28 | 解决问题：如何保证自动化测试的可信性？

2022-05-23 柳胜

《自动化测试高手课》 课程介绍 >

讲述：柳胜

时长 14:46 大小 13.53M

你好，我是柳胜。

咱们今天讨论一个十分影响自动化测试 ROI 的具体问题，就是自动化测试的可信性。你可能听过健壮性，稳定性，但什么是自动化测试可信性呢？

我说一个场景你就明白了，当你的 UI automation 测试报告显示 Login 案例运行失败了，查了 Log 之后，发现是测试机上出了一个弹窗，转移了浏览器的焦点导致失败。这种情况下，就是自动化测试的失败 != 产品的 Bug，你白忙活了一通。

同样的，另外一个场景，你的 API automation 测试报告一直显示成功，直到有一天，生产环境发生了 Bug，你检查 API test 的执行日志，才发现 API 的 Response 早就出了问

通过这两个例子，你能够看出来，可信性指的是自动化测试报告是否直接反映产品真实的质量状态。如果它经常误导报告的使用者，那么自动化测试快速可靠的价值就会大打折扣。你也没有信心把自动化测试当做一个服务，提供给开发人员和运维人员去使用。

如果你的自动化测试项目有可信性不足的问题，那怎么办呢？学完今天的内容，你就知道怎么用度量驱动来解决这个问题了。

可信性

我们首先分析一下自动化测试报告失信是怎么发生的。我们把测试结果的成功和失败，跟产品功能的正确和错误，做一个组合矩阵。

测试结果		功能表现	
		正确	错误
运行结果	成功	真阴 true negative	Bug泄漏 false negative
	失败	测试噪声 false positive	真阳 true positive

极客时间

结合矩阵我们可以看到四种情形：

1. 产品功能出错的时候，自动化测试会检测出来也报错，这叫做 True Positive，真阳。

2. 产品功能出错的时候，但是自动化测试运行结果是成功，这叫做 False Negative，假阴。

3. 产品功能正确的时候，自动化测试也运行成功，这叫 True Negative，真阴。

4. 产品功能正确的时候，但自动化测试运行结果是失败，这叫 False Positive，假阳。

其中假阳的后果是浪费了自动化测试人员的维护时间，所以我管它叫做**测试噪声**。而假阴的后果是导致 Bug 没能被检测到，泄漏到了生产环境中，我管它叫做 **Bug 泄漏**。

