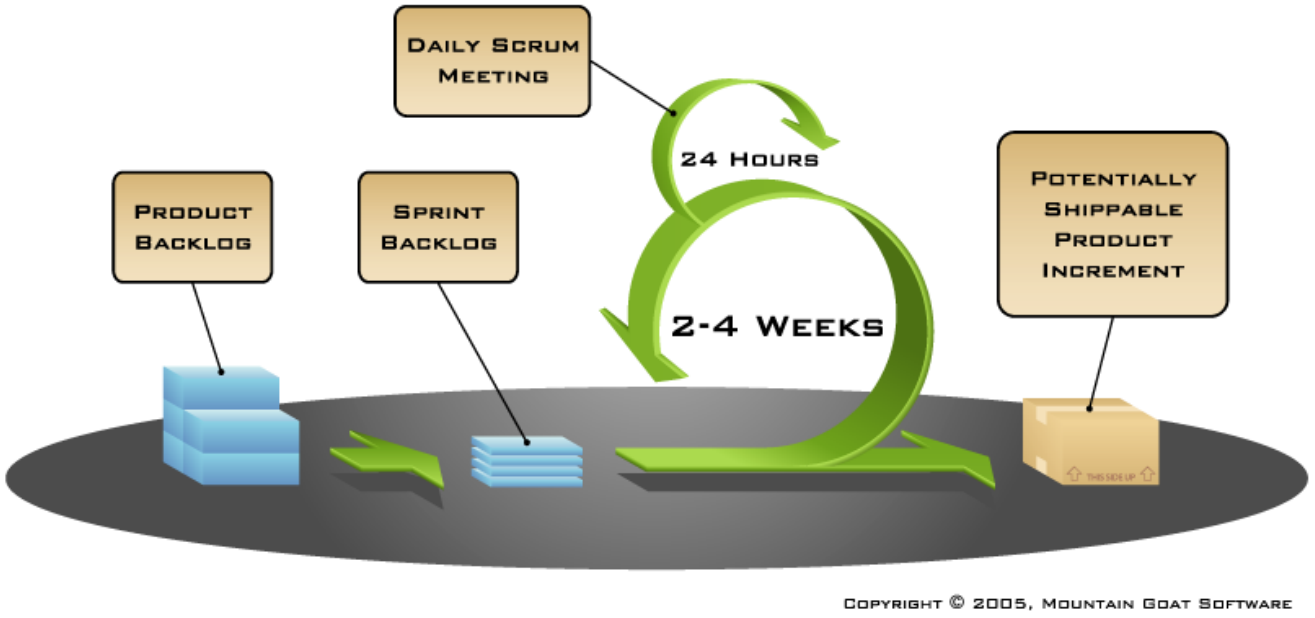


然而这种开发方式很难应对市场的变化——当我们花费了几年的时间开发出了一个软件，而这个软件是几年前人们才需要的。同时，由于软件开发本身的复杂度的限制，复制的系统在后期需要大量的系统集成工作。这样的集成工作可能要花费上大量的时间——几星期、几个月。

当人们意识到这个问题的时候，开始改进工作流程。出现了敏捷软件开发，这可以解释为什么产品经理会经常改需求。如果一个功能本身是没必要出现的话，那么为什么要花功夫去开发。但是如果一个功能在设计的初期就没有好好设计，那么改需求也是必然的。

