

02 | 如何设计一个“好的”测试用例？

2018-07-02 茹炳晟

软件测试52讲

[进入课程 >](#)



讲述：茹炳晟

时长 16:05 大小 7.44M



在上一篇文章中，我以“用户登录”这一简单直接的功能作为测试对象，为你介绍了如何设计测试用例。现在你应该已经知道，为了保证软件系统的质量，测试用例的设计不仅要考虑功能性需求，还要考虑大量的非功能性需求。

那么，今天我会重点和你探讨如何才能设计出一个“好的”测试用例。

什么才算是“好的”测试用例？

在正式开始讨论之前，我先跟你聊聊，什么才是“好的”测试用例，这个“好”又应该体现在哪些方面。这是一个看似简单实则难以回答的问题，即使深入思考后，也很难有非常标准的答案。

通常，你的第一反应很可能是“发现了软件缺陷的测试用例就是好的用例”，我可能会反问你“如果说测试用例发现了缺陷就是好用例，那么在该缺陷被修复后，同样的用例难道就不是好用例了吗？”。

你可能还会说“发现软件缺陷可能性大的测试用例就是好用例”，这话看起来还是蛮有道理的，但是我同样会反问你“你打算用什么方法来量化测试用例发现缺陷的可能性？”。

类似地，你可能还会说“发现至今未被发现的软件缺陷的测试用例就是好用例”，那么我想问你的是：如何评估是否还存在未被发现的缺陷？如果软件中根本就没有错误了呢？

其实，是你定义“好的”测试用例的思路错了，这就有点像“傻子吃烧饼”，连吃五个不饱，吃完第六个终于饱了，于是他说：早知道吃了第六个就会饱，何必吃前面五个呢。细想，他吃的六个烧饼其实是一个整体，一起吃下去才会饱，而你无法找到吃一个就能饱的“好”烧饼。

对于测试用例其实也是同样的道理，**“好的”测试用例一定是一个完备的集合，它能够覆盖所有等价类以及各种边界值，而跟能否发现缺陷无关。**

我举一个“池塘捕鱼”的例子，可以帮你更好地理解什么是“好的”测试用例。

如果把被测试软件看作一个池塘，软件缺陷是池塘中的鱼，建立测试用例集的过程就像是在编织一张捕鱼网。“好的”测试用例集就是一张能够覆盖整个池塘的大渔网，只要池塘里有鱼，这个大渔网就一定能把鱼给捞上来。

如果渔网本身是完整的且合格的，那么捞不到鱼，就证明池塘中没有鱼，而渔网的好坏与池塘中是否有鱼无关。

“好的”测试用例必须具备哪些特征？

一个“好的”测试用例，必须具备以下三个特征。

1. **整体完备性：**“好的”测试用例一定是一个完备的整体，是有效测试用例组成的集合，能够完全覆盖测试需求。
2. **等价类划分的准确性：**指的是对于每个等价类都能保证只要其中一个输入测试通过，其他输入也一定测试通过。
3. **等价类集合的完备性：**需要保证所有可能的边界值和边界条件都已经正确识别。

做到了以上三点，就可以肯定测试是充分且完备的，即做到了完整的测试需求覆盖。

三种最常用的测试用例设计方法

明白了“好的”测试用例的内涵和外延后，我再回过头来给你讲讲，为了能够设计出“好的”测试用例，你通常都要使用哪些设计方法。

从理论层面来讲，设计用例的方法有很多，如果你去翻阅测试图书或网络教程，会发现一堆让人眼花缭乱的测试方法，比如等价类划分法、边界值分析法、错误推测方法、因果图方法、判定表驱动分析法、正交实验设计方法、功能图分析方法、场景设计方法、形式化方法、扩展有限状态机方法等等，但是从软件企业实际的工程实践来讲，真正具有实用价值并且常用的只有前三种方法。

当然，对于那些与人的生命安全直接或间接相关的软件，比如飞行控制、轨道交通的列车控制、医疗检测相关的软件或者系统，由于需要达到几近变态的测试覆盖率要求，会采用更多的测试设计方法。但对大多数的软件测试而言，综合使用等价类划分、边界值分析和错误推测这三大类方法就足够了。

接下来，我会结合实际例子，给你解释一下这三类方法的核心概念以及在使用时需要注意的问题。

第一，等价类划分方法

从上一篇文章中你已经知道了，等价类中任意一个输入数据对于揭露程序中潜在错误都具有同等效果。后续我们只要从每个等价类中任意选取一个值进行测试，就可以用少量具有代表性的测试输入取得较好的测试覆盖结果。