我们期望能够避免假阴和假阳，那应该怎么做呢？ 数据驱动的解决思路很简单，两步走：首先我们要能计算出来假阴和假阳的数量，然后找到降级它们的办法。

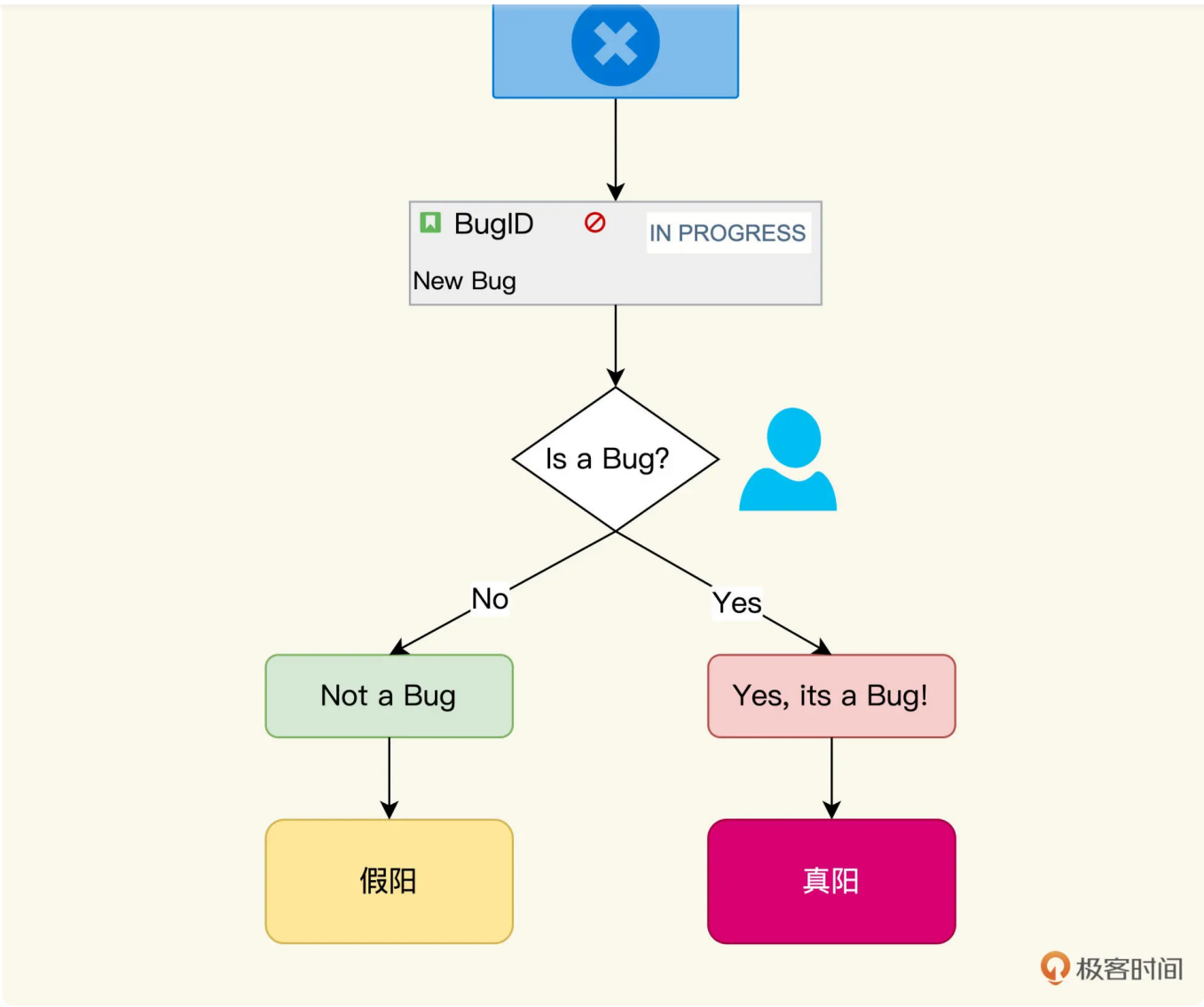
度量假阴和假阳

怎么度量假阴和假阳？ 有几种思路来实现。

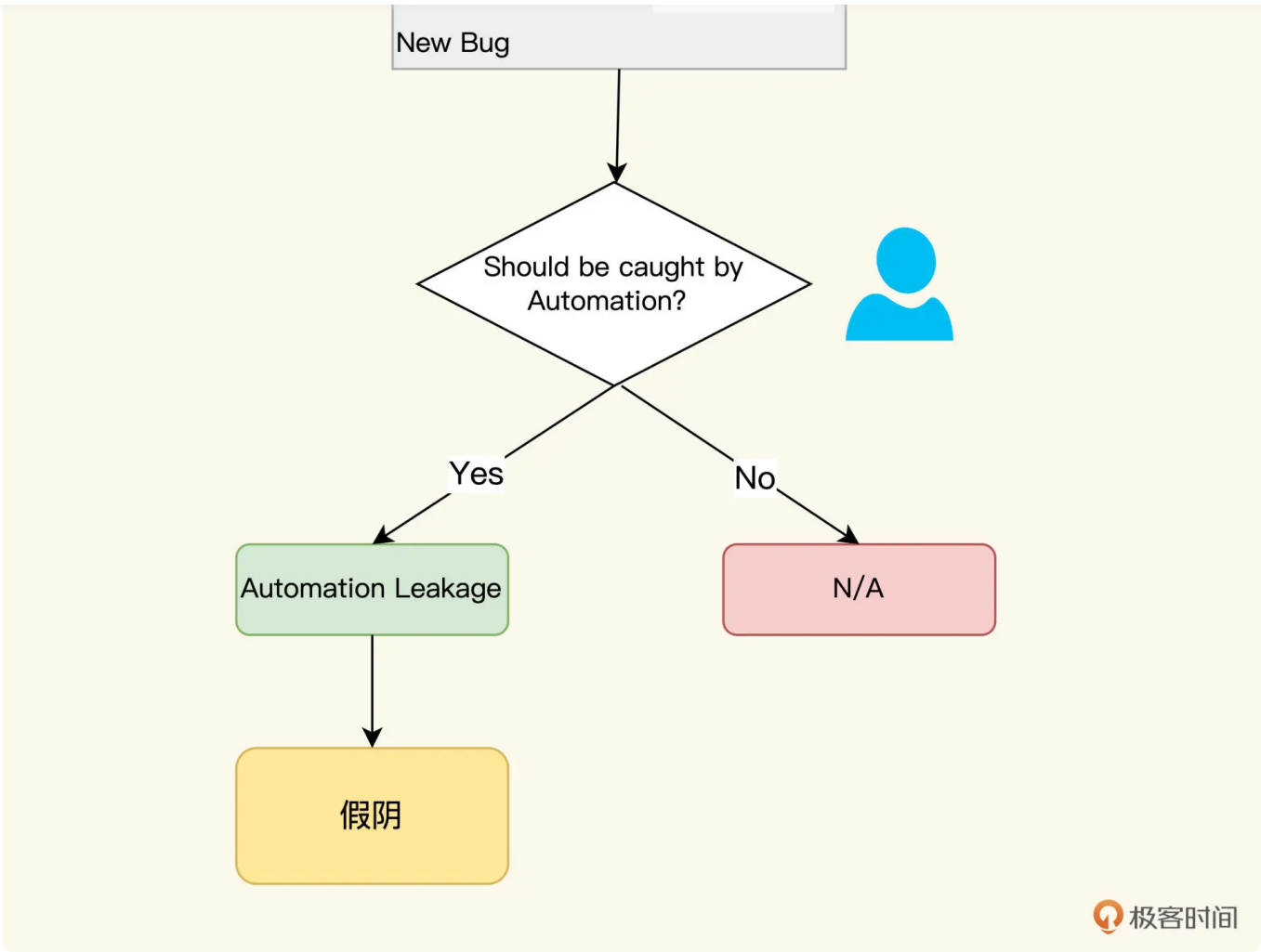
基于 Bug 的方式

第一种，基于 Bug 的方式。

每一个自动化测试的失败会触发提交一个 Bug，然后交给自动化测试人员诊断。如果是真的 Bug，就移交给开发人员；如果是假阳的话，需要给 Bug 标记一个“Not a Bug”，最后我们统计“Not a Bug”的 Bug 数量作为假阳的数量。



同样，假阴也可以用 Bug 来度量。我们通过 Bug 溯源，就可以知道 Bug 是在哪个阶段泄漏的，如果是自动化测试泄漏的话，那做一个标记“Automation Leakage”，最后再统计所有带有“Automation Leakage”标记的 Bug 作为假阴的数量。这样，假阳和假阴的数量我们就都有了。



这种基于 Bug 的方式，从理论上看起来讲得通，但我们要是实践起来，就会发现其中的困难。有两个因素会导致你的统计不全面：第一是需要增加人工的工作量，第二是人的主观判断会影响到度量结果。

这两个因素会导致度量的收集不全面，不准确。比如，Bug 不做标记，就不会被统计到；Bug 标记错了，就会不准确，标记错误是很常见的，有的人会觉得 Bug 是 UI 自动化测试的泄漏，有的人会认为是 API 自动化测试的泄漏，不同人去做这个归类，都会得出不同的结果。

