高效程序员的45个习惯

- 1 敏捷开发宣言
  - 1.1. 个体和交互胜过过程和工具
  - 1.2. 可工作的软件胜过面面俱到的文档
  - 1.3. 客户协作胜过合同谈判
  - 1.4. 响应变化胜过遵循计划
- 2 敏捷开发概念
  - 敏捷开发就是在一个高度协作的环境中，不断地使用反馈进行自我调整和完善
    - 1 重构，在功能不变的情况下，不断重新设计部分代码，改善代码的质量
    - 2 迭代，频繁发布系统版本让用户确认，确保工作方向正确
- 3 敏捷工具箱
  - 3.1. Wiki
  - 3.2. 版本控制
  - 3.3. 单元测试
  - 3.4. 自动构建、持续集成
- 4 态度决定一切
  - 习惯1：做事，把做事的方向对准问题的解决方法，而不是人
  - 习惯2：不要陷入快速的简单修复之中，要投入时间和精力保持代码的整洁、简洁
  - 习惯3：对事不对人，让我们耐心的应该是解决了问题，而不是比较出谁的主意更好
  - 习惯4：排除万难，奋勇前进，做正确的事情，要诚实，要有勇气去说出实情，有时，这样做很困难，所以我们要有足够的勇气
- 5 学无止境
  - 习惯5：跟踪变化，跟踪技术变化，你不需要精通所有技术，但需要清楚知道行业的动向，从而规划你的项目和职业生涯
    - 迭代和增量式的学习
    - 了解最新行情
    - 参加本地的用户组活动
    - 参加研讨会议
    - 如饥似渴的阅读
  - 习惯6：对团队投资，提供你和团队学习的更好平台，通过午餐会议可以增进每个人的知识和技能，并帮助大家聚集在一起进行沟通交流，鼓励人们对技术和技巧的激情，将会对项目大有裨益
  - 习惯7：懂的差异，学习心得东西，丢弃旧的东西，在学习一门新技术的时候，要丢弃会阻止你取进的旧习惯，毕竟，汽车要比马车车厢强的多
  - 习惯8：打破砂锅问到底，不停地问为什么，不能只满足于别人告诉你的表面现象，要不停地提问直到你明白问题的根源
  - 习惯9：把握开发节奏，解决任务，在事情变得一团糟之前，保持事件之间稳定重复的间隔，更容易解决常见的重复任务
- 6 交付用户想要的软件
  - 习惯10：让客户做决定，开发者、经理或者业务分析师不应该做业务方面的决定，用业务负责人能够理解的语言，向他们详细解释遇到的问题，并让他们做决定
  - 习惯11：让设计指导而不是编纵开发，好设计是一张地图，它也会进化，设计指引你向正确的方向前进，它不是缰绳，它不应该标识具体的路线，你不要被设计（或设计师）操纵
  - 习惯12：根据需求选择技术，首先决定什么是你需要的，接着为这些具体的问题评估使用技术，对付何要使用的技术，多问一些挑剔的问题，并真实地作出回答
  - 习惯13：保持你的项目即时可以发布，保证你的系统随时可以编译、运行、测试并立即部署
  - 习惯14：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，持续所有程序地进行集成
  - 习惯15：提早实现自动化部署，使用部署系统安装你的应用，在不同的机器上用不同的配置文件测试依赖的问题，质量保证人员要像测试应用一样测试部署
  - 习惯16：使用演示获得频繁反馈，清晰可见的开发，在开发的时候，要保持应用可见，每隔一周或者两周，邀请所有的客户，给他们演示最新完成的功能，积极获得他们的反馈
  - 习惯17：使用短迭代，增量发布，发布带有最小却可用功能模块的产品，每个增量开发中，使用1-4周左右迭代周期
  - 习惯18：倒定的价格就意味着背叛承诺，基于真实工作的评估，让团队和客户一起，真正地在当前项目中工作，做具体实际的评估，由客户控制他们要的功能和预算
- 7 敏捷反馈
  - 习惯19：使用自动化的单元测试，好的单元测试能够为你的代码问题提供及时的警报，如果没有别位的单元测试，不要进行任何设计和代码修改
  - 习惯20：先用它再实现它，将 TDD 作为设计工具，它会为你带来更简单更有实效的设计