现在，我给你看一个具体的例子：学生信息系统中有一个“考试成绩”的输入项，成绩的取值范围是 0~100 之间的整数，考试成绩及格的分数线是 60。

为了测试这个输入项，显然不可能用 0~100 的每一个数去测试。通过需求描述可以知道，输入 0~59 之间的任意整数，以及输入 60~100 之间的任意整数，去验证和揭露输入框的潜在缺陷可以看做是等价的。

那么这就可以在 0~59 和 60~100 之间各随机抽取一个整数来进行验证。这样的设计就构成了所谓的“有效等价类”。

你不要觉得进行到这里，已经完成了等价类划分的工作，因为**等价类划分方法的另一个关键点是要找出所有“无效等价类”**。显然，如果输入的成绩是负数，或者是大于 100 的数等都构成了“无效等价类”。

在考虑了无效等价类后，最终设计的测试用例为：

有效等价类 1：0~59 之间的任意整数；

有效等价类 2：59~100 之间的任意整数；

无效等价类 1：小于 0 的负数；

无效等价类 2：大于 100 的整数；

无效等价类 3：0~100 之间的任何浮点数；

无效等价类 4：其他任意非数字字符。

第二，边界值分析方法

边界值分析是对等价类划分的补充，你从工程实践经验中可以发现，大量的错误发生在输入输出的边界值上，所以需要对边界值进行重点测试，通常选取正好等于、刚刚大于或刚刚小于边界的值作为测试数据。

我们继续看学生信息系统中“考试成绩”的例子，选取的边界值数据应该包括：-1，0，1，59，60，61，99，100，101。

第三，错误推测方法

错误推测方法是指基于对被测试软件系统设计的理解、过往经验以及个人直觉，推测出软件可能存在的缺陷，从而有针对性地设计测试用例的方法。这个方法强调的是对被测试软件的需求理解以及设计实现的细节把握，当然还有个人的能力。

错误推测法和目前非常流行的“探索式测试方法”的基本思想和理念是不谋而合的，这类方法在目前的敏捷开发模式下的投入产出比很高，因此被广泛应用。但是，这个方法的缺点也显而易见，那就是难以系统化，并且过度依赖个人能力。

比如，Web 界面的 GUI 功能测试，需要考虑浏览器在有缓存和没有缓存下的表现；Web Service 的 API 测试，需要考虑被测 API 所依赖的第三方 API 出错下的处理逻辑；对于代

码级的单元测试，需要考虑被测函数的输入参数为空情况下的内部处理逻辑等等。由此可见，这些测试用例的设计都是基于曾经遇到的问题而进行的错误推测，很大程度上取决于个人能力。

在软件企业的具体实践中，为了降低对个人能力的依赖，通常会建立常见缺陷知识库，在测试设计的过程中，会使用缺陷知识库作为检查点列表（checklist），去帮助优化补充测试用例的设计。

对于中小企业，可能最初的方法就是建立一个简单的 wiki 页面，让测试工程师完成测试用例的最初设计后对应这个 wiki 页面先做一轮自检，如果在后续测试中发现了新的点，就会继续完善这个 wiki 页面。

对于测试基础架构比较成熟的中大型软件企业，通常会以该缺陷知识库作为数据驱动测试的输入来自动生成部分的测试数据，这部分内容我会在后面的文章中详细介绍。

如何才能设计出“好的”测试用例？

掌握了最基本的三种设计测试用例的方法，你就相当于拿到了打仗所需要的枪支弹药，接下来就是如何在实战中用这些武器打个大胜仗了。

在真实的工程实践中，不同的软件项目在研发生命周期的各个阶段都会有不同的测试类型。比如，传统软件的开发阶段通常会有单元测试，软件模块集成阶段会有代码级集成测试，打包部署后会有面向终端用户的 GUI 测试；再比如，电商网站的测试会分为服务器端基于 API 的测试、中间件测试、前端 GUI 测试等。

对于每一种不同的测试类型，设计出“好的”测试用例的关注点和方法论可能会有很大的差异，有些可能采用黑盒方法，有些可能采用白盒方法，有些还会采用灰盒方法（比如，微服务架构中的测试），所以很难有一套放之四海而皆准的套路。

