读书笔记：高效程序员的45个习惯

gaolhjy

张哥的粉丝推荐的。2022年8月份购买。

近期在工作中认识到自己的一些不足，于是把它翻出来想好好看一下， 并记录一些重点持续更新，与大家共勉。

第二章 态度决定一切

专业的态度应该着眼于项目和团队的积极结果，关注个人和团队的成长，围绕最后的成功开展工作。

做事

1. **最高优先级应该是解决问题**。
2. 一个重大的错误应该被当作是一次学习而不是指责他人的机会。

欲速则不达

1. 不要坠入快速的简单修复之中。
2. 代码审查、单元测试等。
3. 除了深入了解自己负责的部分代码外，还需要从更高的层面来了解大部分代码的功能。

对事不对人

让我们骄傲的应该是解决了问题，而不是比较出谁的主意更好。

1. 分享并融合各种不同的想法和观点，远胜于单个想法为项目带来的价值。
2. 消极扼杀创新。
3. 不要害怕批评。不需要很出色才能起步，但是必须起步才能变得很出色。
4. 能欣赏自己并不接受的想法，表明你的头脑足够有学识。
5. 大家对“最好”的含义要达成共识。

技巧：

    1. 设定最终期限

防止陷入无休止的理论争辩之中。没有最好的答案，只有更合适的方案。

2. 逆向思维

先是积极地看到它的正面，然后再努力地从反面去认识它。目的是要找出优点最多缺点 最少的那个方案，而这个好办法可以尽可能地发现其他缺点。

3. 设立仲裁人

仲裁人的责任就是确保每个人都有发言的机会，并维持会议的正常进行。

4. 支持已经作出的决定

我们的目标是让项目成功满足用户需求。客户并不关心这是谁的注意--他们关心的是，这个软件是否可以工作，并且是否符合他们的期望。结果最重要。

排除万难、奋勇前进

绝妙的计划会因为勇气不足而最终失败。

1. 当发现问题时，不要试图掩盖这些问题，而要有勇气站出来。
2. 做正确的事。要诚实，要有勇气去说出事情。有时，这样做很困难，所以我们要有足够的勇气。
3. 当勇敢地站出来时，如果受到了缺乏背景知识的抉择者的抵制，你需要用他们能够听懂的话语表达。“更清晰的代码”是无法打动生意人的。节约资金、获得更好的投资回报、增加用户利益，会让论点更有说服力。

第三章 学无止境

关键词：跟踪变化，对团队投资，懂得丢弃，把我开发节奏。

跟踪变化

1. 迭代和增量式学习
2. 了解最新行情
3. 参加本地的用户组活动
4. 参加研讨会议
5. 如饥似渴地阅读

对团队投资

一个学习型的团队才是较好的团队。

share, share, share

懂得丢弃

敏捷的根本之一就是拥抱变化。

在学习一门新技术的时候，多问问自己，是否把太多旧的态度和方法用在了新技术上。

丢弃的第一步，就是要意识到还在用过时的方法，另一个难点就是要做到真正地丢弃就习惯。

打破砂锅问到底

不停地问为什么

把我开发节奏

设定一个短时的期限，为任务设定不能延长的最终期限。

第四章 交付用户想要的软件

成功的软件开发-取决于识别和适应变化的能力。

提早集成，频繁集成。

提早实现自动化部署。

使用演示获得频繁反馈。

使用短迭代，增量开发。

让客户做决定

判断哪些是自己决定不了的，应该让企业主做决定。

让设计指导而不是操纵开发

计划是没有价值的，但计划的过程是必不可少的。***避免过度设计***。

合理地使用技术

1. 是否能解决问题?
2. 可取消性?
3. 维护成本?

保持可以发布

简单的工作流：在本地运行测试 -> 检出最新的代码 -> 提交代码

提早集成，频繁集成

代码集成是主要的风险来源。要想规避这个风险，只有提早集成，持续而有规律地进行集成。

提早实现自动化部署

使用演示获得频繁反馈

演示是用来让客户提出反馈的，有助于驾驭项目的方向。

使用短迭代，增量发布

分析、设计、实现、测试和获得反馈，所以叫做迭代。

发布带有最小却可用功能块的产品。每个增量开发中，使用1～4周左右迭代周期。

第五章 敏捷反馈

不同环境，就有不同问题

只要环境不同，就很有可能会有不同的问题

度量真实的进度

判断工作进度最好是看实际花费的时间而不是估计的时间。

6分钟作为一个时间单位，它的颗粒度太细了，这不是敏捷的做法

一周或一个月的时间单位，它的颗粒度太粗了，也不是敏捷的做法

倾听用户的声音

每一个抱怨的背后都隐藏了一个事实。找出真相，修复真正的问题。

第六章 敏捷编码

动态评估取舍

真正的高性能系统，从一开始设计时就在向这个方向努力。

过早的优化是万恶之源。

第七章 敏捷调试

记录解决问题的日志

日志需要记录的一些内容

- 问题发生日期
- 问题简述
- 解决问题详细描述
- 引用文章或网址，以提供更多细节或相关信息
- 任何代码片段、设置或对话框的截屏，只要它们是解决文案的一部分，或者可以帮忙更深入地理解相关细节

维护一个问题及其解决方案的日志。**保留解决方案是修复问题过程的一部分**，以后发生相同或类似问题时，就可以很快找到并使用了。

第八章 敏捷协作

定期安排会面时间

站会的内容

- 昨天有什么收获?
- 今天计划要做哪些工作
- 面临着哪些障碍

1. 只能给与每个参与者很少的时间发言（大约两分钟）。如果要详细讨论某些问题，可以在站会结束之后。
2. 站会的时间最长不超过30分钟，10-15分钟比较理想。
3. 要报告细节。给出具体的进度，但不要陷入细节之中。

实现代码集体所有制

相比找出谁的主意更好、谁的代码很烂，解决问题竟让应用满足用户的期望更为重要。

成为指导者

与团队其他人一起共事是很好的学习机会。

允许大家自己想办法

这么做的好处

- 你在帮忙它们学会如何解决问题
- 除了答案之外，他们可以学到更多东西
- 他们不会再就类似问题反复问你
- 这么做，可以帮助他们在你不能问答问题时自己想办法
- 他们可能想出你可能没有考虑到的解决问题或主意。这是最有趣的——你也可以学到新东西。