Web全栈工程师的自我修养

余果◎著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

Web全栈工程师的自我修养

余果◎著

人民邮电出版社
北京

图书在版编目（CIP）数据

Web全栈工程师的自我修养 / 余果著. -- 北京 : 人民邮电出版社, 2015.9
ISBN 978-7-115-39902-1

Ⅰ. ①W… Ⅱ. ①余… Ⅲ. ①网页制作工具－程序设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2015)第165233号

◆ 著　　余　果
责任编辑　赵　轩
责任印制　张佳莹　焦志炜
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京方嘉彩色印刷有限责任公司印刷
◆ 开本：720×960　1/16
印张：15
字数：340 千字　　2015 年 9 月第 1 版
印数：1－3 000 册　　2015 年 9 月北京第 1 次印刷

定价：49.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言

您手中的这本书，是我在腾讯五年工作和学习的一些个人心得。

- 我从助理 UI 工程师，一步步晋升为高级 UI 工程师。
- 我从稚嫩的毕业生，变成了领导数十人的团队管理者。
- 我独立设计、制作、发布并开源了一个淘宝客 CMS 系统，并登顶 GitHub 热门关注排行榜首。
- 我作为发起人和主导者，组织数十人一起，翻译了《众妙之门：网站重新设计之道》和《响应式 Web 设计全流程解析》两本书。
- 我从零开始学习 iOS 开发，半年后独立制作并发布了一个 iOS App，赚回了两年的开发者费用。
- 我从一个不敢对陌生人讲话的菜鸟，变成了在几百人面前分享的演讲者。

在这五年中，我最大的收获就是，领悟到做一个“全栈工程师”的快乐。能够做自己喜欢的事情，能够全心投入，能够边学边做，能够不追求完美，能够自我驱动，能够不被自己的头衔局限，能够看到不同技术的联系，能够被老板认可，能够被业界认可，能够相信自己……

由于平时的工作和技术学习都比较跨界，所以在几年前全栈工程师的话题刚刚兴起的时候，我就进行了很多研究和思考。哪些技术对一个组织是真正有用的？全栈工程师的标准能力模型是怎样的？为什么有些人学习和晋升更快？

带着这样的思考，从 2014 年开始，我在豆瓣网发表专栏《谈谈全栈工程师》，发表了 20 篇连载专栏之后，得到了很多读者的欢迎，有五千多人订阅了我的专栏，并且在评论中跟我交流心得、表达感谢。我在开心的同时，也知道自己写得还不够好，文章还有很多语法错误和逻辑不清的地方。于是我打算投入更多心力出一本更好的作品。

经过半年的整理和撰写，这本书终于完成了。我把这本书定义为“轻松的技术杂文集”，希望读者可以以轻松一点的心态来读。书中一小部分内容来

自豆瓣网专栏的扩充，一小部分来自我的博客（http://yuguo.us），一小部分来自这一年多来的梦境和灵感，一大部分想法来自阅读。

本书需要读者有基本的编程基础，能理解基本数据结构，了解一门编程语言的语法。

如果可能，本书尽量不提供某种具体语言的代码实现。此外，读者可能对某一章的内容想作深入的了解，因此我在每一章节的末尾提供了延伸阅读推荐。

关于我

简单说说我自己吧，我从小一直很喜欢读书，高中开始对计算机技术燃起狂热的兴趣。还记得高中时候我每个月必读的两本杂志是《大众软件》和《散文》，即使是最忙碌的高三也没有停止，毕业的时候杂志堆起来一米多高。

《大众软件》话题覆盖面很广，从游戏评测到硬件展览报导，从软件推荐到硬件速递，从手机评测到 CPU 架构介绍……也许从那个时候起，我就养成了对各种新技术来者不拒的习惯吧，这也是我下定决心报考 IT 专业的原因。

小学时候看过很多散文、唐诗宋词，这可能跟我父母都是文科生有关系；可我偏偏热爱并擅长理科，尤其数学和物理，长大后渐渐喜欢看编程类的书。在父母都是文科生的环境下长大，导致我可能有一种感性和理性相结合的特质。

