

Exploratory Testing – An Agile Approach

“STC-2009”

Aman Arora

Xebia IT Architects India Pvt. Ltd.

Unit No-612, 6th floor, BPTP park Centra,

Sec-30, Gurgaon – 122001, Haryana

Abstract

As the IT industry is growing and getting older more mature approaches for the development of the software are emerging which are more effective and fast as compared to the traditional one. The organizations that are traditional invest heavily in implementing test methodologies. These methodologies usually require testers to start writing the test cases as soon as the requirement document is there. In this approach there are many formalities such as creating the documents for each step and then maintaining them.

Now organizations are moving towards more agile, informal and an iterative approach – an “Agile approach”. Being informal, testers are not required to write the test cases rather they can concentrate more on exploring the application and can focus on testing it. This approach of testing and exploring the application is referred to as ‘Exploratory Testing’. It’s a technique through which one can learn the application and can uncover most of the bugs at an early stage and then can add new and better test cases for the functionality.

This paper concentrates more on the Exploratory Testing as a technique, suggests how and when ET should be included in the testing strategy, in what situations it is most useful, and which aspects of the testing process should never be exploratory or agile. This paper also throws light on a concept called “paired testing”.

Exploratory Testing – An Agile Approach

Introduction

In the past, for the traditional methodologies usually a tester is required to create test scripts and test cases using a test basis such as a requirements document, in a process known as ‘scripted testing’ (ST). A lot of time is invested in creating the formal documents, reviewing these docs and then the testing is started.

Now the companies are moving towards faster approach of designing, developing and testing of software. This approach is more agile, informal and iterative such as eXtreme Programming¹ and Scrum². The new methods deemphasized the use of documentation and challenged the reliance by traditional test approaches such as ST on documentation as the test basis. Furthermore, ST led to excessive testing of easy cases, while failing to allow sufficient scope for creative testing of more interesting and complex cases.

A new technique, referred to as ‘exploratory testing’ (ET), was introduced by *Cem Kaner in 1999* and further elaborated by *James Bach* in recent years. Exploratory Testing, as the name signifies is an approach to explore the application and simultaneously testing it. For various reasons, many people still associate ad hoc testing with an aimless black box approach. In practice, skilled testers use methods that are neither aimless, nor restricted to black box methods. The old notion of ad hoc testing is currently enjoying a metamorphosis into the more highly evolved approach called Exploratory Testing.

Many debates are going on in the software industry between ST and ET, documentation against no documentation, procedures against freedom, uniformity against creativity, and manageability against efficiency.

Defining Exploratory Testing

ET is a process all software testers know and have been using for decades. James Bach calls it “scientific thinking in real time” and defines it as:

Simultaneous learning test design and test execution; any testing to the extent that the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests.

or it can be defined as following

Exploratory testing is a style of software testing that emphasizes personal freedom and responsibility of the tester to continually optimize the value of their work by treating test-related learning, test design, and test execution as mutually supportive activities that run in parallel throughout the project.

In ET, the tester explores a piece of the system, thinks about what should be tested and how, and then carries out the appropriate tests. In the process, the tester learns about the system and its behaviour, develops new tests based on what he or she has learned so far, executes these tests, and so on. Test design and execution occur more or less concurrently, often without documentation. The process is a major departure from ST, where tests are first designed and documented and then executed at a later time, not always by the same tester. ET encourages the tester to be creative and flexible: to act intuitively. It is investigative rather than confirmative. The tester is free to explore new areas as test ideas arise during test execution. As a result, it significantly increases the chance of finding bugs that might have gone undetected using ST.

Exploratory Testing – a situational practice

ET is not a replacement to the traditional approach but is a part of the agile approach. An agile approach is a fully elaborated testing philosophy that incorporates principles such as short iterations, continuous integration, extensive communication, ready acceptance of changing requirements, close involvement of business users and concentration on developing software rather than documenting it.

