THOUGHTS ON EXPLORATORY TESTING AND SCRIPTED TESTING

BY MARINA GIL SANTAMARIA
SUMMER 2007

Exploratory testing (ET) "in lieu of/or complimenting" scripted testing continues to be a hot topic of discussion in the industry. If you have recently attended any testing conference or event, read any recent publications, or participated in any QA professional local association meeting, you have probably witness some type of "heated" debate about this topic, with strong proponents of one approach versus the other. I personally believe that, even thought the concept of exploratory testing has been around for a while, there is still a lot of confusion and misperceptions about what ET really is, and the value it can provide to any testing organization.

The purpose of this write-up is to take a closer look at exploratory testing and scripted testing and it's respective advantage and disadvantage (yes, there are benefits and drawbacks on both approaches), and provide some recommendations and guidelines about when you should consider using which type of testing approach. On an additional note, because Empirix offers tools to help you manage and organize your manual testing efforts, as well as automation and performance testing tools, we feel that we are well positioned to provide an unbiased and "neutral" point of view when looking at both types of testing types.

What is Exploratory Testing?

In his article, "Exploratory Testing Explained"[1], James Bach defines exploratory testing as, a "simultaneous learning, test design, and test execution". One could think about exploratory testing as a technique—or a sophisticated, thoughtful, and creative approach— to perform ad hoc testing without having a pre-defined set of test cases. Exploratory testing is not new; in fact, exploratory testing as defined by Cem Kaner[2] has been around for 23 years.

In "freestyle" exploratory testing, the only result of the testing activities is a set of bug reports. In "session-based test management" exploratory testing, a set of written notes for managing and measuring tests are produced at the end of each exploratory session, which are then debriefed by a test lead. And as Jon Bach mentions in our QAZone Interview, a session is a unit of time roughly 45 minutes to 3 hours long where the tester is guided in their exploratory by a charter or mission statement[3]. In both "freestyle" and "session-based management" exploratory testing, no specific results are expected; the tester decides what needs to be verified, critically investigating the accuracy of results as needed. In addition, because there are not too many formal processes in place, and there are certainly less associated QA documents and overhead, exploratory testing

can provide meaningful results quickly. (One common conviction among the exploratory testers community is their belief that writing down test scripts could disrupt the intellectual processes needed to find important bugs quickly4).

Thinking of my time as a Software QA Engineer, I realized that we have all done exploratory testing at some point in our professional career, without realizing that we do it –unless you simply don't create tests at all. Think about your first weeks as a new QA team member, and your tasks then…You were probably exploring product documentation and exercising the product to learn it. In fact, the majority of your written test procedures were probably created through a process of performing exploratory testing, while investigating the new features and functionality available on a new product release. So if exploratory testing is so common, why is it still the subject of many industry discussions and debates, and even a little bit of controversy?

I really think that the problem here is that folks try to generalize and evangelize one versus the other – exploratory testing OR scripted testing –, when in an ideal world a software project could typically benefit from a combination of both testing approaches. Only in a very few cases and under very specific circumstances will you effectively achieve your testing mission by using only one approach or the other. Some of the conditions that will make you lean towards one approach versus the other are related to:

- QA team size, skills, and seniority
- Product stage (stable, mature, end of life, etc.) as well as future roadmap
- Your team testing objectives (compliance, regression, functional testing, performance testing, sanity checks, etc.)
- Environmental complexity
- Nature of your applications
- Your own development cycles and schedules

Advantages and Disadvantages of Exploratory Testing

Exploratory testing is built on the philosophy that people are "finite capacity information processors"; therefore, when they are focused on some things (such as testing scripts and expected results) they don't pay attention to others (and could miss important bugs). This is what Arien Mack and Irvin Rock called "*Inattentional Blindness*"[5]. (You can read more about inattentional blindness and other experiments conducted by Daniel Simons and Christopher Chabris here as well here http://www.jimcarson.com/2006/11/inattentional-blindness/).

The main advantage of exploratory testing is that less preparation is needed, important bugs are found fast, and it is more intellectually stimulating than executing test scripts. The main disadvantages are that tests usually depend on the skill, experience, and intuition of individual testers, and therefore they may not be reviewed in advance, tracked, or easily repeated. In

addition, because exploratory testers typically build mental maps and models in their heads rather than on paper, their skill and knowledge leaves a project when they leave. This is something that you should really consider when working with mission-critical applications, or with products that have long-term plans.