敏捷式

现有的互联网公司的工作流程和敏捷软件开发在很多部分上是相似的，都有迭代、分析等等的过程：

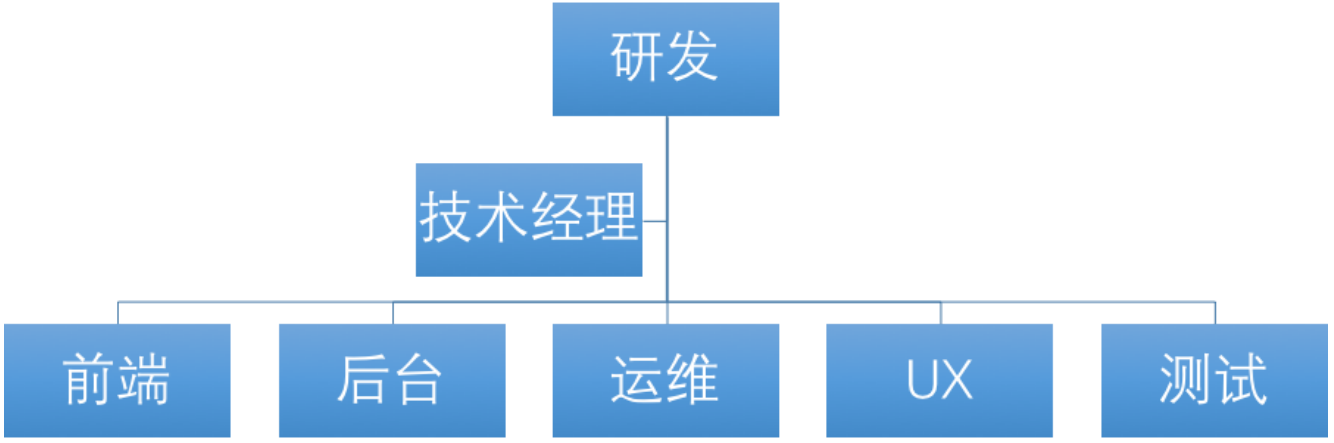


但是据我的所知：国内的多数互联网公司是不写测试的、没有Code Review等等。当然，这也不是一篇关于如何实践敏捷的文章。敏捷与瀑布式开发在很大的区别就是：沟通问题。传统的软件开发在调研完毕后就是分析、开发等等。而敏捷开发则会强调这个过程中的沟通问题：

在整个过程中都不断地强调沟通问题，然而这时还存在一个问题：组织结构本身的问题。这样的组织结构，如下图所示：

如果市场部门/产品经理没有与研发团队坐一起来分析问题，那么问题就多了。当一个需求在实现的过程中遇到问题，到底是哪个部门的问题？

同样的如果我们的研发部门是这样子的结构：



那么在研发、上线的过程中仍然会遇到各种的沟通问题。

现在，让我们回过头来看看大公司的专家与小公司的全栈。

大公司的专家与小公司的全栈

如果你经常看一些关于全栈和专家的技术文章的时候，你就会发现不同的人在强调不同的方向。大公司的文章喜欢强调成为某个领域的专家，小公司喜欢小而美的团队——全栈工程师。

如我们所见的：大公司和小公司都在解决不同类型的问题。大公司要解决性能问题，小公司都活下去需要依赖于近乎全能的人。并且，大公司和小公司都在加班。如果从这种意义上来说，我们可以发现其实大公司是在剥削劳动力。

专家

我们所见到的那些关于技术人员应该成为专家的文章，多数是已经成为某个技术领域里的专家写的文章。并且我们可以发现很有意思的一点是：他们都是**管理者**。管理者出于招聘的动机，因此更需要细分领域的专家来帮助他们解决问题。