从理性的角度来讲，我做事情非常在意逻辑、证据、数据和对比；从感性的角度讲，我喜欢把我理解的知识用图形化的方式储存在脑海中。还记得高中做数学题的时候，有些关于象限的题，既可以用方程和公式去计算，又可以用图形去推理，我就非常喜欢用图形去推理，看到一个方程式就能“脑补”出解的集合曲线。一个方程组的解就是象限图种几条曲线的交集，因为线是点的集合，所以交点就是既满足方程 A 也满足方程 B 的解，这对我来说是非常容易理解的事情。但副作用是，由于长期不开发自己的记忆能力，所以我很容易忘事，也经常记不住人的名字和脸。理科中，我的生物和化学成绩就不怎么好。

后来我在西安电子科技大学读软件工程专业，西安是一个很美的城市，在沙尘中有一种古老的沧桑感，整个城市也方方正正（处女座最爱），鞋子和衣服在

阳台上放两天就会有一层灰。我很喜欢西安，我在西安度过了美好的四年。

从大学第一天起我就开始写博客，大学生时间比较多，期间折腾了很多域名和很多服务器，以及各种各样的博客程序，也丢过很多内容。在大学毕业那年，我开始启用 `http://yuguo.us/` 这个域名，并抛弃 WordPress，开始用静态站点生成器 Jekyll 生成站点，并使用 GitHub Pages（https://pages.github.com/）提供的免费服务器来托管页面。使用静态页面的最大优点就是访问速度非常快，而且不会出现服务器错误和数据库错误。如果说各种博客程序之间的 PK 就像高手对决，那么 WordPress 这种重型 CMS（Content Management System，内容管理系统）就像降龙十八掌，变化多端，是力量和技巧的极致，有一种无法掩饰的王者霸气。Jekyll 则像是小李飞刀，不会与您正面争锋，但是“小李飞刀，例无虚发”，是速度和精准的极致。

因为非常喜欢折腾网站前端的技术，所以在毕业的时候，我意料之外而又情理之中地选择了前端工程师这一个职业，并且很幸运地在校园招聘中，初次面试腾讯就被录取，并在腾讯工作至今。后来证明，大学的软件工程专业学习很有用。读书时觉得理论知识和后端的知识比较无用，但在工作中却证实，它们非常重要，所以我现在也经常回头复习一些基础知识。

就像乔布斯在斯坦福大学那场著名的演讲里说的，一个人在年少的时候，可能无法看到自己现在做的事情跟自己的未来会有什么关联。您无法预知未来，只能回顾。但是您需要有信心，当您很多年后回头看时，这些点点滴滴会连接在一起，让您朝自己的理想迈进。

我无法预知未来，但回头看过去的五年，我在这期间遇到种种困难，解决各式各样的痛点，帮助项目和团队成长，并成就自己的成长。虽然这些痛点不是每个人都会遇到，世界上也没有完全相同的项目，但是我觉得全栈工程师的理念是通用的，所以我的经验可能对其他人也是有帮助的，这也是我写这本书的初衷。这些思考，我会在本书中一一道来。

最后我想说的是，做您自己感兴趣的事情，学您想学的知识，不要怕走偏了，如果有人说您不务正业，那就让他们说去吧。如果您能远离传统的路子，您将会不同凡响。

目录

什么是全栈工程师

全栈工程师（Full-Stack Engineer），是一个在 IT 行业圈子里越来越热门的话题，无论是像 Facebook 这样的大型公司，还是刚刚起步的初创公司，都开始招募全栈工程师。据说，Facebook 声称：“我们只招全栈工程师！”

Facebook 只招全栈工程师

“全栈”是一个外来词，对于中国读者而言，会觉得它很陌生。当我第一次对某人提到“全栈工程师”时，他一头雾水：“全栈？您是说全端工程师吗？”

其实，“全栈”翻译自英文 full-stack，表示为了完成一个项目，所需要的一系列技术的集合。“栈”是指一系列子模块的集合。这些软件子模块或者组件组合在一起即可实现既定功能，不再需要其他模块。

全栈中的“栈”与计算机数据结构中的“堆栈”不是同一个概念，后者是指先入后出的串行数据结构。顺便说下，“队列”是指先入先出的串行数据结构。

IT 行业之外的人其实很难理解 Web 开发是多么复杂的工程。人们一般认为，在计算机公司或者互联网公司工作的人，就应该能够解决与计算机相关的所有问题：电脑开不了机、应该买什么型号的手机、家里上不了网，等等。在他们眼中，计算机行业的从业者天生就带有“全栈光环”。

