**Test Better by Exploring**

**Harnessing Human Skills and Knowledge**

**Authors:**

Juha Itkonen - Aalto University

Mika V. Mäntylä - University of Oulu

Casper Lassenius - Aalto University

**Abstract:**

Exploration can directly apply human knowledge and learning to software testing, revealing more relevant bugs earlier. Academics and practitioners should exploit exploration’s strengths in testing and focus on reporting existing practices and bene­fits in different academic and industrial contexts.

**Introduction:**

It focuses on the concept that why do users keep detecting bugs despite the vendors investment in testing, we suggest that developers should focus more on exploration in software testing to better address users needs. In real-world the contribution of human testers is highly relevant in software development because most new bugs are found by human testing the software especially in the case of interactive systems.

Human testers have many advantages compared to machines in many aspects like including knowledge, creativity, intelligence, ability to learn and adapt to new situations and the ability to recognize problems.

**Body of paper:**

**6.1 Exploring as a Facet of Testing:**

Exploration occurs when you don’t know what route to traverse and discoveries to make, which is also true in software testing which aims to discover the tested systems unknown functionality. Software testing involves various degrees of exploring one end of the spectrum is traditional confirmatory testing. In this someone first designs and documents all the test cases then the tester follows the test cases using execution. The expected results are also documented as the hypothesis of how the system should work and defect detection is based on those documented results.

The context driven school of testing has advocated exploratory testing (ET). ET has been defined as a designed style of software testing that emphasizes the personal freedom and responsibility that each individual tester to continuously optimize the value of her work by treating the test related learning , test result, test execution as mutually supportive activities that run continuously in parallel throughout the project.

Exploratory testers use documentation as much or as little as possible. They have a particular goal but no idea about how to accomplish it. Testers can just follow a list of features to be tested in the documentation limiting the exploration at the higher level. ET doesn’t restrict the testers freedom to choose the best ways to test within the limits.

**6.2 Applying Knowledge when Exploring:**

Testerspersonal knowledge and experience may also affect the testing results. Testers apply their knowledge to different tasks indifferent purposes much of the applied knowledge in testing is tacit. In confirmatory testing test design, execution, reporting are all different phases. Confirmatory testing is inefficient due to multiple knowledge transfers. Test design which is highly creative and exploratory is a part of confirmatory testing

ET uses people with required knowledge to do actual testing but some argue that the people with right knowledge does not find time to perform testing. However getting their tacit knowledge to be documented is also expensive or impossible.

**6.3 Exploratory Testing Practices:**

Commonly proposed solution to managing ET is session based test management which enables planning, tracking and reporting without compromising ET’s flexibility. ET employs some traditional test design techniques like boundary analysis and testing.

ET harnesses human capabilities such as tacit domain knowledge to efficiently recognize unanticipated problems in the tested system. Exploration provides benefits for testing system with rich user interactions, a complex application domain or a social context that requires human expertise to understand.

**6.3.1 Organising Level:**

Session Based Test Management is the most commonly used solution for Exploratory Testing. SBTM enables planning, tracking by not sacrificing ET’s flexibility.

**6.3.2 Session Level:**

During a test session testers benefit a lot from exploration testing. The testing activity can be purely exploratory or follow some selected techniques. The central characteristics of exploration is that testers are encouraged to explore anything that seems interesting, suspicious or valuable to the testing task.

**6.3.3 Technique Level:**

ET applies some traditional design techniques which are more popular such as boundary value analysis and complete combination testing. Selection of the testing technique is based on the expertise. Confirmatory testing focuses on the correctness of the documented expected outcome of each test case.

**Conclusion/Future Work:**

* An experiment suggested that Exploratory Testing (ET) has a superior efficiency than test-case-based testing in defect detection.
* According to the practitioners session based test management brings more accountability to Exploration Testing (ET).
* Initially ET was applied by the developers to tackle the combinatorial complexity by applying experience-based test selection, later ET worked well with weak or frequently changing specifications and enabled quick feedback from testers to developers and finally ET facilitated learning about a product’s new features.
* ET has its importance in detecting non-functional aspects such as attractiveness and usability.
* A hybrid approach has been proposed to combine ET and scripted testing, team ET testing was proposed by other researchers.

**References:**

* J.A. Whittaker, Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design, Addison-Wesley Professional, 2009.
* J. Bach, “A Case against Test Cases,” blog, 15 Oct. 2007; www.quardev .com/blog/a case against test cases.
* J. Bach, “Exploratory Testing,” The Testing Practitioner, 2nd ed., E. van Veenendal, ed., UTN, 2004, pp. 253–265.
* C. Kaner, “De­ ning Exploratory Testing,” blog, 14 July 2008; http:// kaner.com/?p=46.
* J. Itkonen, M.V. Mäntylä, and C. Lassenius, “The Role of the Tester’s Knowledge in Exploratory Software Testing,” IEEE Trans. Software Eng., vol. 39, no. 5, 2013, pp. 707–724.