Exploratory testing

Learn what exploratory testing is and its history. Discover the pros and cons of exploratory testing, how it differs from scripted testing, and the appropriate context to use it.

BY DEEPAK PARMAR

Exploratory testing is an approach to software testing that is often described as simultaneous learning, test design, and execution. It focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.

The practice of exploratory testing has gathered momentum in recent years. Testers and QA managers are encouraged to include exploratory testing as part of a comprehensive test coverage strategy.

History of exploratory testing

Exploratory testing has existed for some time but was often referred to as 'ad-hoc testing'. The term "exploratory testing" was formally introduced by software testing expert Cem Kaner in his classic book, Testing Computer Software.

The introduction is now famous: "No matter how many test cases of how many types you've created, you will run out of formally planned tests. You can keep testing. Run new tests as you think of them, without spending much time preparing or explaining the tests. Trust your instincts."

Why use exploratory testing

Teams today need to adopt continuous integration and deliver on the market demand of quality digital experiences to meet rising customer expectations. While speed to market is important, there are instances of million-dollar bugs or simple user experience disasters that are very costly. From Boeing to Instagram, there are plenty of examples where the rush to deliver on deadline and poor-quality testing led to reputational and financial damage.

Most software quality testing uses a structured approach. Test cases are defined based on already defined user stories and the test data is structured based on the test cases defined. Test coverage is measured using software engineering metrics and, in most cases, the coverage is adequate technically.

What it often misses are edge cases, which are discovered through User Acceptance Testing (UAT) and are tested based on user personas. On the other hand, exploratory testing is random or unstructured in nature and can reveal bugs that would go undiscovered during structured phase of testing.

With exploratory testing, testers can play around with a user story that follows a certain sequence. Testers can annotate defects, add assertions and voice memos, and create documentation on the fly. This is how a user story is converted into a test case. This information can also be used for QA.

Effectively, test execution is implemented without formally authoring test steps. The exploratory testing tool then becomes a precursor to automation. It helps formalize the findings and document it automatically. With the help of visual feedback and collaborative testing tools, everyone can participate in exploratory testing. This enables teams to react and adapt to changes quickly – facilitating an agile workflow.

Furthermore, the tester can convert exploratory testing sequences into functional test scripts using tools for automated test case documentation. This reinforces the traditional testing process.

By integrating with tools such as Jira and test management

products, teams can directly export the recorded documentation to test cases.

So, exploratory testing speeds up documentation, facilitates unit testing and helps create an instant feedback loop. As James Bach, co-founder of the Context-Driven School of Software Testing puts it, "exploratory testing encourages scientific thinking in real time."

When should you use exploratory testing?

Exploratory testing is suited for specific testing scenarios, such as when someone needs to learn about a product or application quickly and provide rapid feedback. It helps review the quality of a product from a user perspective.

In many software cycles, an early iteration is required when teams don't have much time to structure the tests. Exploratory testing is quite helpful in this scenario.

When testing mission-critical applications, exploratory testing ensures you don't miss edge cases that lead to critical quality failures. Plus, use exploratory testing to aid unit test process, document the steps and use that information to test extensively during later sprints.

It is especially useful to find new test scenarios to enhance the test coverage.

When to say no to exploratory testing

Organizations must be able to strike the right balance between exploratory testing and scripted testing. Exploratory testing alone can't offer adequate coverage and teams shouldn't attempt it unless they have reached a few initial milestones.

Especially with any type of testing that is regulated or compliance-based, scripted testing is the way to go. In compliance based testing, where certain checklists and mandates need to be followed for legal reasons, it is advised to stick to scripted testing. One example of this is accessibility testing where several laws govern the testing protocol and there are defined standards that need to be passed.

Challenges of exploratory testing

Quantifying the benefits of exploratory testing is one of the biggest challenges because the gains are more qualitative than quantitative. The aim is to find issues that have escaped other forms of testing. The expectation should be to document unexplored test scenarios and increase coverage.

Another challenge is the ability to log defects and document these with proper evidences while the tester is intuitively exploring undocumented test scenarios. Providing a tool that helps to automate the test case documentation, allowing testers to take screenshots, record voice memos, annotate is a big boon.

Additionally, it is a highly skilled testing approach that needs experienced software testers for planning and execution. Discovering the edge cases requires a certain amount of innovation and out-of-the-box thinking on a testers' part.

Importance of exploratory testing for CI/CD

Exploratory testing opens testing to all key stakeholders and not just trained testers. Using an exploratory testing tool, one can capture screenshots, record voice memos and annotate feedback during sessions. This enables faster and more efficient review, by people beyond the traditional software tester.

Exploratory testing complements QA teams' existing test strategy. It comprises a series of undocumented testing sessions to discover yet unearthed issues/bugs. When combined with automated testing and other testing

practices, it increases test coverage, discovers edge cases, potentially adds new features and overall improves the software product. With no structural rigidity, it encourages experimentation, creativity and discovery within the teams.

The almost instantaneous nature of feedback helps close the gaps between testers and developers. Above all, the results of exploratory testing provide a user-oriented perspective and feedback to the development teams. The goal is to complement traditional testing to find million-dollar defects that are generally hidden behind the defined workflow.

Get a test management app from the Test Management collection on the Atlassian Marketplace.

SHARE THIS ARTICLE

DEEPAK PARMAR

I've lived and breathed QA for the last decade now through my experience of working with leading QA services and product companies. I'm currently the head of Marketing and Partnerships at QMetry, bringing with me 20 years of experience in the IT industry, which has instilled in me the strong belief in improving customer delight through software quality.

Bitbucket CI/CD tutorials

Continuous Delivery articles

ARTICLE

The different types of testing in Software

Compare different types of software testing, such as unit testing, integration testing, functional testing, acceptance testing, and more!

Read this article →

ARTICLE

Introduction to Code Coverage

Learn how to get started with code coverage, find the right tool, and how to calculate it.

Read this article →

Up Next

Introduction to Code Coverage →