但是拿着这本书的您知道，要开发一个 Web 页面，工程师需要掌握的知识至少包括：服务器（比如 Linux）、数据库（比如 MySQL）、服务器端编程语言（比如 PHP）、前端标记语言和脚本语言（HTML、CSS、JavaScript）等。这些技术中的每一个，都需要几年的学习和练习才能达到精通的程度。Web 工程是一个如此大的专业类别，以至于 IT 公司为每一个环节都设置了专门的部门和岗位，来把每一个环节做好。

服务器、数据库、服务器端编程语言、HTML、CSS、JavaScript 等组合在一起就是一个“栈”。这个“栈”是用来制作 Web 站点的，所以又叫 Web 栈（Web-Stack）。[1]

1 最常使用的服务器是基于 Linux 的。Web 发布使用 Apache，数据库使用 MySQL，服务器端编程语言使用 PHP 的组合，所以它们往往一起统称为 LAMP（Linux-Apache-MySQL-PHP）整体解决方案。

如果要开发一个在手机中运行的应用，开发者需要的知识包括：服务器、数据库、服务器端编程语言、iOS 或者 Android 开发技术。这些技术的集合称为 App 栈（App-Stack）。

一个简单的 Web 栈模型：包含前端技术和后端技术。

我们知道，前端工程师就是负责页面浏览器端编程的人，后端工程师就是负责服务器端编程的人，那么什么才是全栈工程师呢？

对于全栈工程师，业界并没有严格的定义，并不是说一定要一种都不能少地具备哪几项知识才能叫做全栈工程师。我倾向于认为，**应该从能力和思维方式两方面，来判定一个人是否是一个合格的全栈工程师。**

国外是怎么定义全栈工程师的呢？在著名的问答网站 Quora 上有人提出了这个问题。一个获得了高票的回答是：

> 全栈工程师是指，一个能处理数据库、服务器、系统工程和客户端的所有工作的工程师。根据项目的不同，客户需要的可能是移动栈、Web 栈，或者原生应用程序栈。

基本上，当客户需要一个全栈工程师的时候，客户需要的是一个**全能的“大神”**。简单来说，全栈工程师就是可以独立完成一个产品的人。当客户让他去做一些舒适区之外的工作时，他敢于迎难而上，并成功完成任务。

我们每一个工程师，进入到公司和企业工作之后，就会有一个职位头衔。我的职位头衔是“UI 工程师”，其他人的头衔可能是“交互设计师”“PHP 开发工程师”，等等。“全栈工程师”不需要头衔。他既有全面的技术能力，也渴望跨界工作的状态。

“全栈”好像是一个遥不可及的梦想，所以对于初次了解“全栈工程师”这个概念的工程师而言，有可能觉得“不可思议”或者抱着“这不可能”的排斥心理。但如果我们回头看看 Web 开发的历史，就知道“全栈”其实没那么难。

Web 开发流程

有人曾开玩笑说，全栈工程师是资本家的阴谋，因为老板想雇一个人来做三个人的工作。

其实在 2000 年第一次互联网泡沫破裂之前，那时候的 Web 工程师也许符合“全栈工程师”的简单定义：一人包揽整个网站的构建。

那时的 Web 工程师们所面临的挑战比今天小很多，他们可能只是制作一些静态的页面，不会面对如今富交互的 Web 应用程序。那时网站可能包含数据库和一些 HTML 表单，但仅此而已，甚至只需要将一些静态页发布到服务器上。在网站的前端无需视觉设计和交互设计，因为网站屈指可数，市场竞争很小，工程师仅用一些基本的 HTML 标签和闪亮的 GIF 图片就可以吸引网民的目光。同时，网站访问量都比较小，前端资源的体积也不大，无需关注服务器压力和 CDN，网民对加载速度的容忍度比较高，也不需要过多考虑用户体验。

但随着技术的发展、用户量的增加、客户端种类变多，每一个小小的细节都需要优化和考虑。在海量的访问量面前，也许改变一个按钮的位置和颜色就能影响上千万的订单。如今的互联网产品已不是以一己之力就可以完成的乐高积木了，Web 开发需要以某种可控的方式来管理。

于是，所有认真对待互联网产品的大公司都引入了流水线开发流程，在这条流水线上诞生了多个非常专业的职位。

大中型互联网公司的产品研发流水线。

产品经理：产品经理其实是对一个产品负根本责任的管理者。他通常的工作包括制订产品规划、协调多方资源、把控产品方向和质量细节，等等。有时候，他会从头策划一个新的产品，而更多的时候，他是在优化已有产品的一个部分。总之，在流水线中，产品经理需要从策划跟进到发布，是一个非常重要的角色。

