测试专栏特别放送 | 答疑解惑第二期

2018-10-19 茹炳晟

软件测试52讲 进入课程 >

你好，我是茹炳晟。

今天这篇文章是“答疑解惑”系列文章的第二期，我还是选取了五个问题。这五个问题来自于第 6~11 篇这 6 篇文章，其中第 9 和第 10 篇这两篇文章中的两个问题被我合并为了一个问题，并且我会针对这个问题，再次为你简单梳理一条学习路径。

然后，我还会选择这 6 篇文章下的精彩留言，为你稍作解答、分析。

在这篇文章中，我依旧为你添加了每篇文章的链接，并用一句话概括了这篇文章的主要内容，你也可以再次回到这篇文章中，回忆一下第一次阅读后的想法，看看第二次阅读又有了哪些新的感想。欢迎你继续给我留言，我仍在关注着你的问题、反馈，希望可以为你提供更多的帮助。

现在，我们就开始今天的五个问题吧。

在专栏第 6 篇文章《你真的懂测试覆盖率吗？》中，我针对代码覆盖率这个主题，和你分享了代码覆盖率的的价值、局限性，并结合 JaCoCo 分析了代码覆盖率工具的实现原理。这也是这个专栏中第一篇为你讲解某种测试工具实现原理的文章，希望你可以认真体会。

通过阅读这篇文章，我希望你回顾一下曾用过的代码覆盖率工具，并说说你的使用感受和遇到的“坑”。

留言中，我也看到了一些留言很精彩，提到了自己使用代码覆盖率工具的一些实践。所以，也很感谢你们的分享。

接下来，我再和你分享下，在使用代码覆盖率工具时可能面临的两个最大的问题：

1. **测试覆盖率越往后越难。**

统计代码覆盖率的根本目的是，指导用户的测试用例设计。也就是说，我们要通过代码覆盖率的结果去发现哪些代码没有被执行到，以此为依据再去设计有针对性的测试用例。

在这个过程中，你会发现前期达到一个不错的覆盖率指标（比如 70%）还是比较容易的，但是越往后就越难提高了。

因为，后期没有覆盖到的代码往往都是一些出错异常分支的处理，为了能够覆盖这部分内容，往往需要构造特殊的数据和环境。很多时候，这些数据和环境并不好得到，就不得不采用各种 Mock 的手段来实现。

因此，在实际项目中，到底要不要一定到达很高的覆盖率，就应该根据项目情况结合风险驱动的概念来综合分析了。

2. **推行代码覆盖率的初期阶段，很难要求很高的覆盖率指标。**

代码覆盖率的挑战并不是来自于技术本身，而是来自于管理。很多公司在刚刚推行覆盖率统计的时候，会发现各个模块和项目的覆盖率普遍较低，有些甚至还不到 20%。这时，我们就不应该依靠行政手段来强行规定高的代码覆盖率。

许出现下降的趋势，至少保持持平或者逐渐增长的态势。

问题二：你在填写软件缺陷报告时，还有哪些好的实践值得分享呢？

在专栏第 7 篇文章《如何高效填写软件缺陷报告？》中，我分享了想要把发现的缺陷准确无歧义地表达清楚，一份高效的软件缺陷报告应该包括的内容，以及需要注意的问题。

在这篇文章最后，我希望你分享一下自己在填写软件缺陷报告时，还有哪些好的实践。

在这里，我想再和你分享另外一个观点。你在实际提交软件缺陷的时候，有没有感觉这个过程很繁琐。对于缺陷的重现规律和步骤需要做很多尝试，然后写文档时还有很多工作量，比如需要考虑措辞和语句的组织，这往往会花费你不少时间。那有没有什么好方法可以减少写文档的工作量吗？

其实，对于传统软件的开发流程来讲，这种重量级的缺陷报告是必须的。特别是对于一些大公司来说，开发人员和测试人员可能散布在全球不同时区的地域，这种严格的缺陷文档就很有必要了。

但是，现在的很多互联网企业，尤其是国内的互联网企业，所有的工程技术人员都在一个办公室，而且普遍采用敏捷开发的模式，此时面对面地沟通软件缺陷的效果将远远好于文档描述，而缺陷报告本身的作用也会退化成一条简单的记录。

所以，**缺陷报告的详细程度应该在很大程度上取决于团队特征。**

另外，我发现读者在文章的留言中也提出了很多建设性的方法。比如，下面这位昵称为“卫宣安”的读者，提出了让开发人员主动来认领缺陷的方式。如果说所有的开发人员都具有很强的责任心，同时系统规模也不是太大，或者说报告的缺陷数量不是很多的情况下，这的确是解决问题的一个好思路。但是，当团队规模比较大，缺陷数量也比较多的时候，要求每个开发人员都去挨个查看所有缺陷的效率就会很低。

