摘要

随着技术的进步，每家企业都将其业务从实体办公室转移到网站和Web应用程序，从而实现在线发展。这带来了一系列更新的测试技术，以迎合最终用户的最佳产品或服务。在启动任何软件，网站或应用程序之前，许多测试技术（例如跨浏览器测试，用户验收测试，回归测试）已变得显而易见，为了确保最佳的用户体验和稳定的功能，还需要一种测试技术是探索性测试。

与其他测试技术不同，探索性测试没有测试人员必须遵循的一组固定方法，但是相反，测试人员拥有发现产品/服务随时间推移不断改进的完全自由。这更像是随着时间的推移以及网站或应用程序的开发而进行的持续改进。

让我们更进一步地探究探究性测试到底是什么，它为何如此重要，如何进行探究性测试，执行它所面临的挑战，优缺点和与其他脚本技术不同的原因的细节和临时测试及其未来。

什么是探索性测试？

顾名思义，探索性测试是基于测试者探索网站或应用程序的能力，以使其随着时间的推移而变得更好。这是敏捷软件开发中的一项重要活动，开发和测试周期是紧密结合的。

探索性测试虽然是黑盒测试，但从整体上考虑了该软件，而没有涉及构成该软件的各个元素的细节。这是一种非常自发的测试方法，测试人员以计划外的方式同时学习，理解，探索和测试软件。与通常在实践测试之前对测试计划，测试用例和测试步骤进行脚本化的脚本化测试相反，探索性测试随着测试人员自行发现和了解网站或应用程序而进行。

它强调测试人员的创造力，自主权和技能，这与其他测试方法遵循固定的方法论方法不同。

为什么探索性测试很重要

探索性测试是实践敏捷软件开发方法时的一项重要活动。在敏捷的冲刺中，该软件是在每几周的时间内发布多个版本而开发的。这意味着开发和测试的时间受到限制，并且需要在更短的时间内完成。因此，为了适应敏捷性，探索性测试的进行小迭代，因为它耗时较少，因此可以通过自动化测试来补充每个版本的软件的质量保证。

自动化测试负责回归测试，而探索性测试主要测试即将推出的版本的新功能。它通过不断学习和使用每个版本来确保强大的功能，更好的用户体验，并通知团队有关可能发生的问题。

如何进行探索性测试

探索性测试涉及发现，调查和学习的紧密结合。因为，它不是预先计划的，与脚本化测试不同，在脚本化测试中，在开始测试软件之前会先制定好测试计划，测试用例和测试步骤。

在探索性测试中，少量的时间用于计划。相反，将最大的时间专用于测试执行。要执行探索性测试，您需要做的就是突出显示您计划涵盖的方案，作为测试计划阶段的一部分。

虽然大多数重点放在测试执行上，但是在整个测试过程中同时进行的关键学习将在测试执行期间实施以增强软件。
在探索性测试执行过程中，通过探索和发现软件来确定关键功能，并记录下所报告的缺陷。这些缺陷将得到进一步分析，以解决和增强产品服务。
探索性测试以这种方式进行，用于敏捷软件开发的学习，测试设计，执行和分析。

有哪些不同类型的探索性测试？

基于该方法的探索性测试，以下是不同类型的探索性测试技术：

1.基于场景的探索性测试

基于场景的探索性测试是指用户浏览并测试特定场景或功能时的情况。基于对网站或应用程序的学习和观察及其功能，测试人员可以使用探索性测试技术来探索和发现不同情况下的缺陷。他们倾向于使用基于方案的探索性测试来检查不同的可能性。

2.基于策略的探索性测试

这种类型的探索性测试的方法基于诸如边界值分析，风险评估，等效技术之类的策略。要执行基于策略的探索性测试，测试人员必须熟悉网站或应用程序功能，以便能够高效地进行操作以获得更好的结果。

3.自由式探索性测试

自由式探索性测试主要用于测试人员想要进行快速冒烟测试的情况。顾名思义，它没有任何明确的测试方法，场景或测试范围，相反，测试人员以自由方式进行调查缺陷。为了能够有效地进行自由式探索性测试，测试人员必须熟悉网站或应用程序，才能在没有任何详细计划的情况下轻松掌握缺陷。
这样，作为测试人员，您可以使用不同类型的探索性测试技术来彻底检查网站或应用程序，以确保改进的产品或服务，以便在每个版本中获得更好的最终用户体验。

探索性测试的优缺点

探索性测试已成为一种现成的测试方法。以下是在测试您的应用或网站时使用探索性测试技术的优点：

- 它不需要大量的测试计划，而这通常是很费时的，这会使整个过程变慢。
- 它与产品/服务的业务可用性和领域非常一致。
- 对于短期项目，探索性测试非常有效。
- 它与敏捷软件开发并驾齐驱。
- 它经常会包含在使用其他技术进行测试时仍未被发现的错误。
- 当需求文档不可用时，这将是有益的。探索性测试技术的最大缺点之一是，它完全依赖于测试人员的技能，因此，如果测试人员的技术水平不高，它就无法产生应有的效果。另一个缺点是由于缺少脚本，通常很难追溯到测