所以，在这篇文章中，我仅以最常见、最容易理解的面向终端用户的 GUI 测试为例，跟你聊聊如何才能设计一个“好的”测试用例。

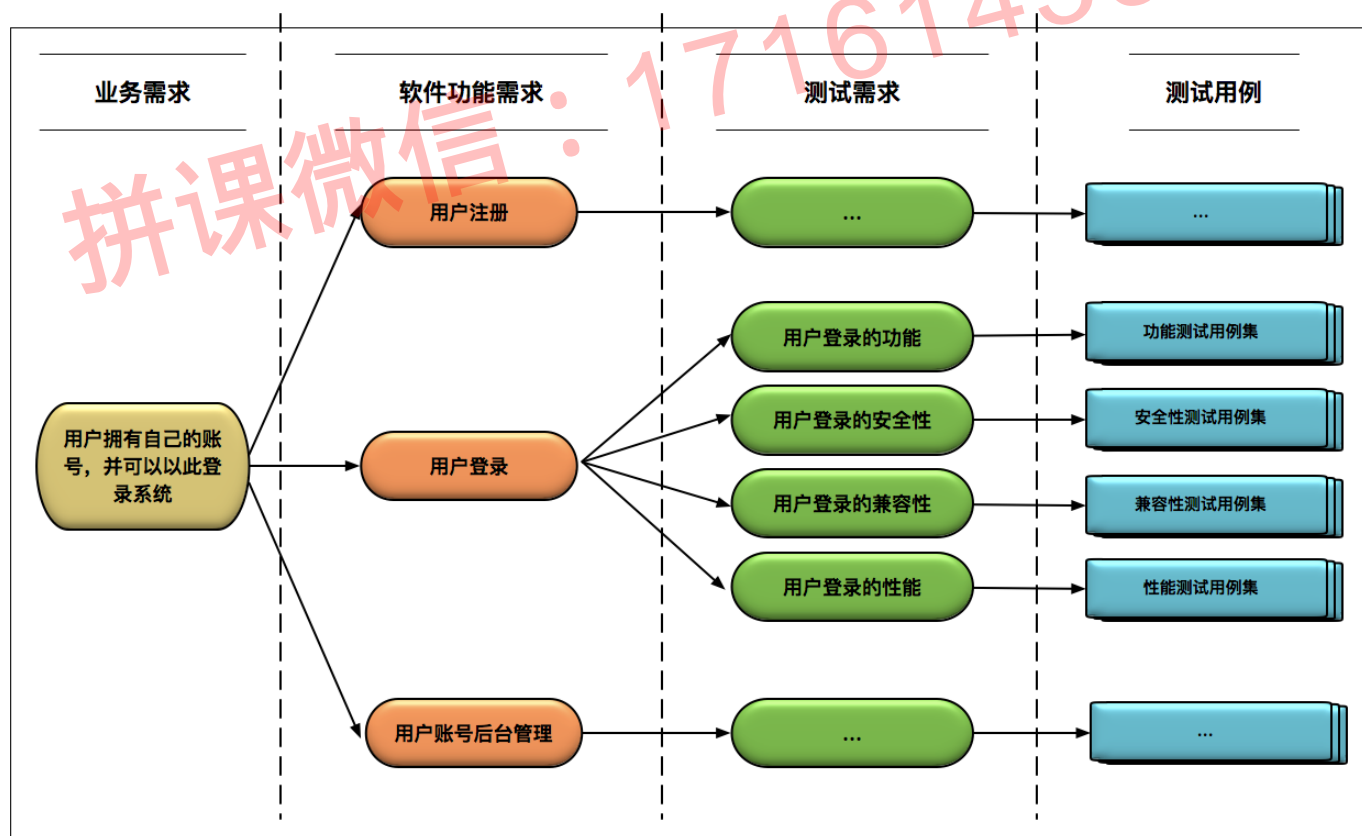
面向终端用户的 GUI 测试，最核心的测试点就是验证软件对需求的满足程度，这就要求测试工程师对被测软件的需求有深入的理解。在我看来，深入理解被测软件需求的最好方法是，测试工程师在需求分析和设计阶段就开始介入，因为这个阶段是理解和掌握软件的原始业务需求的最好时机。

只有真正理解了原始业务需求之后，才有可能从业务需求的角度去设计针对性明确、从终端用户使用场景考虑的端到端（End-2-End）的测试用例集。这个阶段的测试用例设计，主要目的是验证各个业务需求是否被满足，主要采用基于黑盒的测试设计方法。

