A Strategic Review of Exploratory Testing Techniques.

Syed Shujauddin Sameer[#1]

[#] *Lecturer,Department of Computer Science ,King Khaled University,Abha Saudi Arabia*

***Abstract*— Building quality software is always an important part of SDLC stages. Testing builds a confidence to examine the quality of software developed. Exploratory testing requires the use of knowledge of the testers. There is a requirement to reduce the involved in the testing .we provide an overview of exploratory testing with the basic knowledge to the introduction of heuristics for efficient testing.**

***Keywords*— Testing, Exploratory Testing, Knowledge, Heuristics .**

I. INTRODUCTION

Software development gives rise to software bugs .Bugs have always been a part of software development .They keep on their interference with every new software. Bug prevention techniques are developer oriented. Writing code reviews, running tool analysis ,unit testing are some of the techniques. Developers are good at writing the bug detection programs but if they were why would the software develop bugs. Every software needs some input and output data .The accumulation of data over a large amount of time leads to the development of bugs. Humans can test manually the software by themselves. Test-case design techniques are for ensuring coverage, and finding of different types of error [2], [3],

Event though manual testing finds many of the software bugs when compared with automated testing [13].Many manual testers are guided by scripts, written in advance, that guide input selection and dictate how the software's results are to be checked for correctness. When the scripts are removed entirely, the process is called exploratory testing. The testers are free to interact with the application in whatever way they want to find the flaws in the system. A Documentation is often required for the Testers using exploratory methods. Test results, test cases, and test documentation are generated as tests are being performed instead of being documented ahead of time in a test plan. Exploratory testing is especially suited to modern web application development in agile environment.

.

II. EXPLORATORY SOFTWARE TESTING

The Exploratory testing risk's wasting a great deal of time wandering around an application looking for things to test and trying to find bugs. This may result in a lot of time being spent finding no good results. To understand avoiding such situation we need to know Exploratory testing in small, large and hybrid environment

A. Exploratory Testing in Small Domain

Exploratory testing is an experience-based testing approach that differs from the highly document-driven test-case-based testing approach [1].Exploratory testing can be defined as a simultaneous learning approach that includes test design, and test execution. The tests are not defined in advance in an established test plan, but are dynamically designed, executed, and modified [6]. The term exploratory testing was introduced by Kaner et al. [4]. The Exploratory Testing approach is in software testing books since the 1970s [2], but mostly referred to as an "ad hoc approach" or error detection without any description of how to perform it. Exploratory Testing in small can be broken into parts like inputs, state, code paths, user data, and execution environment. An input is a stimulus generated from an environment that causes the application to respond in some manner. Accepting an arbitrarily large number of inputs is usually done in many applications. These may vary from tens to hundreds. Software testers must decide in selecting one input as a better test than another input .It can often team up on software to cause it to fail. The first way that testers can decide over this problem is based on what the developers think constitutes an illegal input. Developers have three basic mechanisms to define error handlers: input filters, input checks, and exceptions. A State refers to the environment of the system. We need to understand the way in which the input travels from one stage to the other stage in the system. This will help us to find the error if caused at a certain point in the system. This can simply be understood as the state may change in a mobile system as to whether the operator is in within the range, or the device does not receive the calls. A code path is a sequence of code statements beginning with invocation of the software. Branching condition like If then else, switch statements can also cause a tester to be mislead in understanding the fault.

Whenever software is expected to interact with large data stores, such as a database or complex set of user files, testers have the unenviable task of trying to replicate those data stores in the test lab. Now a days the data is increasing in large amounts and stores in large data bases like data warehouses. Real user databases evolve over months and years as data is added and modified, and they can grow very large. We may also need to maintain the privacy for the sensitive data, as most of the data may turn out to be personal data which should not be relieved to the outside world .Finally the environment also holds a major part for the system in which the application is developed.

B. Exploratory Testing in Larger Domain

Feature interaction, data flows, and choosing paths are some of the larger decisions that testers are to make through the user interface .Here we need to gain an understanding about how the application works, how bugs can be detected, and how to force the software to express its capabilities. Metaphors can be a powerful guide for software testers. A metaphor will act as a guide to help testers choose the right input, data, states, code paths, and environment settings to get the most out of their testing time and budget. The name of the metaphors itself helps us understand the reality behind it which is very useful in testing.