试用例并再次进行测试。

是什么让探索性测试变得困难？

尽管探索性测试看似非常容易，但在执行过程中也面临着一系列挑战。这是在探索性测试期间遇到的一些挑战：

- 由于缺乏文档，经常要追溯缺陷是一项艰巨的任务，尤其是经过一段时间之后。
- 很多测试执行都取决于测试人员的技能，如果测试人员不那么勤奋，可能很难获得理想的结果。
- 它可能不适用于时间表较长的大型项目，因为如果没有适当的正式文档，可能很难涵盖所有可能的范围。
- 它需要具备良好的领域知识和更好的指令，才能深入研究产品并找出错误和缺陷。
- 以后很难复查测试用例。通过克服探索性测试期间面临的上述挑战，您可以使用敏捷方法来增强跨版本的产品/服务。

探索性测试的谬论

探索性测试有很多谬论。让我们揭穿与探索性测试有关的一些常见谬论。

1.探索性测试与临时测试相同

随机测试，指在测试时，抛开用例，按照对需求的理解，对功能进行随机验证，常用于功能提测时，快速发现问题。

实际工作中，随机测试会摒除很多细节，重点验证大的功能点，其随机性在于，每个人验证的思路和方法不同，可以把自己作为一个真正的用户去使用功能。

随机测试抛开了用例的限制，对测试方法和思路没有固定要求，不同的人会发现不同的问题，在功能模块刚提测的时候，对快速发展高优先级的Bug有很大帮助。

探索性测试也是一种撇开用例进行的测试，与随机测试不同，探索性测试往往有具体的思维方向，而且针对的范围更广泛，一般基于大的功能面，而非功能点。

2.探索性测试无法量化

仅仅因为测试计划没有记录在案，并不意味着探索性测试没有任何文件且无法量化。实际上，它更侧重于测试执行，并且已探究的缺陷已得到充分记录。因此，探索性测试能得到了有效的量化。

3.根本没有计划进行探索性测试

只是在探索性测试中没有为测试计划分配太多时间和重要性，而是同时在探索性测试执行之前计划了场景和策略。由于与其他脚本技术不同，它们的文献记录很多，这根本并不意味着完全没有计划进行探索性测试。

4.探索性测试比脚本化测试花费更多的时间

关于探索性测试的一个普遍误解是，它比脚本化测试更耗时，但是实际上，探索性测试所需的时间更少，因为在探索性测试中节省了测试计划和脚本编写的全部时间。由于其消耗较少的特性，因此在敏捷方法中使用了探索性测试，其中两次敏捷冲刺之间的持续时间缩小为一周甚至更少。

5.探索性测试仅适用于小型团队

通常认为探索性测试仅限于小型团队，但这是一种误解。更大的团队也有效地进行了探索性测试，他们与敏捷软件开发的其他测试方法合作。

6.探索性测试仅适用于敏捷团队

尽管大多数敏捷团队都将探索性测试技术与自动化测试一起使用，但这显然并不意味着探索性测试仅限于敏捷团队。实际上，任何正在寻求快速测试会话，同时探索和了解网站/应用程序的议程的软件开发团队都可以有效地使用探索性测试。由于这个原因，探索性测试通常在初创企业中非常受欢迎。

不要与脚本测试技术相混淆

与传统的脚本测试技术不同，探索性测试是一种非常规的测试技术。尽管使用脚本化测试技术可以从需求文档中预先确定好测试用例，但对于探索性测试，则不遵循这些步骤。

与脚本测试主要依赖于确认的脚本测试不同，探索性测试更多地依赖于测试人员在浏览时调查网站/应用程序。探索性测试为测试人员提供了自由和自主的方式，可以按照自己的方式进行操作，而无需遵循与脚本化测试技术相反的任何脚本。与探索性测试不同，文档化仍然是脚本化测试技术的重点之一。

这种方式的探索性测试是一种免费的，即开即用的测试技术，该技术主要基于发现，并且涉及较少的计划和文档编制，从而减少了耗时，并且不同于脚本化测试技术。

不要与临时测试混淆

尽管由于其自由式测试，探索性测试可能看起来像临时测试，但实际上，探索性测试与临时测试有很大不同。尽管临时测试是完全随机的测试方法，但探索性测试更多地是在正式确定要测试的方案。

临时测试需要初步学习，而探索性测试只涉及浏览网站/应用程序，并与测试同时进行。对于临时测试，需要要求文档，而探索性测试则不需要。与临时测试不同，探索性测试需要工作流来执行测试。

通过这种方式，探索性测试与临时测试不同。

探索性测试有未来吗？

脚本化测试方法是进行用户接受度测试的唯一方法的日子已经一去不复返了。随着技术朝着以用户为中心的方向发展，甚至测试技术也必须以相同的方式进行调整，以便能够在每个即将发布的版本中增