一旦推行设计了这样的度量方式，改进目标很可能最终落空，因为数据可能被扭曲，甚至发生数据造假的情况。实际上，上面说的基于 Bug 的方式，违背了我们讲过的一个度量原则——**度量数据的来源应该是来自未经人加工的数据。**

所以，我们再接着寻找更合适的度量办法。

的时候，代码能自动地去识别这是一个产品的错误，还是自动化测试代码的问题。

方向找到了，使用关键问题法（可以回顾第二十七讲），我们可以提出这样的问题：“产品的错误和测试代码问题有什么区别？”。

下面我们通过一个自动化测试案例的代码示例，也就是 Restassure 工具开发来分析一下。

复制代码

```
@Test
public void searchMovieById() {
    //加载测试数据，电影为哈利波特
    TestData testMovie = TestData.loadJson("movie_harryPotter.json");
    //查询Movie API，验证Movie ID为123456
    get(uri + "/movie/123456").then()
      .assertThat()
      .statusCode(HttpStatus.OK.value())
      .body("id", equalTo(123456))
      .body("name", equalTo(testMovie.getName()))
      .body("Type",testMovie.getType());
}
```

这段 API 测试代码，是测试 GET "/movie/123456"得到的数据。对于这个结果数据，一共有四个验证点，HTTP 状态代码是 200，movie 的 id、name 和 type。为了执行测试，还加载了一个测试数据源 movie_harryPotter.json。

searchMovieById 函数一共有 7 行代码，每一行代码都有可能引起 searchMovieById 的运行失败。测试数据加载找不到文件，movie API 返回的数据不是哈利波特的电影，或者返回的数据里没有 Name 字段，这些都可能是 searchMovieById 失败的原因。

我们来分析一下这些失败情况，哪些是假阳，哪些是真阳呢？

测试数据的加载失败是假阳，因为它是自动化测试代码本身的问题，需要自动化测试开人员来解决；而返回数据不正确则是真阳，它代表着一个产品的错误，需要开发人员，环境人员和业务人员来共同处理。

试构建的，它们必须保证健壮和稳定。

到这里，我们就梳理出一条规则，可以区分出产品错误和代码健壮性问题：**Assert 检查点的失败，是产品错误，其它的失败都属于自动化测试代码健壮性的问题。**

基于这条规则，真阳率和假阳率的度量公式我们就找到了，即：

真阳率 = Assert 语句引起的失败次数 / 自动化测试失败的总数

除了真阳，就是假阳，那么假阳率 = 1- 真阳率。

基于 Assert 语句的度量方法解决了 Bug 方式的人工污染问题。因为 Assert 语句是在开发代码写入的，抛出来的 Assert Exception 也可以被框架识别，它的数据采集是可以完全自动化的，不需要人工参与。但 Assert 度量方式需要基于一个规则，那就是测试人员把验证逻辑都用 Assert 语句来表达。

所以，如果我们以 Assert 语句的假阳率为度量指标，那么它就会起到一个效果：推动自动化测试开发人员做更多的 Assert 检查点。这不正是我们期望的么？ 通过度量驱动，把团队带入到一个积极的正向循环里。

好，假阳度量解决了，我们再来看看假阴的度量。

假阴的场景是：自动化测试运行成功了，但实际没有捕捉到它应该捕捉到的 Bug。这种 Bug 泄漏情况，看起来只能是 Bug 发生了之后，才能度量假阴率。如果没有 Bug 发生，或者即使发生了，没有人去做标记归因，那么假阴率就始终为 0。所以通过基于 Bug 的方式来衡量假阴，也是不靠谱的。

假阴该怎么度量呢？参照上面的假阳度量方式，我们也可以从代码层级上来度量假阴。

没有 Bug 怎么办？那就创造 Bug 来度量。听到这句话，你会不会感觉很熟悉呢？没错，在第二十六讲提到的变异测试里的每一个变异，就是一个制造出来的 Bug。变异测试覆盖率的反向指标就是假阴率，变异测试覆盖率越高，假阴率就越低。