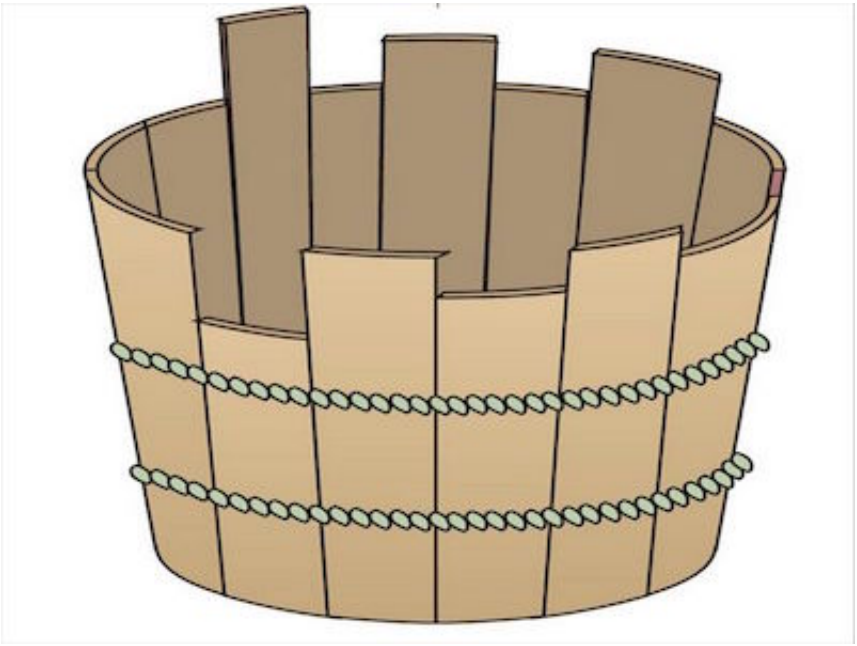
全栈

相似的，我们所看到的那些关于成为全栈工程师的文章，多数是初创公司的CTO写的。而这些初创公司的CTO也多数是全栈工程师，他们需要招聘全栈工程师来帮助他们解决问题。

两种不同的学习模型

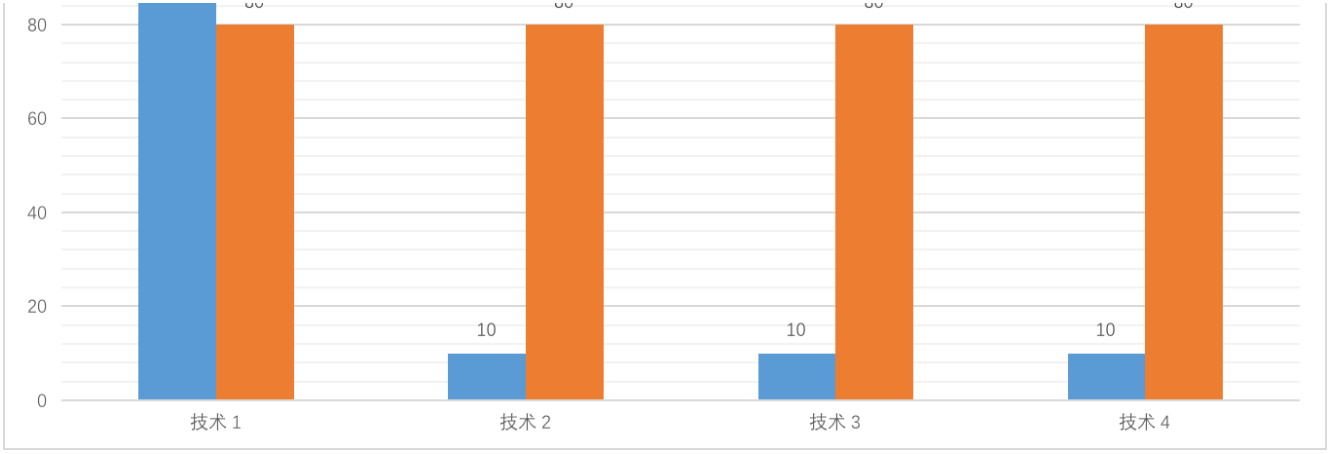
而不知你是否也注意到一点：专家们也在强调**“一专多长”**。因为单纯依靠于一个领域的技术而存在的专家已经很少了，技术专家们不得不依据于公司的需求去开拓不同的领域。毕竟“公司是指全部资本由股东出资构成，以营利为目的而依法设立的一种企业组织形式；”，管理人们假设技术本身是相通的，既然你在技术领域里有相当高的长板，那么进入一个新的技术也不是一件难的事。

作为一个技术人员，我们是这个领域中的某个子领域专家。而作为这样一个专家，我们要扩展向另外一个领域的学习也不是一件很难的事。借鉴于我们先前的学习经验，我们可以很快的掌握这个新子域的知识。如我们所见，我们可以很快地补齐图中的短板：



在近来的探索中发现有一点非常有意思：如果依赖于20/80法则的话，那么成为专家和全栈的学习时间是相当的。在最开始的时候，我们要在我们的全栈工程和专家都在某个技术领域达到80分的水平。

