划重点讲关于“以终为始”，你要记住的9句话

“以终为始”这个主题模块已经全部更新完毕，相信通过对各种实践的深入讲解，你已经对“以终为始”这个原则有了更为全面和透彻的理解。

为了帮助你更好地回顾和复习，我为每个主题模块增设了“划重点”的加餐内容。现在，我就带你一起梳理一下“以终为始”主题的核心要点。

重点复习

在这个模块中，我们学习到了一些行业最佳实践。

- DoD，确定好完成的定义，减少团队内部的理解不一致。
- 用户故事，细化出有价值的需求。
- 持续集成，通过尽早集成，减少改动量，降低集成的难度。
- 精益创业，减少过度开发不确定性产品带来的浪费。
- 迭代0，在项目开始之前，做好一些基础准备。

还学习到一些重要的思维转变。

- 任何事物都要经过两次创造：一次是在头脑中的创造，也就是智力上的或者第一次创造（Mental/First Creation），然后才是付诸实践，也就是实际的构建或第二次创造（Physical/Second Creation）。
- 在更大的上下文内发现自己的“终”。
- 通过推演，找到通往“终”的路径。
- 用可度量的“数字”定义自己的“终”。

实战指南

在每一篇文章的结尾，我们还将全篇内容浓缩为一句实战指南，希望你可以迅速上手，把“以终为始”的原则运用在实际工作之中，我们一起来回顾一下这些实战指南。

- 遇到事情，倒着想。
  — 以终为始：如何让你的努力不白费?
- 在做任何事之前，先定义完成的标准。
  — DoD的价值：你完成了工作，为什么他们还不满意?
- 在做任何需求或任务之前，先定好验收标准。
  — 接到需求任务，你要先做那件事?
- 尽早提交代码去集成。
  — 持续集成：集成本身就是写代码的一个环节
- 默认所有需求都不做，直到弄清楚为什么要做这件事。
  — 精益创业：产品经理不靠谱，你该怎么办?
- 扩大自己工作的上下文，别把自己局限在一个“程序员”的角色上。
  — 解决了很多问题，为什么你依然在“坑”里?
- 在动手做一件事之前，先推演一番。
  — 为什么说做事之前要先进行推演?
- 问一下自己，我的工作是不是可以用数字衡量。
  — 你的工作可以用数字衡量吗?
- 设计你的迭代0清单，给自己的项目做体检。
  — 启动开发之前，你应该准备什么?

额外收获

在这个部分的最后，针对大家在学习过程中的热门问题，我也进行了回答，希望你懂得：

- 作为程序员，你可以管理你的上级；
- 拿老板说事的产品经理，你可以到老板面前澄清；
- 喜欢无脑抄袭的产品经理，让他回去先想清楚到底抄的是什么；
- 分清楚需求和技术，产品经理和开发团队各自做好各自的事。
  —答疑解惑 | 如何管理你的上司?

留言精选

同学们的留言很踊跃，也很有价值。精彩的留言本身就是对文章内容的补充与丰富，在此我挑出一些优秀的留言与你分享。

在讲高效工作的思考框架时，**张维元** 同学提到：

> 思考框架是道，原则是演化下的术，我们从 A → B，有无穷无尽的路径，最有效的唯有那条直线。本质上，各个维度、原则（不限于作者提到的四项原则）都是帮助我们更好地定位 A 在哪里，B 在哪里，那条直线在哪里。

对于以终为始的原则，**WTF** 同学提到：

> “以终为始”，最常见的一个实践就是计划倒排了。先定时间，然后看功能是不是做不过来得砍掉一些，人力是不是不够需要补充一些，提前预知规避风险。

对于用户故事的验收标准，**liu** 同学提到：

> 程序员的核心职责是如何实现产品功能，怎么实现功能；前提是理解产品功能，需要实现哪些功能。有些项目经理，产品经理与程序员角色混淆。你同他谈功能，他同你谈技术实现，你同他谈技术，他同你谈产品（需要实现哪些功能）。

大家对沙盘推演的话题很感兴趣。其中，**西西弗与卡夫卡** 同学提到：

> 推演可以发现达成目标会涉及到哪些部门、哪些利益相关者，需要哪些资源，以及他们需要何时怎样的配合。

ZackZeng 同学也针对这个话题留言：

> 项目上线之前，一般都会有一个launch plan, 数据库迁移这种项目，不去考虑上线回滚我认为是设计上的缺失。我们公司的launch plan一般是写成一步一步的checklist, 在上线之前会做同伴审查。

Scott 同学也提到：

> 我觉得领导说先跑通再说和事前推演是不矛盾的，很多时候，我们需要一个poc来证明这个项目是可行的，这其实也是事前推演的一部分。上线要事无巨细的检查推演，和快速跑通poc不矛盾，当然现实世界是，大家就急着把poc当正式产品上线了，这是无数个悲剧故事的序章。

休息一下马上回来 同学对推演过程进行了很好地补充：

> 上线前，哪些机器什么配置，应该有一个预期，甚至提前准备好。

adang 同学也分享了他在工作中的感悟：

> 想清楚了才能写清楚，这是我在编程工作非常认可的一句话，并且我也认为它是区分合格与不合格开发工程师的重要区别。软件开发过程中，最常见的例子就是拿到需求后不管三七二十一，上来就开始撸代码，但最后往往返工不断，质量问题层出不穷，而且加班没完没了，这里面一个根本原因就是没有系统地想清楚，但很多人都觉得前期澄清需求、分析设计是浪费时间，只有编码才是真正的创造价值，这就是差距。

在讲到工作要尽量用数字衡量时，**西西弗与卡夫卡** 同学提到：

> 比如开发常常关注的是产品经理提的功能有没有实现，实际上也应该了解做出来的有多少人使用，每个页面有多少人使用。此外，看开发是否努力勤奋，不要光听他说，而是要看看他提交git有多频繁、提交的时间段、代码量有多少。代码质量可以用bug数/代码量来衡量。当然，这些量化未必科学，甚至会被误用，但总胜过凭印象拍脑袋的判断。

大彬 同学也提到：

> 上周我把一个方案进行推迟了，让同事去搜集某项指标的数据，没数据，一切方案都是空谈。AB测试，留言量，阅读量，转发量一切数据都是下一步决策和改进的基础。

篇幅限制，就为大家分享这么多，感谢同学们的精彩留言。留言区还有很多同学提出了各种问题，其实都可以用任务分解的方式去解决。不着急，我们下一个主题的内容就是“任务分解”。

任务分解主题预告

像埃隆.马斯克学习任务分解
——将大问题拆解成能够解决的小问题

测试也是你的事吗？
——开发者测试：程序员工作的一部分

先写测试，就是测试驱动开发吗？
——测试驱动开发：一种设计挑战

大师级程序员的工作秘笈
——任务分解：按部就班工作的前提

一起练习：手把手带你拆任务
——任务分解实战：每一步都要可提交

为什么测试很难写？
——测试的属性：A-TRIP

程序员也可以“砍”需求吗？
——需求的拆分：用户故事

太多人给你安排任务，怎么办？
——优先级管理：做重要的事

如何用最小的代价做产品？
——最小可行产品：找到一条可行的路径

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言

UnivTime
到目前为止读过的最好专栏，文字通俗能看下去，功底深厚，期待后面的文章
2019-01-21 09:24

作者回复
你这么高的评价，给了我继续做好这个专栏的动力，多谢！
2019-01-21 10:02

北天魔狼
老师好贴心，每句总结都是精华。
2019-01-21 06:38

王小勃
打卡
2019-01-21 09:50