卫宣安

写于 2018/07/13

作为主管研发的产品经理，一直都被 bug 标题太笼统所困扰，每次检索和分配 bug 都会耗费大量精力和时间。而这虽说有制定一些规范，但毕竟每个人的语义表达能力不一，的确很难达到一个比较高的水准。所以我在尝试将 bug 分配方式改成由 bug 对应研发主动认领的方式，去除中心点，但这要求每个研发有足够的责任心，以及相关奖惩制度来把控。

引自：软件测试52讲
07 | 如何高效填写软件缺陷报告?

归属的团队进行自动分类。

在 eBay 的自动化测试体系中，就有专门的系统对失败用例和缺陷做自动进行分类。

问题三：你觉得在实际工程项目中，一份高效的测试计划应该包括哪些内容？

在第 8 篇文章《以终为始，如何才能做好测试计划？》中，我和你分享了虽然在敏捷开发模式下，软件测试不再局限于厚重的、正式的计划文档，但是测试计划的重要性丝毫没有发生变化。一份成功的测试计划，依旧必须要清楚地描述出测试范围、测试策略、测试资源、测试进度和测试风险预估这五个最重要的方面。

而在读完这篇文章之后，我希望你思考的是，在一份测试计划中，除了这五个最最关键的内容外，你觉得在实际工程项目中还需要再增加的内容，以及是不是所有的项目都需要有很详实的测试计划。

其实很多时候， 你会发现计划的速度远远赶不上变化，尤其是互联网产品的开发。就像下面这两位用户在留言中描述的现象，相信你在实际的工作中一定遇到过类似的场景。

大脸猫

写于 2018/07/16

测试计划总是被打乱，而且在前期需求未分析的情况下，测试进度估算的总是不准，很难把控

> 引自：软件测试52讲
> 08 | 以终为始，如何才能做好测试计划？

Knight

写于 2018/07/18

做好测试计划 很多时候被风险给打乱，也就是不确定性打乱。

如 跨部门合作，技术攻关 预研项目不成熟等的引进等的不确定性

同时测试计划受开发进度 质量影响很大，可能之前估计 1 周搞完的结果质量太差 需要搞两周 那最后的发布日期都定下来 很可能就堆人，压缩测试周期了

> 引自：软件测试52讲
> 08 | 以终为始，如何才能做好测试计划?

吗？或者说有没有可能采用轻量级的测试计划。

首先，轻量级的测试计划并不是说没有计划，而是指计划的文档化表现形式应该尽可能简单，只是用一些关键词来描述纲领性的东西。这样做，一来可以降低测试计划本身的写作时间；二来当测试计划由于各种原因发生变化的时候，也可以非常快速灵活地进行修改和更新。

其实，目前的敏捷开发模式（比如 Scrum 模式）下的测试计划就会在每个 Sprint 最开始的时候，以非常轻量级的方式来确定测试计划，有些时候甚至可以没有文档化的测试计划。这时，关于测试范围、测试策略和测试设计之类的内容都在测试人员的脑子中。

注意，这时候虽然没有书面的测试计划，但并不代表说没有测试计划。

问题四：软件测试工程师的高效进阶路径是什么？

我在第 9 篇文章《软件测试工程师的核心竞争力是什么？》和第 10 篇文章《软件测试工程师需要掌握的非测试知识有哪些？》中，分别和你分享了一个软件测试工程师需要具备的核心竞争力，以及需要掌握的非测试专业知识。看到这两篇文章后面，有很多用户留言说：觉得很迷茫，抓不住自己要重点培养的能力、要怎么快速学习。

虽然今年 7 月 6 日我在极客时间做直播时回答过这个问题，但这里为了帮助你快速找到一条适合自己的道路，我就再简单为你梳理下。

首先，我把在软件测试岗位上工作了 0~5（或者 0~3）年的工程师，划分为初级测试人。为什么有这个划分呢？因为 0-5 年工作经验的测试工程师往往工作的中心都还是在软件产品测试本身，还没有将测试上升到质量工程的高度。当然了，我这里说的测试指的是广义上的测试，包括功能测试、自动化测试、性能测试等各个方面。

然后，我把初级测试人可以进阶的方向，归纳为了三类：

1. **业务专家**，也就是业务功能测试方向；
2. **开发测试工程师**，也就是自动化测试方向，指的是把业务功能的测试转换成自动化的脚本；
3. **测试开发工程师**，指的是负责开发测试平台、工具，以及服务。