在具体的用例设计时，首先需要搞清楚每一个业务需求所对应的多个软件功能需求点，然后分析出每个软件功能需求点对应的多个测试需求点，最后再针对每个测试需求点设计测试用例。

这个用例设计过程，你可能觉得有点绕，但是没关系，我以“用户登录”功能的测试用例设计为例，画了一张图来帮你理清这些概念之间的映射关系。

图中的业务需求到软件功能需求、软件功能需求到测试需求，以及测试需求到测试用例的映射关系，在非互联网软件企业的实践中，通常会使用需求追踪管理工具（比如 ALM、DOORS、JIRA、TestLink 等）来管理，并以此来衡量测试用例对业务需求、软件功能需求的覆盖率。



具体到测试用例本身的设计，有两个关键点需要你注意。

1. 从软件功能需求出发，全面地、无遗漏地识别出测试需求是至关重要的，这将直接关系到用例的测试覆盖率。比如，如果你没有识别出用户登录功能的安全性测试需求，那么

后续设计的测试用例就完全不会涉及安全性，最终造成重要测试漏洞。

2. **对于识别出的每个测试需求点，需要综合运用等价类划分、边界值分析和错误推测方法来全面地设计测试用例。** 这里需要注意的是，要综合运用这三种方法，并针对每个测试需求点的具体情况，进行灵活选择。

以“用户登录”的功能性测试需求为例，你首先应该对“用户名”和“密码”这两个输入项分别进行等价类划分，列出对应的有效等价类和无效等价类，对于无效等价类的识别可以采用错误猜测法（比如，用户名包含特殊字符等），然后基于两者可能的组合，设计出第一批测试用例。

等价类划分完后，你需要补充“用户名”和“密码”这两个输入项的边界值的测试用例，比如用户名为空（NULL）、用户名长度刚刚大于允许长度等。

用例设计的其他经验

除了上面介绍的方法外，我再跟你分享三个独家“秘籍”，希望能够帮你设计出“好的”测试用例集。

1. **只有深入理解被测软件的架构，你才能设计出“有的放矢”的测试用例集，去发现系统边界以及系统集成上的潜在缺陷。**

作为测试工程师，切忌不能把整个被测系统看作一个大黑盒，你必须对内部的架构有清楚的认识，比如数据库连接方式、数据库的读写分离、消息中间件 Kafka 的配置、缓存系统的层级分布、第三方系统的集成等等。

2. **必须深入理解被测软件的设计与实现细节，深入理解软件内部的处理逻辑。**

单单根据测试需求点设计的用例，只能覆盖“表面”的一层，往往会覆盖不到内部的处理流程、分支处理，而没有覆盖到的部分就很可能出现缺陷遗漏。在具体实践中，你可以通过代码覆盖率指标找出可能的测试遗漏点。

同时，切忌不要以开发代码的实现为依据设计测试用例。因为开发代码实现的错误会导致测试用例也出错，所以你应该根据原始需求设计测试用例。

3. **需要引入需求覆盖率和代码覆盖率来衡量测试执行的完备性，并以此为依据来找出遗漏的测试点。** 关于什么是需求覆盖率和代码覆盖率，我会在后续的文章中详细介绍。

总结

最后，我来简单总结一下今天的主要内容。

首先，你需要明白，“好的”测试用例一定是一个完备的集合，它能够覆盖所有等价类以及各种边界值，而能否发现软件缺陷并不是衡量测试用例好坏的标准。

其次，设计测试用例的方法有很多种，但综合运用等价类划分、边界值分析和错误推测方法，可以满足绝大多数软件测试用例设计的需求。

再次，“好的”测试用例在设计时，需要从软件功能需求出发，全面地、无遗漏地识别出测试需求至关重要。

最后，如果想设计一个“好的”测试用例，你必须要深入理解被测软件的架构设计，深入软件内部的处理逻辑，需求覆盖率和代码覆盖率这两个指标可以帮你衡量测试执行的完备性。

思考题

在设计测试用例的过程中，你有哪些可供分享的好的实践和方法？

欢迎你给我留言。

 极客时间

软件测试52讲

从小工到专家的实战心法

茹炳晟

eBay中国研发中心
测试基础架构技术主管



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 01 | 你真的懂测试吗？从“用户登录”测试谈起

下一篇 03 | 什么是单元测试？如何做好单元测试？



sylan215 置顶

2018-08-16

👍 12

其实一直以来，我都是把宣称「能发现 Bug 的用例就是好用例」，在讨论用例有效性的时候，我也经常问别人一个问题「你的 Bug 里有多少是通过用例跑出来的」，现在想想，这些都只是针对当前需求修改点而言的用例，毕竟在有效的时间内，用例的有效性还是非常关键的，而且针对的是具体的用例而言。

...

