**软件工程师能力自我评价表**

2015年02月25日 13:58:16

阅读数：1075

1.保持高标准，不要受制于破窗理论(broken windows theory)[[i]](http://www.cnblogs.com/xinz/p/3852177.html#_edn1)。  
当你看到不靠谱的设计、糟糕的代码、过时的文档和测试用例的时候，不要想“既然别人的代码已经这样了，我的代码也可以随便一点啦。”

2. 主动解决问题。当看到不靠谱的设计，糟糕的代码的时候，不要想“可能别人会来管这个事情” ，或者“我下个月发一个邮件让大家讨论一下”。要主动地把问题给解决了[[ii]](http://www.cnblogs.com/xinz/p/3852177.html#_edn2)。

3. 经常给自己充电，身体训练是运动员生活的一部分，学习是软件工程师职业的伴侣。每半年就要了解和学习一些新的相关技术。通过定期分享（面对面的分享，写技术博客等）来确保自己真正掌握了新技术。

4. DRY （Don't Repeat Yourself）——别重复。在一个系统中，每一个知识点都应该有一个无异议的、正规的表现形式。

5. 消除不相关模块之间的影响，在设计模块的时候，要让它们目标明确并单一，能独立存在，没有不明确的外部依赖。

6. 通过快速原型来学习，快速原型的目的是学习，它的价值不在于代码，而在于你通过快速原型学到了什么。

7. 设计要接近问题领域，在设计的时候，要接近你目标用户的语言和环境。

8. 估计任务所花费的时间，避免意外。在开始工作的时候，要做出时间和潜在影响的估计，并通告相关人士，避免最后关头意外发生。

9. 图形界面的工具有它的长处，但是不要忘了命令行工具也可以发挥很高的效率，特别是可以用脚本构建各种组合命令的时候。

10. 有很多代码编辑器，请把其中一个用得非常熟练。让编辑器可以实现自己的定制，可以用脚本驱动，用起来得心应手。

11. 理解常用的设计模式，并知道择机而用。设计模式不错，更重要的是知道它的目的是什么，什么时候用，什么时候不用。

12. 代码版本管理工具是你代码的保障，重要的代码一定要有代码版本管理。

13. 在debug的时候，不要惊慌，想想导致问题的原因可能在哪里。一步一步地找到原因。要在实践中运用工具，善于分析日志（log），从中找到bug。同时，在自己的代码里面加 log.

14. 重要的接口要用形式化的“合同”来规定。用文档和断言、自动化测试等工具来保证代码的确按照合同来做事，不多也不少。使用断言 (assertion) 或者其他技术来验证代码中的假设，你认为不可能发生的事情在现实世界中往往会发生。

15. 只在异常的情况下才使用异常 (Exception),  不加判断地过多使用异常，会降低代码的效率和可维护性。记住不要用异常来传递正常的信息。

16. 善始善终。如果某个函数申请了空间或其他资源，这个函数负责释放这些资源。

17. 当你的软件有多种技术结合在一起的时候，要采用松耦合的配置模式，而不是要把所有代码都集成到一起。

18. 把常用模块的功能打造成独立的服务，通过良好的界面 (API) 来调用不同的服务。[[YEKA1]](http://www.cnblogs.com/xinz/p/3852177.html#_msocom_1)

19. 在设计中考虑对并行的支持，这样你的API 设计会比较容易扩展。

20. 在设计中把展现模块 (View) 和实体模块 (Model) 分开，这样你的设计会更有灵活性。

21. 重视算法的效率，在开始写之前就要估计好算法的效率是哪一个数量级上的（big-O）。

22. 在实际的运行场景中测试你的算法，不要停留在数学分析层面。有时候一个小小的实际因素 (是否支持大小写敏感的排序，数据是否支持多语言)会导致算法效率的巨大变化。

23. 经常重构代码，同时注意要解决问题的根源。

24. 在开始设计的时候就要考虑如何测试 ，如果代码出了问题，有log 来辅助debug 么? 尽早测试，经常测试，争取实现自动化测试，争取每一个构建的版本都能有某些自动测试。

25. 代码生成工具可以生成一堆一堆的代码，在正式使用它们之前，要确保你能理解它们，并且必要的时候能debug 这些代码。

26. 和一个实际的用户一起使用软件，获得第一手反馈。

27. 在自动测试的时候，要有意引地入bug，来保证自动测试的确能捕获这些错误。

28. 如果测试没有做完，那么开发也没有做完。

29. 适当地追求代码覆盖率：每一行的代码都覆盖了，但是程序未必正确。要确保程序覆盖了不同的程序状态和各种组合条件。

30. 如果团队成员碰到了一个有普遍意义的bug,  应该建立一个测试用例抓住以后将会出现的类似的bug。

31. 测试：多走一步，多考虑一层。如果程序运行了一星期不退出，如果用户的屏幕分辨率再提高一个档次，这个程序会出什么可能的错误?

32. （带领团队）了解用户的期望值，稍稍超出用户的期望值，让用户有惊喜。

33.  (带领团队) 不要停留在被动地收集需求，要挖掘需求。真正的需求可能被过时的假设、对用户的误解或其他因素所遮挡。

34. （带领团队）把所有的术语和项目相关的名词、缩写等都放在一个地方。

35. （带领团队）不要依赖于某个人的手动操作，而是要把这些操作都做成有相关权限的人士都能运行的脚本。这样就不会出现因为某人休假而项目被卡住的情况。

36. （带领团队）要让重用变得更容易。一个软件团队要创造一种环境，让软件的重用变得更容易。

37. （带领团队）在每一次迭代之后，都要总结经验，让下一次迭代的日程安排更可靠。