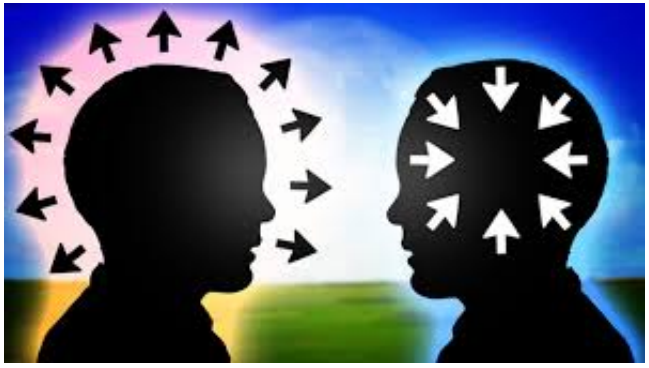
那么专家，还需要80%的时间去深入这个技术领域。而全栈工程师，则可以依赖于这80%的时候去开拓四个新的领域：



尽管理论上是如此，但是专家存在跨领域的学习障碍——套用现有模式。而全栈也存在学习障碍——如何成为专家，但是懂得如何学习新的领域。

解决问题的思路：不同的方式

有意思的是——成为专家还是成为全栈，取决于人的天性，这也是两种不同的性格决定的。成为管理者还是技术人员看上去就像一种简单的划分，而在技术人员里成为专家还是全栈就是另外一种划分。这取决于人们对于一个问题的思考方式：这件事情是借由外部来解决，还是由内部解决。下面这张图刚好可以表达我的想法：



而这种思维依据于不同的事情可能会发生一些差异，但是总体上来说是相似的。当遇到一个需要创轮子的问题时，我们就会看到两种不同的方式。

对于全栈工程师来说，他们喜欢依赖于外部的思维，用于产生颠覆式思维。如Angular.js这样的框架便是例子，前端结合后端开发语言Java的思维而产生。而专家则依赖于内部的条件，创造出不一样的适应式创新。如之前流行的Backbone框架，适应当时的情况而产生。

全栈工程师的未来：无栈

全栈工程师本身不应该仅仅局限于前端和后台的开发，而可以尝试去开拓更广泛的领域——因为全栈本身是依赖于工程师本身的学习能力，正是这种优秀的学习能力可以让他们可以接触更广泛的知识。

全栈的短板

如果你也尝试过面试过全栈工程师，你会怎么去面试他们呢？把你知道的所有的不同领域的问题都拿出来问一遍。是的，这就是那些招聘全栈工程师的公司会问你的问题。

人们以为全栈工程师什么都会，这是一个明显的误区——然而要改变这个误区很难。最后，导致的结果是大家觉得全栈工程师的水平也就那样。换句来说，人们根本不知道什么是全栈工程师。在平时的工作里，你的队伍都知道你在不同领域有丰富的知识。而在那些不了解你的人的印象里，就是猜测你什么都会。

因此，这就会变成一个骂名，也是一个在目前看来很难改变的问题。在这方面只能尽可能地去了解一些通用的问题，并不能去了解所有的问题。在一次被面试全栈工程师的过程中，有一个面试官准备了几个不同语言（Javascript、Java、Python、Ruby)的问题来问我，我只想说Ciao——意大利语：你好！

除了这个问题——人们不了解什么是全栈工程师。还有一个问题，就是刚才我们说的成为专家的老大难问题。

无栈

让我毫不犹豫地选择当全栈工程师有两个原因：

1. 这个世界充满了未解的迷，但是我只想解开我感兴趣的部分。
2. 没有探索，哪来的真爱？你都没有探索过世界，你就说这是你最喜欢的领域

2. 没有探索，哪来的真爱？你都没有探索过世界，你就说这是你最喜欢的领域。

当我第一次看到全栈工程师这个名字的时候，我发现我已然是一个全栈工程师。因为我的学习路线比较独特：

中小学：编程语言 -> 高中：操作系统、内核、游戏编程 -> 大学： 硬件、Web开发 -> 工作：后端 + 前端

而在当时我对SEO非常感兴趣，我发现这分析和Marketing似乎做得还可以。然后便往Growth Hacking发展了：

而这就是全栈学习带来的优势，学过的东西多，学习能力就变强。学习能力往上提的同时，你就更容易进入一个新的领域。

参考书籍

- 《精益企业： 高效能组织如何规模化创新》
- 《企业应用架构模式》
- 《敏捷软件开发》
- 《技术的本质》

更多内容欢迎关注我的微信公众号(搜索Phodal)：