展开 ▾

作者回复: 非常棒的讨论，其实关于什么是好的用例并不存在严格的定义，最关键的是建立全方位以及多角度的思维模式。关于需求合理性测试，我们有对应的UX就是用户体验测试，应该就是对您说的这一点，很好的补充。



么西卡

2018-07-03

👍 37

首先肯定是要保证用例覆盖全面。很多同学写用例就是直接把需求一段段的copy下来就成用例了。写用例重要的是对需求的分解和补充。能够加上自己的经验以及对产品的理解，覆盖各种显性和隐性。功能和非功能的需求。

另外用例还要简洁清晰。你写的用例不仅仅是给你自己看的。要解决用例重复。描述冗...
展开 ▾

作者回复: 非常棒的总结，很到位！厉害👍厉害👍



阿廉

2018-07-02

👍 18

- 1:老师 我上星期由开发转测试 想问一下 您的课程会讲测试用例如何写的吗？
- 2:你讲的对于我来说偏经验和概念 我连测试案例如何写都不会
- 3:或者老师你可以给我推荐一个更基础的测试学习课程或者网站吗 老师老师谢谢您



刘斌宇

在快速迭代的互联网环境下，我们公司测试用例都是采取xmind思维导图的形式。对于移动端来说常见的如安装卸载，版本兼容等常见测试思路可单独提取出来。

展开

作者回复: 是的，列出测试点，我们也经常使用思维导图，我自己用的也是xmind



双子

2018-07-02

9

分享一些个人的想法，首先补充一个角度从用户体验出发完善测试用例，例如一些UI交互设计、push短信发送节点、banner按钮位置、不同客户端的手势快捷操作习惯等，测试人员应该是比产品和开发更了解用户使用习惯的。其次，我觉得测试不仅不能依赖开发也不能过分依赖产品，测试人员应该是对全线产品逻辑细节最熟知的，需求设计的业务漏洞也需要测试来把控，所以需求评审的过程中测试人员就应该能凭经验构思出初步的测试策...

展开

作者回复: 非常棒的分享 你说的这几个点我都非常同意



小志

2018-07-02

9

文章从事前的角度阐述和衡量“好”的概念很系统。大部分项目受限于时间和人力成本，很难完全从覆盖率提升上评估case的“好”。所以另一种思路是从事后的角度，从漏测率和问题严重程度上来评估。漏测率和覆盖率相折中，形成一定的测试策略，最后形成好的case

作者回复: 非常有启发性的观点，感谢和大家分享



wangq

2018-07-11

8

文中很多方法在平时的工作中不谋而合，也是一直强调的。另外，即使是黑盒测试，在测试过程中，与需求确认业务规则，与开发了解代码逻辑，业务规则是准则，代码逻辑是参考，但很多测试人员容易犯的就是跟开发人员确认业务规则

作者回复: 你说得非常正确👍



玉楼人

2018-07-18

👍 5

老师，请问数据库连接方式，缓存分级是如何影响功能的？我们如何设计测试用例呢？能举例说明吗？迫切想知道具体操作。谢谢。



arthur

2018-07-03

👍 4

感谢老师非常棒的分享，我也非常赞同老师说的需要对被测软件的架构和实现细节有很好的理解，这样能更好的进行测试用例的设计。但是对架构和实现细节的理解需要有开发的知识。平时测试工作量很大的情况下，怎样才能提高技术，去了解软件背后的架构和实现方式呢？老师有木有什么好的建议？

作者回复: 测试前置，也就是测试人员去参加系统设计的review会议，一方面可以了解被测系统的架构设计和实现细节，另一方面可以向开发突出可测试性需求



胜杰

2018-07-02

👍 4

@阿廉，测试用例的编写没有固定样式，但大体上是由测试点、测试步骤、期望结果、实际结果等部分构成的。到网上找几份来看看就知道了。

作者回复: 嗯嗯，很好的建议



上进的少年

2018-07-02

👍 4

请教怎么深入理解软件的设计架构？

展开 ∨

作者回复: 首先我觉得先要想明白作为测试为什么要了解软件架构, 然后才能有所侧重的去了解和测试密切相关的架构。

天天开心

2018-09-25

👍 2

Web Service 的 API 测试, 需要考虑被测 API 所依赖的第三方 API 出错下的处理逻辑; 对于这句话, 我想询问老师, 被测试的API接口需要包装第三方API出错的信息吗? 还是直接把第三方API出错的信息抛出来呢?