  - 习惯21：不同环境，就有不同问题，使用持续集成工具，在每一种支持的平台和环境中进行单元测试，要积极地寻找问题，而不是等问题来找你
  - 习惯22：自动验收测试，为核心的业务逻辑创建测试，让你的客户亲自验证这些测试，要让他们像一般的测试一样可以自动运行，
  - 习惯23：度量剩下的工作量，不要用不切当的度量来欺骗自己或者团队，要评估那些需要完成的待办事项
  - 习惯24：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
- 8 敏捷编码
  - 习惯25：代码要清晰地表达意图，编写清晰的而不是讨巧的代码，向代码阅读者暗示你的意图，可读性差的代码一点都不聪明
  - 习惯26：用代码沟通，用注释沟通，使用核心选择的，有意义的命名，用注释描述代码意图和约定，注释不能替代优秀的代码
  - 习惯27：动态评估权衡，考虑性能、便利性、生产力、成本和上市时间，如果性能表现足够了，就将注意力放在其他因素上，不要为了感觉上的性能提升或者设计的优雅，而将设计复杂化
  - 习惯28：增量式编程，在原始的编辑/构建/测试循环中编写代码，这要比花费长时间仅仅编辑写代码的工作好得多，可以创建更加清晰、简单、易于维护的代码
  - 习惯29：保持简单，开发可以工作的、易简单的解决方案，除非有不可预知的原因，否则不要使用模式、原则和高难度技术之类的东西
  - 习惯30：编写内聚的代码，让类的功能尽量集中，让组件尽量小，要避免创建庞大的类或组件，也不要创建无所不包的大杂烩类
  - 习惯31：告知，不要询问，不要抢别的对象或是组件的工作，告诉它做什么，然后打看你自己的职责就好了
  - 习惯32：根据契约进行替换，通过替换代码来扩展系统，通过替换遵循施加已知约束，来添加并改进功能特性，要多使用委托而不是继承
- 9 敏捷调试
  - 习惯33：记录问题解决日志，每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
  - 习惯34：警告就是错误，将警告视为错误，差人导有警告的代码，就跟差人有错误或者没有通过测试的代码一样，都是极差的做法，差人构建工具中的代码不应该产生任何警告信息
  - 习惯35：对问题各个击破，在解决问题时，要将问题域与其周边隔离开，特别是在大型应用中
  - 习惯36：报告所有的异常，处理或是向上传播所有的异常，不要将它们压制不管，就算是临时这样做也不行，在写代码时要设计预会发生的问题
  - 习惯37：提供有用的错误信息，展示有用的错误信息，提供更易于查找错误细节的方式，发生问题时，要展示出尽量多的支持细节，不过别让用户陷入其中
- 10 敏捷协作
  - 习惯38：定期安排会面时间，使用立会，立会可以让团队成员达成共识保证会议短小精悍不跑题
  - 习惯39：架构师必须写代码，优秀的设计从积极的程序员那里开始演化，积极的编程可以带来深入的理解，不要使用不易修改的架构师-不知道系统的具体情况，是无法解开设计的
  - 习惯40：实行代码集体所有制，鼓励同代码的集体所有制，让开发人员轮换完成系统不同领域中不同模块的不同任务
  - 习惯41：成为指导者，分享自己的知识原有最一付出的同时便有收获，还可以激励别人获得更好的成果，而且提升了整个团队的实力
  - 习惯42：允许大家自己想办法，给别人解决问题的机会，指给他们正确的方向，而不是直接提供解决方案，每个人都能从中学到不少东西
  - 习惯43：准备好后再共享代码，绝不要提交尚未完成的代码，放弃差入幽泽未通过或是没有通过单元测试的代码，对项目来说，自编编代码是职守的犯罪行为
  - 习惯44：做代码复查，复查所有的代码，对于提升代码质量和降低缺陷率来说，代码复查是无价之宝，如果以正确的方式进行，复查可以产生非常实用而高效的成果，要让不同的开发人员在每个任务完成后复查代码
  - 习惯45：及时通报进展与问题，发布进展状况，新的想法和目前正在关注的主题，不要等着别人来问项目状态如何