**测试之道**

1. **测试流程中的常见缺陷**
2. **对测试与质量理解的偏差。**

* 测试通常被误看作是质量的代名词，如果你问一位开发人员做了哪些与质量相关的事，他的回答往往是“测试”。可是测试并不能保证质量。**质量是内建的，而不是外加的**。因此，保证质量是开发者的任务，这一点毋庸置疑。这就带来了第一个致命的缺陷：测试成了开发的拐杖。测试应该需要一些痛点，需要开发人员费尽心思。某种程度上我们已经把测试变的太轻松，把开发养的太懒了。
* 质量不等于测试！！！**质量不是被测出来的**。如果在最开始设计创建的时候就是错的，那它永远不会变成正确的。虽然质量不是测试出来的，但同样有证据可以表明，未经测试也不可能开发出有质量的软件。
* 小何：质量就像是一个人的体重。首先，质量是内建的，我们一出生体重就是已经存在的一个指标，而一个产品初步开发完成的时候，它的质量也就已经确定了。但是，质量虽然是内建的，但是可以通过测试提高的，就类似我们通过吃饭去增加体重。质量也是需要质量流程去监控和保障的，就类似人类的体重计，胖了就多锻炼，瘦了就多吃点。

1. **测试人员更关注自己的角色，而不是他们的产品。**

如果产品不被关注，那它就好不了。

1. **测试人员往往崇拜测试产物胜过软件本身。**

测试的价值在于测试的动作，而不是测试产物。独立的测试团队，倾向于把重点放在建设和维护测试产物上。如果把测试的目标定位在产品的源码上，整个产品都将受益。因此，测试人员必须把产品放在第一位。

1. **产品经过最严格的测试发布之后，用户有多大可能仍能发现测试中遗漏的问题？答案是：几乎必然发现。**

内部试用者、可信赖的测试者、众包测试者，以及早期用户都可能比测试工程师更容易发现bug。实际上，让测试工程师做的越少，支持其他人做的测试越多，效果就越好。

1. **测试工程师的未来**

（1）**软件交付和用户反馈机制**

通过互联网交付软件，意味着我们有能力选择部分用户进行发布，响应这部分用户的反馈，并迅速进行更新。开发者和最终用户之间沟通合作的障碍不复存在。bug的寿命从几个月变成了几分钟。我们非常快速的进行构建、交付（给内部试用者、可信赖的测试者、早期用户或真实用户）、修改、重新迭代交付，让很多用户根本来不及发现缺陷。这是一种很好的软件交付和用户反馈机制。

（2）**测试工程师会转型测试设计**

少量的测试设计师快速地规划出测试范围、风险热图和应用程序的漫游路线，再安排人进行测试。类似安全工程师这样的专家型角色，或者会变成测试活动的管理者，而那些具体的测试活动则由其他人来完成。

1. **测试为什么要快？**

* 我一直认为，**对于一个坏点子或者考虑欠周的产品，即使再多的测试，也无法把它变成一个成功的产品。**但如果测试方法不当，却会扼杀一个本来有机会成功的产品或公司，至少会拖慢这个产品的速度，让竞争对手有机可乘。——Alberto Savoia（谷歌工程总监）
* **测试不能成为导致创新和开发过程变慢的阻碍。**

1. **角色**
2. **开发工程师**【**深度**】

实现最终用户所使用的功能代码。需要编写与测试代码，包括测试驱动的设计、单元测试、参与构建各种大小规模的测试等。

1. **测试开发工程师【深度+宽度**】

工作重心在可测试性和通用测试基础框架上。

1. **测试工程师【宽度】**

把用户放在第一位来思考。组织整体质量实践，分析解释测试运行结果，驱动测试执行，构建端到端的测试用例。

1. **为什么要阅读、修改设计文档？**

**（1）原因**

* 需要熟悉了解负责项目的系统设计
* 早期提出的建议反馈在文档和代码里面，增加影响力
* 了解所有迭代过程，对项目的整体了解程度超过技术人员
* 在项目初期与相应开发工程师一起建立良好的工作关系

**（2）要点**

* **完整性**

找出文档中残缺不全或需要一些特殊背景知识的地方。通常情况下团队里没人会去了解这些知识，特别是对新人而言。鼓励文档作者在这方面添加更多细节，或增加一些外部文档链接，用以补充这部分背景知识。

* **正确性**

看一下是否有语法、拼写、标点符号等方面的错误。

* **一致性**

确保配图和文字描述一致。

* **设计**

文档中的一些这几要经过深思熟虑。考虑到可用的资源，目标是否可以顺利达成？要使用何种基础的技术框架？期望的设计在框架使用方法上是否正确？设计是否太过复杂？有可能简化吗？太简单了？还需要增加什么内容？

* **接口与协议**

协议、接口的清晰定义。是否完整？是否标准？

* **测试**

系统或文档中描述的整套系统的可测试性怎样？是否需要新增测试钩子（指为了测试而增加一些接口，用以显示系统内部状态信息）？系统的设计是否考虑到易测试性，而为之也做了一些调整？是否可以使用已有的测试框架？预估在测试方面需要做哪些东西，加入文档。

1. **如何编写测试计划？**
2. **ACC（Attribute Component Capability）**

即特质、组件、能力。可以从这三个角度去认识一个产品、分析产品、写测试计划等。

* **A代表特质**

特质是系统的形容词，代表了产品的品质和特色，是区别于竞争对手的关键。在某种程度上，是人们选择你的产品而不是竞争对手的产品的原因。如快速、安全、稳定、优雅等。

* **C代表组件**

组件是系统的名词，在特质被识别之后确定。组件是构成待建系统的模块，是使一个软件之所以如此的关键代码块。

* **C代表能力**

能力是系统的动词，代表着系统在用户指令之下完成的动作。它们是对输入的响应、对查询的应答，以及代表用户完成的活动。能力处于特质和组件的交点。组件执行某种功能来满足产品的一个特质，这个活动的结果是向用户提供某种能力。如“从购物车里增加或删除物品”、“获得信用卡和验证数据”等。能力最重要的一个特点就是：可测性

1. **理解**

测试计划是有一定的基础情况下，对测试的预期。但如果测试计划没有把测试用例应该怎么执行描述的足够详细，它就没有达到预先设定的帮助测试的本意。

1. **如何进行风险分析？**

* 风险发生频率：偶现、必现
* 风险影响范围：用户、公司
* 风险缓解：解决、回归

1. **bug**

于测试人员发现bug，它们的生命是这样的：测试人员发现bug，花些时间细细品味。这一点很重要，不仅仅是因为我们有权利享受自己的劳动成果，而且，这对于理解此bug微妙的细小差别及其出现的条件也是很重要的。它是否在用户可达之路上？这些路径被走到的可能性有多大？除了发现bug这条路径，是否还有更多的路径也会导致相同的问题？是否存在可能影响数据或者其他应用（这将增加其严重性）的副作用？是否存在隐私、安全、性能，或者可访问性方面的影响？当父母听到小孩子的一声轻轻的咳嗽时，常会想象最坏的衰竭性疾病，他们一定非常理解软件工程师对于软件bug的感受。

小何：目前的百度搜索一样，要抱着“小题大做”的心态。