展开 ∨

才子

2018-08-09

👍 2

结合自己的理解, 测试的目的是对产品质量的验证, 随着现在的敏捷开发, 测试已经转变为质量的提升的推动者。好的用例是用来进行质量守护的, 发现bug的多与少不是衡量测试用例的主要标准, 衡量测试用例的主要标准是需求的覆盖, 功能性需求和非功能性需求。但是需求覆盖率高, 意味着编写时间长, 现在快速迭代的情况下, 测试人员要把握好覆盖率和编写效率的平衡。

展开 ∨

任大树

2018-08-01

👍 2

我最近用到的一个经验:先根据需求和用户使用方式整理出一份用例初稿, 然后找出需求中几个重要节点(时间允许的范围内越多越好), 针对这些节点 请教开发他的实现逻辑(如根据那些数据库表 的哪些字段 做哪些判断等), 此判断逻辑对哪些场景会有关系, 针对这些场景进行发散, 看看自己的用例中有没有遗漏的, 会有意想不到的收获哦

展开 ∨

Kola

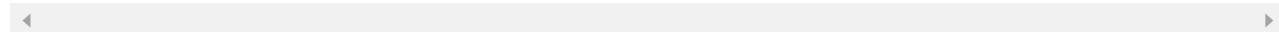
2018-07-30

👍 2

测试是一场考验耐力的运动, 必须坚持到上线前最后一秒, 无痛上线是测试人员毕生的追求, 但是再好的测试用例也跟不上日新月异的需求, 所以上线前临时改的需求必须予以极大关注并且争取更多的时间, 用最快时间评估改动可能涉及的影响点; 最后, 千万不能让开发跳开测试, 总之一句话, 可以没有测试人员 但是不能没有测试!

展开 ▾

作者回复: 很棒



keven

2018-07-16

👍 2

看到老师写的很棒，还有很棒的留言，从这也看出对于用例每个人都有自己的思路和方法，说下个人方法，一个是对需求的理解，要有场景化，对于业务的理解，要有出入口，从哪里来到哪里去，全面覆盖。对于功能，考虑功能本身的实现，考虑关联功能的实现，考虑功能存在哪些状态，每种状态都有哪些场景可互相转换，这是我写用例的习惯

展开 ▾



Cynthia◆...

2018-07-14

👍 2

文中所阐述的缺陷知识库这一条，个人觉得很有意义。虽然最终目的都是一致的，但是对于不同规模的企业，考虑到具体的投入产出比，会有不同的工程实践形式：建立wiki页面，或者作为自动生成测试数据的输入来源。

展开 ▾



红娟

2018-07-04

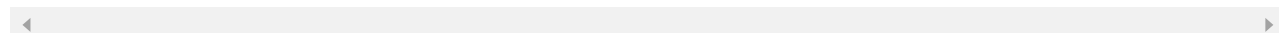
👍 2

我的测试用例一般分为几个阶段

- 1:一般基础阶段，软件实现了什么功能，测试用例一般是围绕功能或者需求来
- 2:边界值，异常值阶段。测试用例围绕在什么条件，不会怎么样？边界值之类，异常之类的测试用例。还有一些场景组合测试用例
- 3:压力测试阶段，如果以上两类测试用例都通过了，那么考虑压力测试用例。比如说长期...

展开 ▾

作者回复: 非常棒👍



arctic

2019-03-05

👍 1

非常棒的课，一个全面，一个深入了解架构，才能设计好用例，这个是要很多年的测试经



小老鼠

2018-10-04

👍 1

1, 错误推测法与探索式测试有相同点, 但不是一回事儿, 现在鼓励使用的是探索式测试法, 而非错误推测法。

2, 对于测试人员而言, 业务学习是比较难的方法, 我测试过机顶盒嵌入式软件、通信系统、财务系统。测试知识可以逐步积累, 但是作为一个好的测试工程师, 对于机顶盒嵌入式软件中的电子技术、通信系统中的无线通信知识(比如GPRS, 3G, 4G知识)、财务系...

展开 ∨