> 罗马非一日建成，软件系统也不是一天能够写出来的，在经年累月的编码生活中，总会有那么些个不经意的瞬间暴露出来，而这些不经意的外在表现日积月累，犹如水滴石穿，会产生巨大的力量反作用于程序员的成长。我简单列了几条，你来看一看，兴许就在身边实实在在发生过。

拿到开发任务后，直接上手写代码。缺少必要的沟通与设计，返工的机率极大。前后端数据的交互格式，功能潜在的关联点不清晰，接口调用方功能是否完备，存储结构的设计，复杂业务的流程设计等等，都需要事先沟通确定好，再动手写代码才能游刃有余，不然会走一步卡一步，进展缓慢，甚至倒退。

在逻辑混乱的地方加入新东西，而不是去重构。由于功能的新增或变更，需要在旧有的代码逻辑中添加新功能，本是一个很好的重构机会，但很多的做法时在原有的基础上填填补补，结果一片混乱。特别是在本已混乱的地方还要加入新代码逻辑，运行起来确实没有问题，但对自身代码质量的保证零意义。

不愿意与别人分享技术成长。教是学习最快的一条路，将自己所学传播分享给他人，并使他人能消化吸收，是对自己知识掌握一个最好的检验。同时在分享的过程中温故而知新，更加深对知识技能的掌握。如果你有教会徒弟饿死师傅的想法，会显得很落伍。互联网时代下，还有什么知识技能，是你独有而别人从来没有的？不如拿出来分享，大家共同成长。一个人走的快，一群人才走的远。

遇到BUG首先否定是自己的问题。这是一个普适性的问题，也是程序员遇到BUG时的第一反应。到底是不是别人的问题呢，往往是问题转了一圈又回到自己手里。耽误了大家的时间，同时降低的解决问题的效率。正解的姿势应该是立即检查自身，确定是不是自己的问题，是就立即改正，若不是，能找到问题所在最好，交由他人去处理，这是一种良好的习惯。

缺乏验证条件时，开发的功能不经测试直接交付给测试人员。一种是过于自信的表现，还有一种是懒惰的表现。有自信是好的，但如果能经过实际的场景来检验，双重保险，对自己对团队都是保证。懒惰就是不负责任的表现，有些功能确实测试起来很复杂，但为了保证功能的可用性，没有条件去创造条件也要完成，这也是一种态度。

修复某BUG时，夹带其它问题。一个未被测试人员发现的BUG，自我发现后私自修复，并提交源码。这在测试阶段比较常见，但后期如果还出现这种问题，对产品/项目的稳定性是个极大的隐患，特别是生产环境。这是一个流程规范问题，也是一个职业素养问题。

先暂时写到这里吧，以上这些都不是大问题，但如果不注意，久而久之会演变成大问题。程序员是个特殊的物种，物种的进化依赖我们自身的知识结构、思维层次，更依赖于日常工作生活的三省其身，经常复盘总结回顾，才能发现问题，近而解决问题。

扩展阅读：

歪脖贰点零 | 认知升级 · 持续学习
程序员，除了编码，生活还应该有沉淀！
长按，识别二维码，加关注