When solving a puzzle, one has to think about all the entry and exit criteria's and has to explore all the possible way out to solve it. Solving a puzzle can be the best and easiest example to tell about the ET as in the start of solving a puzzle you don't have any idea about it but then when you start exploring it, you get more and more insights of it. Each glance and a thought is a test case (“Is this can be the first step to solve the puzzle, if not, then if the author do this then what will happen and if he does it that way then what will happen?”). Now, if a puzzle is a mathematical puzzle then you might start with some calculations, but if it is a jigsaw puzzle then you might start it with exploring the different parts of the image and gathering them. As the type of the puzzle changes, the approach towards solving the puzzle changes, similarly, exploratory testing changes as the type of the application or the motive of the testing changes.

Making Exploratory Testing Manageable

ET is a well planned and a structured approach and by no means synonymous with ‘unstructured’ and ‘unplanned’, but still some test managers complain that it is not an adequately transparent or a manageable process. They found it difficult to monitor their teams’ activities or to accurately gauge the status of the test object. In response, test managers have developed techniques to gain better control of the ET process and explain it more clearly to project stakeholders³.

The resulting variations on ET ranged from informal ‘freestyle ET’, where the tester produces only bug reports, to the more formal ‘chartered ET’ where the tester works according to a specific test assignment, but with no designated procedure. In another variation, the test effort is divided into manageable time segments.

Confronted with the limitations of freestyle ET, greater structure and manageability were sought, resulting in methods like those below:

- *Sessions*: 90-minute time boxes for ET
- *Charters*: A clear mission for the session describing what to test, how to test, what bugs to look for, what risks are involved, and what documents to examine
- *Session sheets*: Reviewable results of a session, including notes, bugs, issues, and basic metrics such as time spent on set-up, test execution, and bug reporting
- *Session logs*: Hansel and Gretel-like ‘breadcrumb trails’ used during test execution
- *Debriefings*: Meetings with the test manager
- *Dashboards* for reporting purposes

When to Apply Exploratory Testing

The main question to the mind of a tester and management comes is when one should go for the ET, when can it be ideally useful. While the following list is not exhaustive, it suggests several ideas as to where ET can best be incorporated into the overall test approach:

- *It’s an agile project*, there is no scope of the documentation and so ET technique is followed.
- *A new tester enters the team*: ET can make the learning phase an active testing and exploration experience. A new tester with a little help of the other senior tester in his team or a developer can learn the new application.
- *A quick assessment of the application is needed*: ET offers fast insight into product quality in the short term when there is no time for test preparation.
- *Validation for the work of another tester is required*: ET lets you explore the feature that the other tester has tested.
- *There is no test basis*: ET is useful when there is no documentation or other sources that can define clear, expected results for the tests.
- *You want to isolate and investigate* a particular defect.
- *You want to determine the status of a particular risk* in order to evaluate the need for ST in that area.
- *It’s an early iteration* in which the product is not stable enough for ST.

- *It's a beta test*, where users are invited to provide early feedback on a prototype or a preliminary test version.
- *New information comes to light during the execution of scripted tests*: The new information might suggest another test strategy that would warrant switching to an exploratory mode.
- *You want to augment ST to diversify testing*: The combination of ST and ET is appropriate for testing features with a high risk and/or priority profile.

Paired Testing – an ET concept

Now-a-days paired programming concept is getting more and more common in the IT industry because of its benefits like design quality, overcoming difficult problems and better time management etc. Similarly a new concept of Paired Testing is getting famous now among the testers who are working in an agile environment. This concept of testing is similar to paired programming approach. In this approach the person doing the testing is called the driver while the other suggests ideas or tests, pays attention and takes notes, listens, asks questions, grabs reference material, observes or reviews etc.

Paired testing can be really helpful when a new team member has joined the team and the project is in the middle of the sprints⁴. A team member who is mature enough and have a good knowledge about the application can then be taken as a driver making the new team member as an observer so that he can learn the functionality.