么 Assert 语句越多，产品的验证越完备，Bug 的泄漏可能性也会降低。

基于这个逻辑，我们可以把验证点的数量 / 测试案例作为一个度量，就是每个测试案例中出现的验证点数量，来驱动自动化测试开发人员养成在测试案例多做检查点的习惯，最终达到减少假阴的目标。

建立度量是度量驱动循环中最重要的一环，合理有效的度量指标，就是自动化测试的前行路上的灯塔。有了灯塔指引方向，大船航行就不至于迷失偏航。接下来，我们就来看看“航行路线”，也就是完整度量过程的生命周期。

建立完整度量驱动周期

一个完整的度量周期可以分解成四个阶段：

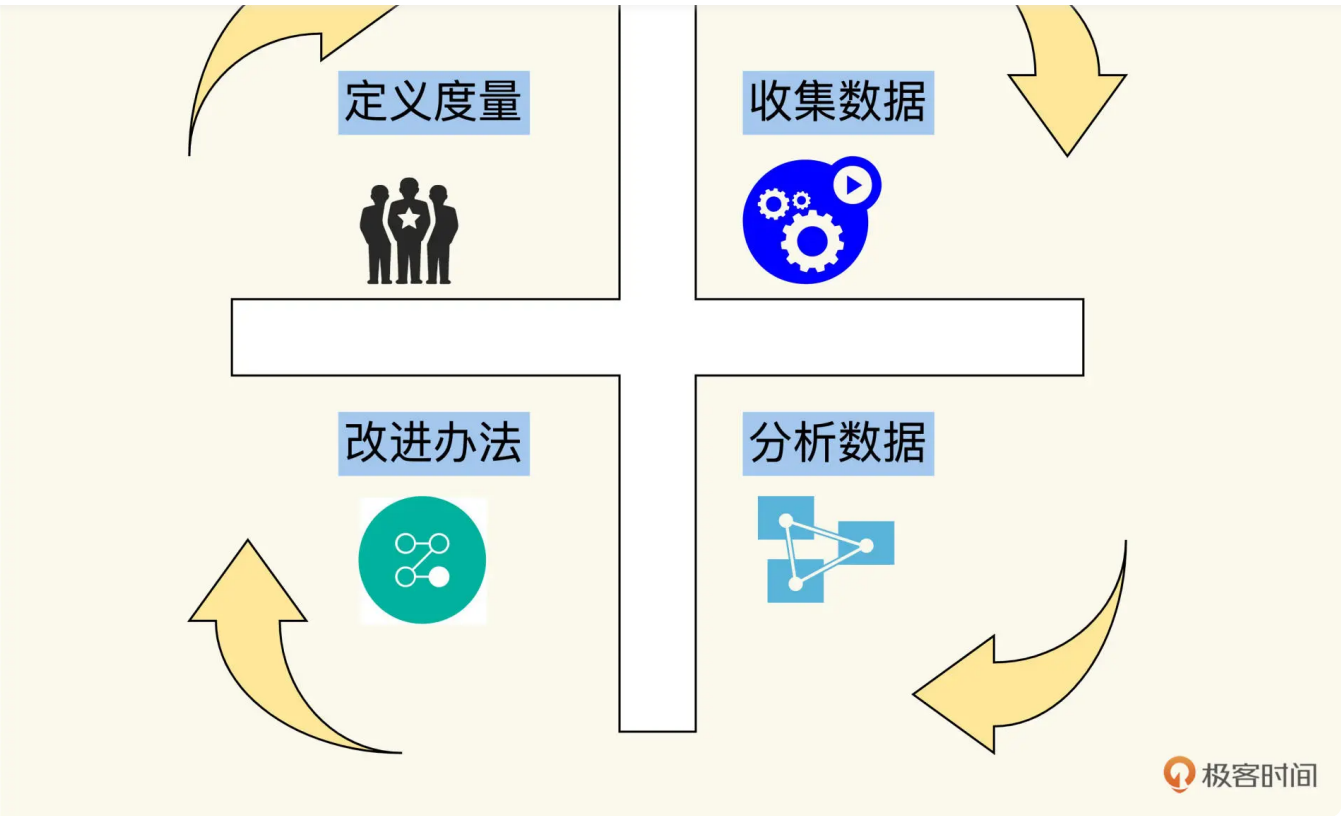
1. 度量定义：建立度量指标，设置目标

2. 数据采集：收集原始数据，聚合成度量指标

3. 问题分析：查看度量数据，并找出问题

4. 优化改进：提出和应用解决办法

经过这四个阶段，一个循环周期就宣告完成。然后可以重新开始第一阶段“度量定义”，进入第二个循环周期，持续改进，直到度量达到目标。这时候，就可以考虑换一个新的度量指标来提升了。如下图：



就拿前面我们说的假阳度量为例。在第一个循环周期，我们先采集出来假阳率是多少，比如是 40%。

接着我们从第二个循环周期开始，设置假阳率的目标是控制在 15%，推动团队想尽办法，通过技术或流程，来降低假阳率。然后是第三个周期，如果得到假阳率是 30%，那说明办法奏效了，继续努力，直到最终假阳目标达到 15%。

为了保持成果，你可以把假阳率保留在观测列表上，这样，一旦有什么变动影响到了假阳率，我们可以快速得知，作出反应。

其中，第一步的度量设计和第四步的改进办法，需要测试架构师带领团队完成。而第二步和第三步，数据的采集和分析，应该自动化来完成，不需要人工的干预。自动化采集数据和分析如何实现，下一讲度量技术里我们再展开说。

小结

到这里，我们总结一下今天学习到的内容。

为了抓住解决问题的重点。我们先对自动化测试的结果做了分析，得出了假阴，假阳，真阴，真阳四种结果。