Keeping all of these in mind, exploratory testing is better when:

- Testing starts very late in the project development cycle, or when requirements and specifications are incomplete or non-existent. (This is not the ideal working environment, but as I am sure you already know by experience, it does happen.)
- Your goal is to find new errors quickly, and you don't need to perform comprehensive functional testing (e.g. you are working with mature products that already have had multiple releases.)
- You want to provide quick feedback about a new feature to your development counterparts.
- You have already tested using scripts and would like to broaden your testing cases.
- Your QA team is very senior. (Exploratory testing relies on an individual tester's ability, so a very skilled QA team is required to perform it. Another option would be for senior and junior engineers to work collaboratively and take advantage of cross-skill pollination.)
- Your QA team is relatively small and communication is working well. (With exploratory testing it might be difficult to know exactly which tests have already been run and for which product version.)
- You are writing new test scripts or adapting existing scripts to changing conditions, such as new product features, new platforms, new bugs, etc.)
- You want to double-check the work of junior engineers by doing an additional brief examination.
- You want to evaluate risks and the need to write additional scripted tests for specific product areas.
- The product is not stable and is not ready for automation yet.
- The product is a one-time project, and no future development is planned, or you are working with a mature product that is entering its end-of-life stage.

Advantages and Disadvantages of Scripted Testing

Scripted testing is the opposite of exploratory testing, in the sense that it takes a more formal approach, as you are manually testing by following predefined test procedures or by running automated scripts. With twenty-five percent of organizations using both manual testing and automation and with a growing number of QA teams tackling automation projects, it is likely that you will perform both types of scripted testing over the long term. During the design phase,

QA teams create test cases, test specifications, test plans, and scripts, which are used later to conduct formal testing during the execution phase.

The main advantage of scripted testing is efficiency, repeatability, and reusability, so once you have invested resources and cycles during the design phase, testing is much faster. (It is likely, however, that you were doing exploratory testing when building your test cases and scripts). And if you have automation in place, it can really help you increase test case coverage and cover a broader test plan without slowing down the development cycles because automated scripts can be run before, during, or after work hours. Also, if you are working on a performance testing project, you will definitely need to automate and run scripts to simulate various loads and multiple virtual users.

On the other hand, the main disadvantage of scripted testing is the time that it takes to document and write test cases and scripts in the first place, as well as the additional overhead that more rigid QA processes can add to any development project.

Keeping these advantages and disadvantages in mind, scripted testing is better when:

- Your testing starts very early in the development cycle, so you can leverage this time to build and review test scripts, and modify them based on input from development personnel and other QA peers, or as product requirements and specifications evolve.
- You have limited access to specific hardware and/or software resources, so you want to carefully plan and execute test scripts to maximize limited access to shared resources.
- Your QA team has engineers with varying levels of experience and skills (keep in mind that processes tends to even out testing activities across engineers, and you could have senior folks working on the design phase while junior members execute your testing scripts with guidance from senior QA engineers.)
- You are working with a product that will have multiple minor releases, and there are no plans for drastic changes in functionality, or you are working with a product entering an introduction or growth stage with plans for continuous investment. (In this scenario, it could be really advantageous to invest in building repeatable scripts and processes that can be leveraged in subsequent product releases.)
- You need to perform in-depth functional testing and regression testing. (You really need formal test plans and documented test cases to achieve this, so you can quickly identify which bug was associated with each product test release and what has changed across product versions.)
- You are working with very complex environments and n-tier applications that are deployed across multiple systems and platforms. (With carefully documented test scripts you can ensure that your tests cover a comprehensive set of pre-conditions and post-conditions for all possible end-to-end scenarios.)

- You need to verify compliance. (Formal test scripts describe expected outcomes as derived from the requirements and specifications documents, and therefore, you can measure compliance.)
- You are working with mission-critical applications. (Exploratory testing carries a risk, since full testing coverage cannot be truly guaranteed.)
- You are working on a performance testing project, so you definitively need scripts and automation to test your application under various loads.

Conclusion

There are only few situations when it is clearly preferable to use solely one type of testing; typically throughout a development project you will benefit from using both scripted and exploratory testing at different points of your cycle. For example, you might want to do exploratory testing at the beginning when a product is not ready for formal scripted testing. Similarly, if you have been conducting exploratory testing, by the time you move into a performance testing cycle you may want to use less labor-intensive testing through automation and scripts. In summary, both exploratory testing and scripted testing are complementary and deliver great value to any QA or testing organizations. Thus, the main issue is not really taking a decision of exploratory testing only, or scripted testing only, but taking the right decision of at which point of your testing cycle is better to use one testing approach or the other. And if your project is being developed within a large corporation, chances are that you will have two teams of QA engineers running exploratory tests and scripted tests in parallel!

References:

[1]*Written by James Bach in 4/16/03 is available here http://www.satisfice.com/articles/et-article.pdf. An interview with James Bach is also available in QAZone here http://qazone.empirix.com/thread.jspa?threadID=277&tstart=0*

[2]*Cem Kaner, J.D., Ph.D., is a Professor of Software Engineering at Florida Institute of Technology, and the Director of Florida Tech's Center for Software Testing Education & Research (CSTER) since 2004*

[3]*Jon Bach, QAZone Interview available here http://qazone.empirix.com/thread.jspa?threadID=217&tstart=0*

4 As mentioned in http://www.stickyminds.com/sitewide.asp?ObjectId=2255&ObjectType=COL&Function=edetail

5 See http://www.scholarpedia.org/article/Inattentional_Blindness