用户研究员：用户研究员的工作是研究用户行为，有时候他会从宏观的角度分析数据，有时候也从微观的角度分解用户场景，有时候会召集一些用户专门来访谈，或者观察用户对产品的使用情况。从输出品的角度来说，用户研究员一般输出用户研究报告来交付给产品经理和交互设计师，作为产品设计的目标参考。

交互设计师：交互设计师常被简称为“交互”。他与视觉设计师最大的区别是，交互设计师更多着眼于如何优化用户界面的信息分布和操作流程。交互设计师的输出品一般是描述用户与网站“交互”过程的流程图，以及描述页面信息结构的线框图。输出的线框图会交付给视觉设计师。

视觉设计师：在细分交互设计师和视觉设计师的大公司，视觉设计师根据交互设计师输出的线框图来做一些润色和设计，输出最终的产品视觉稿之后将视觉稿交付给前端工程师。在一些不细分交互设计师和视觉设计师的小公司，二者被统称为“设计师”，他们的职责就是负责整个用户界面的设计。

前端工程师：产品视觉稿在得到产品经理和交互设计师等多方确认之后，会交给前端工程师，由前端工程师制作页面，实现视觉稿以及交互功能。从头衔上的变化就可以看出，这时候才真正开始编码。前端工程师需要非常熟悉 HTML、CSS 和 JavaScript，以及性能、语义化、多浏览器兼容、SEO、自动

化工具等广泛的知识。[1]

后台工程师：使用服务器编程语言，进行服务器功能的开发。在编程语言的选择上，很多公司都会出于团队已有成员的知识储备、程序员的供给量或者语言性能方面来进行选择。在这一方面，后台语言的选择是相对自由的一件事，不像前端工程师，为了页面兼容性，必须使用 HTML 和 CSS。如果关注各大公司招聘信息的话，您就会了解，不同公司使用不同的后台语言，比如传统的 C# 和 C++、Java、PHP，或者新潮的 RoR 和 Python。小公司的后台工程师除了负责功能开发，可能还会负责服务器的配置和调试、数据库的配置和管理等工作。在大公司，这些工作会分别委派给后台工程师、运维工程师、数据库管理员（DBA）等岗位。

运维工程师：运维工程师是跟服务器打交道的人，他会关注服务器的性能、压力、成本和安全等信息。

测试工程师：顾名思义，测试工程师保证产品的可用性，即使在小公司，这一职位也是不可或缺的。

流水线的优势

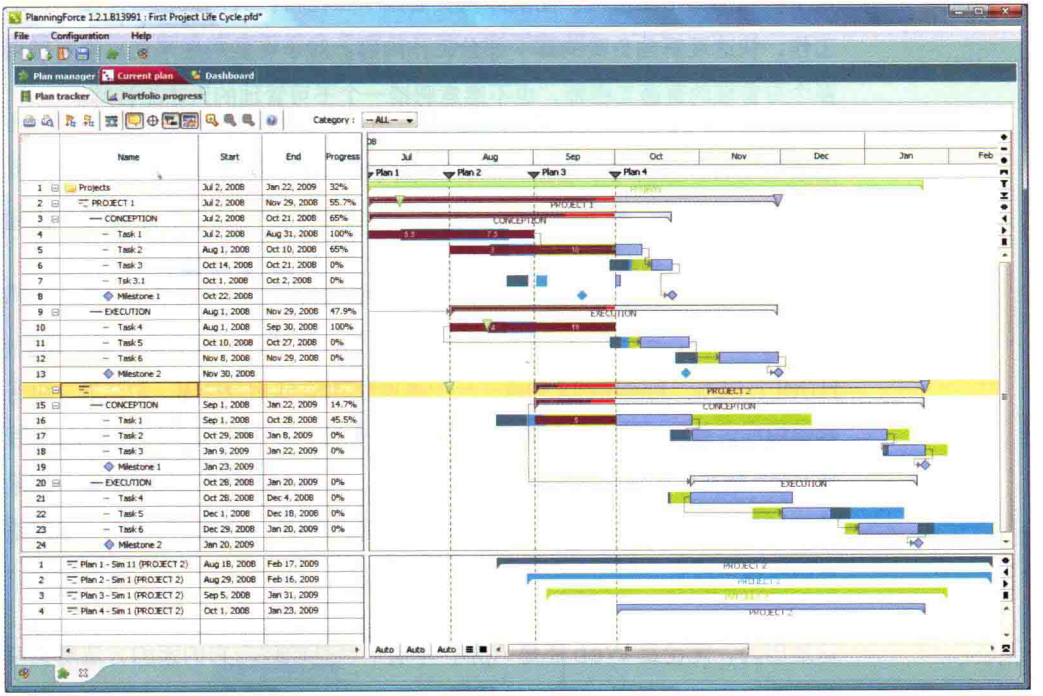
由于有了流水线，其中每个职位的可用工作时间都会作为“资源”来管理，因此需要一位项目经理来把控项目进度，并对人力资源进行调控。比如一个项目立项时，就要预约好这个周期版本需要实现哪些优先级较高的特性，而把优先级不那么高的特性推迟。对于确定在这一周期要实现的特性，就要安排本周进行设计、下周完成开发、下下周进行测试等。