Some examples include tourist metaphor is just as a tourist travels around different places exploring, entering in mysterious lanes finding new routes will help them a lot. This requires a lot of planning, Just like it Exploratory testing also should be applied as if we do not know where to end .We keep on testing the things as they turn up till we reach our solution .Like a tourist who finds new routes we also end up finding new faults in the system. Some of the tours may be through business districts, historical districts, tourist districts, entertainment districts, hotel districts , Landmark tour, intellectual tour[14] . Tours give a structure to testing and help guide testers to more interesting and relevant scenarios than they would ordinarily come up with using only freestyle testing. A goal is required to be set for the testers, by giving a goal to testers, the tours help guide them through interesting usage paths that tend to be more sophisticated than traditional feature-oriented testing where an individual tester will try to test a single feature in isolation.

A tester can also create his own strategies to be applied to the application to find the faults. The inputs should be handled very carefully not giving rise to new bugs.

C. Hybrid Exploratory Testing

Here we try and explore some traditional scenarios to be induced to the strict working strategy of testing. Based on the above two environments of testing in small and large we may include describing the requirements of domain ,demonstrate how the feature works ,demonstrating integration with new data, demonstrating things that could go wrong. Scenario operators are constructs that operate on steps within a scenario to inject variation into the scenario. When we apply a scenario operator to an existing scenario, we get a new scenario that we call a derived scenario. The tester here can increase the number of records to be tested, repeat some of the records many number of times, make use of additional inputs ,optional steps can be removed, finding more than one way to do the step with the optimal one .Few tours that can be included may be landmark tour like selecting a specific feature landmarks out of the scenario. Now randomize the order of the landmarks so that it is different than the original scenario. Run some tests with the new order of landmark features and repeat this process as often as you think is necessary. Scenarios can represent an excellent starting point for exploration, and exploration can add valuable variation to otherwise limited scenarios. When I implement such strategy on my students they all end up wandering as a new tourist and really finding the flaws. But they stick up to the type of data in the input box, Length of the data, size of the data.

D. Heuristic Knowledge

One way of applying knowledge in software testing is to use it as a test oracle. A test oracle is a concept referring to a method used to distinguish between a correct and an incorrect result during software testing [3], [6], [7], [8].Recognizing failure is one of the most crucial activities in testing, and the existence of a test oracle is recognized as a fundamental requirement in all kinds of testing [7], [8],[9], [10].

The main idea of Exploratory testing is that a tester uses any available source of information available at the context of the system. We need to discuss the different types of knowledge and search for their application in different parts of the system. Faults have to be identified within the system and rectification should be carried on. Searching a strong Hypothesis is essential in Exploratory Testing . Analysis revealed three types of knowledge that testers utilized to recognize failures in the observed sessions.[1]. There is also a question of application of knowledge to exploratory testing. Based on analysis, there are two main approaches. First, the most common way of applying knowledge was using knowledge as a test oracle . A predefined set of assumptions of results .In real testing, the outcome is predicted and documented before the test is run [3]. In practice, requirements, specifications, and test cases, are not more perfect in terms of accuracy. The test results are evaluated in practice using the human oracle [11], [12].

Applying knowledge as an oracle differs clearly from the traditional test-case based paradigm in which the expected result is specified prior to test execution. Knowledge can be categorized into domain knowledge, system knowledge, and generic software engineering knowledge.

These issues can still be enhanced by the application of heuristics .Heuristics can be defined as the process that requires a number of testers to find the usability problems in various scenario's of interfaces and also most important require less amount of time .It is one of most important mechanism and used in both industry and academics .The testers should become familiar with the environment first. This will help them to understand the purpose of their work. The domain should be made familiar to the testers. The testers should add new heuristics as they get deep in their effort and most important they should validate them whenever required. Usability testing in makes use of end user representatives who perform a set of carefully designed task. The heuristical approach finds more usablity problems when compared to other testing strategies .The testers are free to use their knowledge in many scenarios and find new ways to explore faults.It is the responsibility of the tester to ensure that no path is lead unexplored. Heuristic approach would help him in a lot of extent.