首先，如果你想成长为一名业务专家的话，那你就需要精通于某一项具体的测试业务，比如说电子商务网站、EPR 系统和 SAP 系统等等。这样的角色更像是产品经理，你需要能准确把握业务产品的定位，了解整个业务流程的操作、用户的使用习惯，以及如何提到整个产品的转化率。

而要成为这样的人，你首先需要在该领域中有较长时间的积累，能够从真正的终端用户视角来使用被测软件，能够对该软件应用领域的行业知识有比较清楚的了解。

但精通于某一个业务领域的缺点是，一旦离开了这个业务领域，之前的业务积累就没用了。

其次，如果想要成长为一名开发测试工程师的话，你平时需要积累高效的测试用例组织方法、对自己用到的测试框架的优劣势有深入理解，并在使用某种测试工具时要深入到其原理的层面。

这样的话，你就可以快速具备测试用例设计、测试框架选型、灵活运用测试工具的技能。并且，你也会因为这种长期的“知其所以然”的积累，可以从容应对新的技术趋势。比如说，你掌握了 Selenium 1.0、2.0 的实现原理后，对 API 测试的原理也就可以做到触类旁通了。

最后，如果想要成长为一名测试开发工程师的话，那你需要培养的是开发能力，以及测试意识。说白了，测试开发工程师，更像是一个开发人员，只不过需要在理解测试上下文的基础上，为其他测试工程师开发一些平台、工具、服务。这时，我建议的成长路径，就是一个开发工程师的成长路径了。

问题五：你觉得你现在团队采取的自动化测试策略，有哪些好的地方，又有哪些需要改进的？

在第 11 篇文章《互联网产品的测试策略应该如何设计？》中，我和你分享了互联网产品的特性决定了它采用的测试策略，遵循的是“重量级 API 测试，轻量级 GUI 测试，轻量级单元测试”的原则，更像是一个菱形结构。这与传统软件产品的金字塔测试策略有所区别。

而在文章最后，我希望你可以针对你所在公司采取的测试策略，谈谈自己的看法。

eBay 在早期阶段，也和现今大多数公司一样，采用的是基于 GUI 来大规模开展自动化测试。尤其早期阶段的网站本身还不是太复杂，而且还不是现在的微服务架构，所以基于 GUI 的自动化测试在页面对象模型和业务流程模型的支持下，能够很好地完成业务功能的验证和测试。

这个阶段，我们关注的重点是通过 GUI 层面的操作来对整个网站的业务功能进行验证。

可是，后来随着业务的不断发展与壮大，基于 GUI 的测试用例数量不断增长，同时再加上浏览器兼容性测试等的要求，测试用例的数量越来越多，测试执行的效率也越来越低下，所以单靠 GUI 测试已经很难满足全面测试的要求了。

再加上系统架构本身也从原本的单体应用逐渐发展成为了微服务，甚至是服务网格的架构形式，同时普遍采用了前后端分离的架构设计，所以此时的测试重点也就从原来以 GUI 为主转变成了以后端 API 为主的阶段。

此时 GUI 测试只会去覆盖一些最最基本的业务功能，而 API 测试则会关注各种参数的组合以及各种边界场景。

通过这个实际的例子，我们可以看出，**并不存在一个放之四海而皆准的自动化测试策略。测试策略的选择，在很大程度上取决你的测试诉求以及被测系统本身的架构设计。**

最后，感谢你能认真阅读第 6 到 11 这六篇文章的内容，并写下了你的想法和问题。期待你能继续关注我的专栏，继续留下你对文章内容的思考，我也在一直关注着你的留言、你的学习情况。

感谢你的支持，我们下一期答疑文章再见！

上一篇　测试专栏特别放送 | 答疑解惑第一期

下一篇　48 | 优秀的测试工程师为什么要懂大型网站的架构设计?

精选留言 (7)

写留言

顾范燕
2018-10-25
1

老师你好！我在mac自动化上有个问题，导入文件的时候要用mac自带的文件选择器，这个要怎么实现呢！网上都是关于windows的解决方案呢！

笨熊
2018-10-20
1

十一之后开始阅读，终于追上了。

展开

展开

口水窝
2019-06-03

感谢茹老师在写完课程后还能根据所提问题按照测试系统给我们总结一下！

展开

90后糊涂宝...
2018-10-19

1、如何提高需求覆盖率
2、敏捷开发模式下，测试人员即能保证质量又能提高效率
3、测试能全覆盖吗?

展开

Robert小七
2018-10-19

很好的总结

展开

涅槃Ls
2018-10-19

打开49，很好的总结 温故而知新2

展开