在项目管理中，经常会用到甘特图。甘特图（Gantt Chart）是柱状图的一

1 对于前端工程师这个环节，腾讯公司进行了更进一步的分工，分为“UI 工程师”和“前台工程师”。UI 工程师主要负责 HTML 和 CSS，在制作的过程中要考虑非常精细的设计还原、语义化、页面性能和 SEO 等，而不用考虑 JavaScript 以及页面数据。前台工程师主要负责 JavaScript，他会在静态页面的基础上增加动态数据以及 JavaScript。不是所有的大公司都细分这一职位，在百度和阿里巴巴，前端工程师一个人同时负责页面还原和 JavaScript 开发。

种，显示项目、子项目、进度以及其他与时间相关的系统的进展情况。

流水线在大公司的任何一个严谨的大型项目里都是必不可少的，因为无论是 Web 产品还是 App 产品，它的复杂性都已经超出了单个工程师可以控制的程度。通过把复杂度分解到各个组件，每一个组件就可以进行很好的质量控制。



用于项目管理的甘特图。

Web 页面的生成和传递需要经历复杂的过程，因此容错能力就是首当其冲要考虑的问题。数据从位于深圳某个机房里的服务器传输到用户手机浏览器页面上进行运算和渲染，这个过程中的每个环节都可能出错，所以每一步都要做好容错处理。如果服务器出现错误，是否能在 30 秒内切换到备用机？后台数据异常时返回什么结果给前端，等等。

Web 页面可以在无数种设备上显示。兼容性在此时成为了前端工程师需要考虑的一个重要问题。不同的用户在不同手机上浏览页面，显示的方法会有些许不同，甚至要考虑到如果浏览器不支持 JavaScript，则需要给出特定的提示。

模块化的 Web 开发流程在很大程度上提高了服务的可靠性和可用性，让我们对每一个环节都能单独进行测试。这让大型 Web 开发真正变得可管理、可控制、质量可评估。

流水线带来的另外一个好处是，产品以团队的方式来运作和生产，公司不会过于依赖某一个工程师。团队即使失去某个工程师，其他人也可以接手他的工作，快速理解他负责的那一部分工作内容。对于有些经理来说，宁可雇用多个可管理的普通工程师，也不愿意聘请一个不可管理的天才工程师。

所以到现在，我们可以看到大部分互联网公司都会招聘很多专门的工程师，比如前端工程师、交互设计师，还有一些具体到实现语言的工程师（比如 PHP 程序员），这都是为了提高可靠性、可用性和可管理性。

刚才我们说到，一个基本的 Web 栈由服务器、数据库、服务器端编程语言、HTML、CSS、JavaScript 构成；一个基本的 App 栈由服务器、数据库、服务器端编程语言、手机客户端编程语言等技术构成。您可能已经注意到，App 栈跟 Web 栈在后台技术上几乎是完全相同的，只有在跟用户最接近的那一端采用了不同的技术——要么使用 HTML 制作用户界面，要么使用客户端编程语言制作用户界面。

这是因为，无论是 Web 还是 App，本质上都是软件，它的架构方法是类似的。服务器端接收数据和发送数据，它无需关注客户端采取何种技术制作用户界面。客户端处理用户交互以及显示数据，它不关心服务器使用的是 Java 还是 PHP。如果说开发一款软件就像制造一辆汽车，那么服务器端就像动力系统，客户端就像汽车的车身，不同的动力系统和车身可以自由组合搭配（我不太熟悉汽车的制造过程，这里只是作个比喻）。

