Abstract

Scripted testing alone often fails to find hidden or divergent risks in a product under development. Exploratory testing, however, can expose these risks because it incorporates human intuition and experience into the testing process.

This white paper examines a few ways that adding exploratory testing to your testing regimen can improve your test coverage and help reduce risk.
The Problem with Scripted Testing

Many testing teams rely solely on scripted testing, both manual and automated, to decrease the risk of defects in the product release. The problem is, scripted testing is not meant to identify error conditions in scenarios that significantly deviate from the design or requirements. To find these hidden or divergent risks, you need to go off script, and that's where adding exploratory testing can help.

It's almost impossible to plan tests that cover every variation in data, configuration, interaction, sequence, timing, and so on. Scripted tests are designed to ensure that the application meets the requirements (using new feature test cases) and to mitigate the risk of new features breaking existing functionality (via regression test cases).

Experienced testers can anticipate issues that might occur, but it may be too costly or time-consuming to write a test case for every scenario that comes to mind.

In her book *Explore It! Reduce Risk and Increase Confidence with Exploratory Testing*, Elisabeth Hendrickson says the best test strategy answers two core questions:

1. Does the software behave as intended under the conditions it's supposed to be able to handle?
2. Are there any other risks?

Scripted testing can answer the first question, but to find those potentially critical “other risks,” you need to explore.

Push the Envelope

Exploratory testing puts the thinking back in the hands (or rather, the head) of the tester. As an exploratory tester, you design the test, you execute it immediately, you observe the results, and you use what you learn to design the next test. You're not just following steps in a test case someone else created; you're pushing the application to its limits to gain a better understanding of how the application works and where it breaks. You see it from the user's point of view, rather than the developer's.

Ultimately, you get a more complete view of the application—including its weaknesses and hidden risks.
Discover More Defects

On average, 11% more overall defects are discovered through exploratory testing vs. scripted testing. For defects that should be immediately obvious, such as a missing button in the UI, exploratory testing discovers 29% more vs. scripted. When it comes to “complex” bugs (bugs requiring three or more user actions to cause an error or failure), it jumps to 33% more defects found. *(Source: Defect Detection Efficiency: Test Case Based vs. Exploratory Testing.)*

The reason you find more defects when using an exploratory method is because you have more latitude to try different types of tests, using your past experience and knowledge of the product. Scripted testing, on the other hand, limits you to only the steps outlined in test cases, which in turn limits your ability to consider other test scenarios.

There are numerous reasons why test cases don’t always lead to finding bugs, such as: how well the test case was written (did the analyst understand the requirement?), who wrote the test case (is the analyst writing the test case knowledgeable about how the product works?), how well the requirements document described new functionality, and so on.

Even if you had perfect test cases, exploratory testing would still find more defects over the course of a release, for several reasons:

- You tend to find a good number of defects when “testing around” functional areas while verifying defects. Fixing an issue often breaks something else.
- If a defect exists and is not found while executing the initial test run (following the test cases steps), it is unlikely that the next tester running the same test will find the defect. However, exploratory testing in the same functional area may reveal the bug.
- Exploratory testing allows you to think outside the box and come up with use cases that might not be covered in a test case. For example, you might perform one test and then ask yourself, “What if I tried this? What if I didn’t do that?”
- Some defects, typically the hard ones to find, are dependent on a sequence of events. If you don’t have really deep test cases, you can miss finding defects that exploratory testing can find in a less-structured, but longer, test session.
Find the Most Important Defect in the Shortest Time

Because you are not bound by the test case steps, exploratory testing makes finding important defects faster; it allows you to cover more ground and focus on testing the “what ifs.”

For example, let’s say you’re assigned to test the “Edit Bookmark” functional area in a product. There are two possible scenarios for your assignment:

1) **Execute a test run**

-OR-

2) **Perform exploratory testing of the “Edit Bookmark” functional area.**

In the first scenario, you would follow the steps outlined in the test case to verify that each step works, reporting any bugs that are found. But what if the most important bug—the one that crashes the application and deletes the user’s data, say—doesn’t occur during these steps? You won’t find it.

In the second scenario, you explore or “test around” editing bookmarks. At first, you might perform the same steps as at the beginning of the test case. As testing progresses, however, you might ask, “What happens when I click Edit, delete the name, and try to save the bookmark with a blank field?” And boom, the application crashes and deletes all of the user data. You just found the most important bug by exploring something that wasn’t a step in the test case.

### Exploratory Testing Feeds Scripted Testing

When exploratory test sessions are recorded, they can easily be converted into repeatable test cases or regression tests without the need for you to take notes in a separate document or notebook.

A test session recording application records all activity and builds a detailed history of the test session, including descriptions of the user interface controls used and a screen shot of every step with the relevant graphical user interface (GUI) element automatically highlighted.
Once you’re done exploring, the app generates a step-by-step written script of the test steps in plain English. You can then save this script as a test case in your test case management solution.

Test session recording apps simplify the defect reporting mechanism for risks found in exploratory test sessions, because they make it easy for you to go back and reproduce the error. Steps and screens are automatically captured, so there’s no need to do all of that manually when submitting a bug report to the developer.

**Keep Exploring**
This white paper only covers a few ways that adding exploratory testing to your testing regimen can improve your test coverage and help reduce risk. There are many, many other advantages not covered here.

If you want to learn more, you can easily find a wealth of details from experts such as James Bach, Cem Kaner, Michael Bolton, Elisabeth Hendrickson, and many others.

**Fill the Gap**
As corporate belts tighten and shrink testing budgets and staff, you might be tempted to settle for an incomplete test approach because you know it will satisfy your company’s quality standards or auditors. All that really does, however, is open the door to a buggy release.

Combining exploratory testing with mandated scripted testing can help fill the gap left by shrinking resources to keep your test effort strong and reduce the risk of defects in the release.
About Seapine Software
With over 8,500 customers worldwide, Seapine Software, Inc. is the leading provider of quality-centric product development solutions. Headquartered in Cincinnati, Ohio, with offices in Europe, Asia-Pacific, and Africa, Seapine’s development solutions help organizations ensure the consistent release of high quality products, while providing traceability, metrics and reporting, and compliance.

Learn more at seapine.com.