强用户体验。这种以用户为中心的软件开发和敏捷方法，为探索性测试以及针对软件产品和服务的自动化测试提供了光明的未来。

鉴于发布新版本的时间紧迫，探索性测试将是与自动化测试一起使用的理想解决方案，以确保功能齐全，功能强大稳定且以用户为中心的软件的质量。

一种软件测试的风格

它是用户故事测试和自动化回归集的重要补充。它是一种经过深思熟虑的测试方式，没有测试脚本，可以使你的测试超出各种明显已经测试过的场景。探索测试将学习，测试设计和测试执行整合在一起，形成一种测试方法。

用户故事(User Story)会是个绝佳的起点，通常实施敏捷的团队会要求创建用户故事，在用户故事中描述需求和产品的行为。测试人员可以向用户故事衍生出的场景中注入合适的变化，通过有系统的考虑输入选择、数据使用和环境条件，一个场景可以衍生出多个测试用例。

探索性测试方法：

指南测试法：城市的题图通常都会标注一些热门的旅游景点，测试这些热门的区域是非常重要的。不管在任何一次发布的过程中，核心功能是肯定要覆盖到的。指南测试要求测试人员严格按照用户手册的建议执行操作，有可能是手册描述不到位或者核心功能并不像宣传的那样好。

卖点测试法：此方法是鼓励测试人员观看销售给客户演示的Demo，理解销售的角度哪些功能对客户来说是最大的卖点，他们未必就是核心功能，但值得测试人员把它们当核心功能来对待。同时，也许刁钻的客户会提出各种质疑，这些质疑和稀奇古怪的问题也可以纳入测试人员的设计中。

地标测试法： 在旅游的时候，如果我们设计了要到访的地方，通常会在地图上插上旗子，这就是地标。但是没有人规定我们应该按照何种顺序去到访这些地标。不同的测试人员可能会选择不同的顺序，有经验的测试人员会基于对软件产品架构和技术层面的理解，采取一些古怪的路径但更可能发现缺陷。

极限测试法：向软件提出难以回答的问题，比如最大可以发挥到什么程度，承受多少用户，承载多少数据。那个特性或功能会把软件逼到极限运作，哪些输入和数据会消耗软件最多的计算能力？哪些输入可能绕过它的错误检测？

快递测试法：快递运送的货物，就好比软件里的数据，结果不同的地点转接，拆包装包最终到底目的地。所以快递测试专注的是数据，跟随它们走遍整个软件。

深夜测试法：城市灯火阑珊会在午夜过后逐渐安静下来，商场店铺纷纷打烊。但是软件不应该停止工作，软件测试人员有时应该刻意的关注在冷门时间段软件所做的附属工作，比如数据备份归档、维护监控任务的运行等等。

博物馆测试法：这是针对软件的遗留代码，保留了些许年代的一些功能，找出它们来验证是否有出现失效。当初开发它们的时候，甚至可能缺乏文档，但这并不意味着它们应该被忽略。

深巷测试法：在每个城市，都有些地方并不吸引游客意味着不吸引人群，但作为测试人员来说，反而是这种最不可能被用到或者最不吸引用户的特性，容易潜藏着难以发现的Bug。

通宵测试法：繁华的都市总会有通宵热闹的地方，比如夜总会KTV之类的，它们从不中断。那么应用程序是否也能坚持到最后呢？当它面临持续不断的调用、输入、重读重写之类的操作，如果运行时间

足够长，就很可能会出问题，内存会需要回收、数据需要清空，永远不要关闭它，保持不间断的运行。（更多的时候会采用自动化或者机器人思想）

长路径测试法：把那个在应用程序埋藏最深的界面当做测试目标（离起始点最远的那个界面），观察经过的每一个界面

超模测试法：针对UI的表面测试，衡量软件的展现能力，像T型台的超模那样，不去关注她们幕后的辛酸痛苦劳累，跳出产品专家或测试的头衔，以普通观众的角度，去关注那些能看到的界面展现，元素是否被正确绘制、是否难看、颜色风格是否一致、界面的切换变化是否表意清晰？

取消测试法：此方法的思想是启动了立即停止，特别是一些运行流程比较耗时的功能如备份还原或者搜索，在启动之后，立即取消。发散一点还可以变成，启动一个耗时操作，不停止立即启动另外一个耗时操作，以此来检测应用程序的自我清除能力。

懒汉测试法：选择尽可能少的输入，能不输入尽量不输入，能不修改尽量不修改，观察应用程序是否能响应得出正确结果。

反叛测试法：你见过去酒吧不喝酒点果汁的么？反叛思想要求输入最不可能的数据，或者已知的而已输入，测试人员可以采用逆向思维、明知一些数据违反规则，却偏偏要采用这样的数据，或者不按照正确的顺序来输入。

强迫症测试法：测试人员强迫软件一边又一边接受同样的数据，反复执行同样的操作，最重要的特点就是重复。此种思维方式，常常打破了开发人员设计代码的思路，他们预想着你会按步骤操作，却不曾考虑过你反复的执行第一步应该如何处理。