Finding Bugs Under Software Rocks - Exploratory Testing as a Bug Hunt

By Derrick Collins, Software Tester
September 29, 2014

One aspect of software testing is trying to find as many of the bugs in the software under test as possible and, in that sense, software testing can be viewed as a "Bug Hunt". As a metric for evaluating the effectiveness of our "Bug Hunts", we can measure the total number of bugs found, with more value being given to bugs of a higher severity. We should pay careful attention to where in the software we're looking for bugs by selecting particular testing styles in our test strategies. One testing style we often find ourselves using is scripted testing, which brings with it a certain kind of a "Bug Hunt" that is different than when using exploratory testing. When we do find ourselves engaged in scripted testing, do we still allow ourselves to look under any rocks in our software for hidden bugs, or are we being so rigid in how closely we follow the steps in our test cases that we're essentially limiting ourselves to only finding bugs "hiding in plain sight"?

"more of a spectrum of testing styles rather than two distinct styles"

A Few Term Definitions

We think of software testing as having two main styles – scripted testing and exploratory testing – but it is really more of a spectrum of testing styles rather than two distinct styles. We don't have to limit ourselves to thinking of testing as being either purely one or the other, but instead we can consider where on this spectrum of testing styles we may want to land, both while we are devising our test strategies and while we are performing our testing.

Scripted Testing: This testing style, in its purest form, involves separating the activity of designing our tests from the activity of executing them. Our test cases are often detailed down to each test step that will be executed and the expected results of each step. Our completed test cases are then executed, as they were designed.

Exploratory Testing: This testing style involves merging the activity of designing our tests with the activity of executing them into a single activity where the test design and the test execution guide each other as we explore specific areas of the software.

A Comparison of Styles

Considering the above definitions of scripted and exploratory testing, we can begin to see, through comparison, that each testing style may have distinct strengths and weaknesses when we use them for "Bug Hunting".

Scripted testing is a structured activity often based on validating the software requirements and related software design documentation. While creating our test cases, we consider the bugs we expect to find and where we expect to find them. On executing our test cases, it's likely we'll find exactly the kinds of bugs the test cases were designed to find in exactly the places we're expecting to find them. It's also likely that any kinds of bugs we hadn't expected to find while designing our test cases in places we hadn't expected to find them will go unnoticed. Additionally, the bug finding potential of particular test cases will be exhausted if we continue executing them, since our testing will essentially remain focused on hunting for bugs that have already been caught.

"following those observations wherever they may lead us"

Exploratory testing is an investigative activity of changing our test activities based on what we actually observe while we're interacting with the software and then following those observations wherever they may lead us, which we hope will be places in our software where bugs are hiding, just waiting for someone to come along and find them. Software requirements and related software design documentation still guide our testing, however, the primary factors for finding bugs during testing are simply the tester's own observations from interacting with the software, combined with their intuition and prior experience regarding where to seek out potential bugs.

I'd like to propose that it's entirely possible to perform scripted testing using an exploratory mindset if we allow ourselves to use our instincts to know when straying a bit from a scripted path may uncover additional bugs that wouldn't otherwise be uncovered.

Staying on the Path or Straying from the Path

Imagine we're observing a pair of software testers on an adventure through the first build of an application and that they are going to perform manual scripted testing. Let's think of the software as a bug-infested forest that's waiting to be explored, and this round of the manual test effort is a purposeful walk through that forest seeking out these bugs.

Tester number one follows very closely to each of the paths that have been laid out in the test cases. He or she will likely find all of the bugs that are "hiding in plain sight", in the precise spots where each of those test cases has been designed to find them. Tester one is also able to execute the testing at a fairly steady pace, thanks to following so closely to the scripted path of each test case.

"allow ourselves to use our instincts"

Tester number two isn't following the paths defined in the test cases as closely as tester one. A few of the test cases lead down minor variations of paths that tester number one has already been down without finding any bugs. Tester number two spots something kind of odd out of the corner of his or her eye. Tester two has been doing a lot of that while testing, which is likely why he or she hasn't been proceeding at a rapid pace through the assigned test cases. In fact, tester number one also briefly spotted this same oddity, but since he or she was focused on following the path in each test case, he or she didn't pay too much attention to it. Tester number two pursues this distraction, the test case execution having been put on hold while leaving the scripted path for a bit of exploration in this particular area. Sure enough, that oddity is a rock, and there's something strange about the way it's moving. It's at this point that our explorer picks up the rock, curious to see what may lie underneath it. Ah hah, it's a bug! The bug list grows by one. In fact, this turns out to be a major bug which was much better found sooner rather than later.

I leave you with some points to consider as we try to evaluate the pros and cons of scripted and exploratory testing. Considering that tester number two found the only major bug during this first round of testing, could we conclude that their "Bug Hunt" was the more successful one, even if they didn't execute as many test cases as tester number one? Considering the outcome of this round of testing, how could this experience influence your test strategy going forward?

About The Author

Derrick Collins is a Software Tester with Professional Quality Assurance Ltd., having over 13 years of experience providing software QA solutions for a variety of industries, including video game development and health insurance. Derrick is knowledgeable in many software testing techniques, such as black box software testing, exploratory testing and designing test cases based on documented functional specifications and business requirements. He is also responsible for test execution, defect reporting, and test tracking.