Paired testing can be started with defining of a charter which include what to test, what tools to use, what testing tactics to use, what risks are involved, what bugs to look for, what documents to examine, what outputs are desired, etc. and can be ended once the reports are generated which can have the defect numbers that were logged while doing the testing.

Benefits of Paired Testing

- Idea generation, as the testers are working in a pair so one can really come up with new ideas while the other is implementing the same.
- Switching of roles, if one of the tester is tired of doing the testing so they can switch the roles and then start doing the testing again.
- Helps the tester stay on task.
- Restricts the interruption, when 2 testers are working together it limits others to interrupt them which increases the productivity.
- Defect reproducibility, bugs can be easily reproducible as while doing the exploratory testing alone a tester tends to forget the steps due to which the defect occurred.
- Good learning platform, paired testing gives a good learning platform if a new tester has joined the team.

ET in Action

When the author joined his current organisation, he was the only tester over there and the application was in the middle of the iterations. Since the company works on the agile methodology and so he had to be on his toes as soon as possible or you can say to be productive from the day 1 and that's a great challenge. As soon as the author entered the room where the entire team of his project was sitting, he was expected to start doing the testing for the application. Now when he asked them about the functional documents, requirement specifications everyone in the room just laughed at him and said we are 'agile', which meant no documentation and no test cases. In the authors previous organization they (team of testers) were getting the functional documents/requirement specifications from which they were writing the test cases and then were testing the application.

Now, the author had to kick start testing the application and simultaneously learning it. He started looking at the application, going through some basic functionality and hence exploring it. Being the only tester in the team he had the entire responsibility of sending a bug free application. With this rather clear charter in mind, he set himself to test. Applying one of the simplest heuristics of exploring, he chose to begin by walking through the menus of the application, trying each one. While doing so, he started creating an outline of the primary functions of the product. This would become the basis for reporting what he did and did not test.

A basic strategy of ET is to have a general plan of attack, but you have to allow yourself to deviate from it for short periods of time. And so the author took all the menus one by one and started exploring them to learn the application and started asking about the expected behaviour of the application in the certain situations from the scrum master⁵ or the product owner⁶. Within no time he had the control over the certain areas of the application and thus by the previous experience and using ET, he learned the application and started logging the defects in the application.

By the above example of the approach that the author followed in testing we can easily say that the ET is a well disciplined and a purposeful testing. A report can be easily be generated for the areas that the author covered for the testing and the defects logged by him while doing the testing. It was also quite repeatable, or at least as repeatable as most scripted tests, because at all times the author followed a coherent idea of what he was trying to test and how he was trying to test it. The fact that those ideas occurred to him on the fly, rather than being fed to him from a document and so he was able to un-turn some hidden defects.

Can Exploratory Testing be Automated?

Regression testing is often better suited than ad hoc testing to automated methods. However, automated testing methods are available that provide random variation in how tests are executed. Yet, a key strength of ad hoc testing is the ability of the tester to do unexpected operations, and then to make a value judgement about the correctness of the results. This last aspect of exploratory testing, verification of the results is exceedingly difficult to automate.

Partial automation is another way in which automated testing tools and methods can be used to augment ad hoc testing in particular, and exploratory testing in general. Partial automation implies that some of the test is automated, and some of it will rely upon a human. There are yet other ways in which automation tools can help the manual exploratory tester. A good recording tool may help in capturing the steps in a complex series of exploratory operations. This is especially important if

you uncover a defect, because it will bring you that much closer to identifying the steps to reproduce it.

Exploratory Testing: Pro and Con

Every approach comes with some advantages and with some disadvantages and so comes, ET, even though it is clearly a valuable component of a modern test approach. Among its many important advantages:

It encourages creativity.

- Its greater efficiency enables the tester to find more bugs in a shorter time period and make a faster assessment.
- It is adaptable and flexible.
- It increases the chance of finding new and otherwise undetectable bugs.
- It allows more time for testing interesting and complex cases.
- It demonstrates an application's ease of use.
- It's more fun than ST.