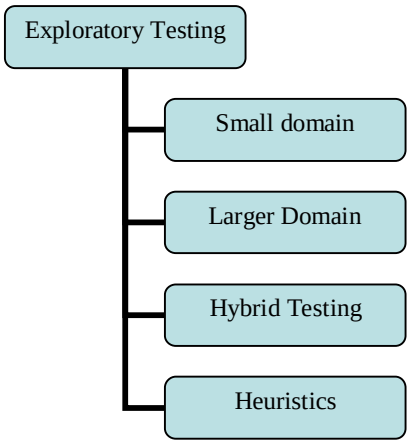

Fig 1.Classification of Exploratory Testing.

The above Figure aids us in understanding Exploratory Testing in various domains.

III.CONCLUSION

Exploratory testing is an essential approach from the low levels to the high levels. It is efficient way to understand the domain knowledge. Software failures aid in developing an efficient product at the end. Exploratory testing has many more aspects to be explored .Experience and knowledge play a very important role in it. Here we have focused on some basic aspects of Exploratory testing with a application of knowledge ,failure recognition .Application of heuristical approach should be at a greater depth to analyse exploratory testing with results. Further this study will help the researchers to get an overview of the exploratory testing techniques helping them to perform better analysis of software.

REFERENCES

[1] *The Role of the Tester's Knowledge in Exploratory Software Testing* Juha Itkonen,Member, IEEE,MikaV.Ma¨ntyla¨, Member, IEEE, and Casper Lassenius,Member, IEEE. IEEE Trans on Software Engineering, vol. 39, No. 5, May 2013

[2] G.J. Myers,*The Art of Software Testing.*John Wiley & Sons, 1979.

[3] B. Beizer,*Software Testing Techniques.*Van Nostrand Reinhold,1990

[4] C. Kaner, J. Falk, and H.Q. Nguyen,*Testing Computer Software.*John Wiley & Sons, Inc., 1999

[5] J.B. Goodenough and S.L. Gerhart, *"Toward a Theory of Test Data Selection,"*IEEE Trans. Software Eng., vol.1, no. 2, pp. 156-173,Mar 1975

[6] A. Abran, J.W. Moore, P. Bourque, R. Dupuis, and L.L. Tripp,*Guide to the Software Engineering Body of Knowledge.*IEEE CS, 2004.

[7] W. Howden, "*Theoretical and Empirical Studies of Program Testing,*"IEEE Trans. Software Eng.,vol. 4, no. 4, pp. 293-298, July 1978.

[8] L. Baresi and M. Young, "*Test Oracles,*" Technical Report CISTR-01-02, Dept. of Computer and Information Science, Univ. of Oregon, Eugene, Aug. 2001

[9] J.A. Whittaker, "*What Is Software Testing? and Why Is It So Hard*?"IEEE Software,vol. 17, no. 1, pp. 70-79, Jan./Feb. 2000.

[10] A. Memon, I. Banerjee, and A. Nagarajan, "*What Test Oracle Should I Use for Effective GUI Testing*?"Proc. 18th Int'l Conf. Automated Software Eng.,pp. 164-173, 2003.

[11] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville,*'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company,"*Proc. Int'l Conf. Software Eng.,pp. 602-611, 2007.

[12] J. Rooksby, M. Rouncefield, and I. Sommerville, "*Testing in the Wild: The Social and Organisational Dimensions of Real WorldPractice,"*Computer Supported Cooperative Work,vol. 18, nos. 5/6, pp. 559-580, 2009.

[13] Defect Detection Efficiency: Test Case Based vs. Ex-p loratory Testing Juha Itkonen, Mika V. Mäntylä, and Casper Lassenius Proceedings of International Symposium on Empirical Soft-ware Engineering and Measurement, 2007, pp. 61–70.

[14] Exploratory Software Testing ,James A.Whittaker. 1965.