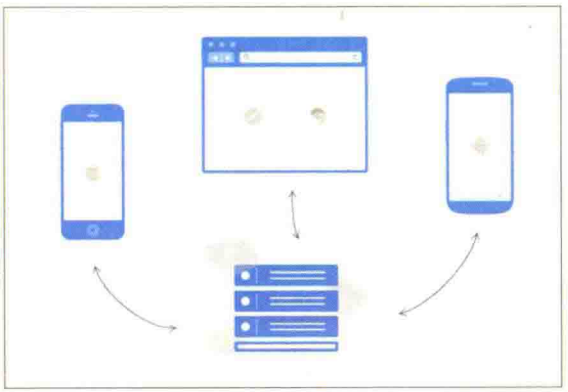
服务器和客户端之间通过 HTTP 协议传递信息。正是因为 HTTP 协议的通用性，使得服务器端和客户端得以实现完全的技术分离。无论是开发 Web 服务还是手机里运行的 App，一套后台开发技术，可以为所有的前端展现方式实现软件的商业逻辑。

HTTP 协议类似于汽车组装过程中的一个通用标准，动力系统和车身都要采用这个统一的标准来实现才可能完美对接。

用户量的大小、服务器承受压力的能力、软件对服务器计算量的要求、对服务器响应速度的要求……诸多因素会影响开发者决定使用哪一种后台技术。汽车的动力性能主要由发动机来决定，汽车厂商也会根据市场需求、消费者定位和制造成本等综合考虑使用哪一种发动机。

而前端技术是根据产品所面向的用户来选择，这要看用户是更喜欢用浏览器还是手机应用来使用服务。就好像汽车的造型要考虑消费者喜欢什么样式的外观。

如果二者功能分离得当，后台服务跟前台服务一般可以自由搭配，互不干涉。



如果服务器逻辑和客户端逻辑分离得当，二者可以自由搭配。

“各司其职”的弊端

虽然流水线式的职业划分和工程管理有很多优点，但是它就像一把双刃剑，在带来高可控性、可用性和可管理性的同时，也给工程师带来了一些困境。

工程师职责不清导致效率低

因为分工太细，所以在不同职业的交接处往往会有一些既不属于上游，也不属于下游的“灰色地带”。

这部分工作没有明确规定由谁去做，所以有时候时间会浪费在沟通上。员工会认为自己的头衔代表了自己的责任边界。比如，一个前端工程师可能会不加思考地实现视觉设计稿，因为他的岗位说明里规定了自己的职责，这其中不包括质疑设计稿，所以他忽视了自己的最终目标：让产品更好。

在一个开放平等的环境中，他实际上可以对影响可用性和性能的设计提出自己的想法。甚至如果他很熟悉这个项目的话，对设计的一致性和一些交互细节都可以说出自己的看法。

- 工程师缺乏主人感导致产品质量差

流水线工作流程对专精工程师的要求是，能很好地执行动作或者执行任务，而不需要对产品的目标有很好的理解。其实在工程师的初级阶段，执行任务的能力是必需的，因为他还没有能力把握产品的目标，而且也需要更多的练习来提升专业能力。但随着经验的积累，如果工程师还不能对产品整体有自己的理解和贡献，就很容易缺乏主人感，要么他会跳槽，要么产品本身缺乏亮点而导致失败。

- 工程师缺乏全局的视野影响个人成长

当工程师希望晋升到更高级的职位，如高级工程师或者管理岗位时，公司对他的大局观会有更高的要求，这就不仅仅是做好“分内”的工作就行的。

高级工程师需要有对设计的理解、对后台知识的了解，以及有跨团队推动项目的能力。长期研究专精的专业知识会让一个人视野变窄，变成“学术派”，而不是“实践派”。

- 更多角色导致项目效率低下

软件工程项目与工业中的标准流程化项目有一个很大的区别：标准流程化项目中每一个流程所接受的输入都是一样的，所需要的输出也都是完全相同的。

比如，一个汽车生产流水线，将“造汽车”这个任务分解成“造轮胎”“造方向盘”等。流程拆分得越细，每一个工人或者机器人就能做得越快，整