其中，假阴和假阳是需要解决掉的问题；假阳相当于噪声，浪费自动化测试维护工作量。假阴是 Bug 泄漏，直接影响自动化测试的最终质量。

怎么减少假阴和假阳呢？当然，如果你是一个人做自动化测试，这就靠个人的责任心和技术能力就可以解决了。但如果你所在的是一个规模化的自动化测试团队，那就需要一个可见、透明且合理的方法来推动工作。这就是度量驱动。

但度量驱动的最关键的是设计一个合理的度量指标。针对假阳和假阴，我们探讨了两种实现方式，基于 Bug 的方式和基于 Assert 的方式。Bug 是从结果的来度量，Assert 是从实现过程来度量。而结果容易被人工污染，相比之下，Assert 更具有优势，客观，无人干预，自动化。

有了度量后，就要驱动工作的优化和提升了。这里我把一个周期分解成了四个阶段：度量定义，数据采集，问题分析，优化改进。

在下一讲我会给你介绍度量技术的实现，让整个生命周期能够自动化运行，IT 大佬李国庆对当当网的数据改造项目说过这样一句话“数据是最有价值的，观点是廉价的”。度量的实现，也是自动化测试工作价值的可视化。如何用数据向你的平级和领导描述自动化，比起单纯的观点更有说服力，敬请期待。

思考题

对于假阴和假阳的情况，你是怎么处理和提升的，有没有用到度量呢？如果用到了，请分享你的度量方法。

欢迎你在留言区跟我交流互动。如果觉得今天讲的方法对你有启发，也推荐你分享给身边朋友、同事。

上一篇 27 | 眼见为实：如何用数据描述你的自动化测试ROI ？

下一篇 29 | 落地实践：搭建可持续度量的技术平台

精选留言 (1)

写留言

亭子

2022-05-25

Assert 检查点的失败，是产品错误，其它的失败都属于自动化测试代码健壮性的问题。

个人习惯一般在用例结尾处才去编写Assert断言，那用例执行中出现的如页面加载不出，系统异常等问题，也可能是系统bug，那是不是要每操作一步就写一条断言，那这样会不会造成代码行数太多可读性变差呢

展开