At the same time, ET also has disadvantages that must be taken into account:

- Its limited manageability makes it difficult to coordinate.
- It provides only limited project intelligence, with little insight into risks, test coverage, or test depth.
- It offers limited test reusability or reproducibility of failures.
- It depends heavily on the testing skills and domain knowledge of the tester.
- When combined with ST, it involves a risk of redundant tests.
- It provides low accountability, making it unsuitable for a regulatory compliance context.
- It does not provide absolute assurance that the most important bugs have been found.
- It is inappropriate for security testing, performance testing, or some other highly specialized test types.
- It is not suitable for tests that are complex to set up or those with a slow feedback loop, such as batch programs running at night.
- It can start only when the test object is available—that is, situated directly on the critical path of the project.

Having some advantages and disadvantages with ET, as a tester/ a team lead, the author will combine ET with the ST so as to have an effective testing approach.

Conclusion

Skilled exploratory testing can be a powerful way of thinking about testing. ET is now recognized as a valuable technique in modern test approaches, representing an evolution towards more complete and efficient testing. The most effective test strategies are a combination of ST and ET.

While ET is a useful tool, it does not provide a ‘free pass’ to unstructured testing practices. Some activities should never have an exploratory character, and should always be formal and structured in nature. Test strategy, test planning, and all other test management activities must be extremely well structured, because they serve as the basis for delivering high-quality project intelligence to management. And as projects become more agile and exploratory, these activities—and hence the test manager’s job—become even more crucial.

If testers want to win respect and credibility as they perform exploratory testing, they have to be able to describe their work to a stakeholder’s satisfaction.

References

- [1] James Bach, 2003, "Exploratory Testing Explained,"
- [2] James Bach, "What is Exploratory Testing?"
- [3] Andy Tinkham & Cem Kaner, "Exploring Exploratory Testing"
- [4] Cem Kaner, 2004, "Examples of Exploratory Testing"
- [5] James Bach, 2003, "Scripted versus Exploratory Testing," Eurostar 2003
- [6] James Bach, 2003, "Inside the mind of an Exploratory Tester"
- [7] James Bach, 2001, "Where does Exploratory Testing fit?"
- [8] James Bach, 2001, "Exploratory Testing and the Planning Myth"
- [9] Lee Copeland, 2001, "Exploratory Planning"
- [10] Elisabeth Hendrickson, 2004, "Agility for Testers"
- [11] Rex Black, 2005, "The importance of the Right Technique"
- [12] William Rollinson, 2005, "Face-Off: Structured vs. Exploratory Testing and Error"

Author's Biography

The author, Aman Arora, is an IT Professional and has more than 7 years of overall experience in software development and testing. He has worked with top notch companies like CSC and Adobe Systems, Noida and has worked on different platforms like MAC – 10.4, iMAC, Mainframes (MVS) and Windows platforms. He is currently seeking challenging position in the area of Software Testing as the Team Lead Testing in Xebia India.

The author actively participates in the conferences and one of his papers titled '*Agile Automation Testing*' has already been published in Test 2008 conference. The link to the paper is "<http://test2008.in/Papers%20to%20be%20published/Aman%20Arora/Agile%20Automation%20Testing.pdf>". His other published works include the blogs on the "<http://blog.xebia.com>" website.

Appendix

¹ eXtreme Programming (XP): XP is a software engineering methodology which is intended to improve software quality and responsiveness to changing customer requirements. It is a type of agile development. (*Wikipedia*)

² Scrum: **Scrum** is an iterative incremental framework for managing complex work (such as new product development) commonly used with agile software development. (*Wikipedia*)

³ Stakeholders: customer / client.

⁴ Sprint: Iterations in the software life cycle are referred as sprint.

⁵ Scrum master: is the one who maintains the processes.

⁶ Product owner: is the one who represents the stakeholders.