Exploratory Testing: An Overview

Rashmi N.
Dayananda Sagar College of Engineering
S.M. Hills, K.S.Layout
Bangalore 560078

Suma V.
Dayananda Sagar College of Engineering
S.M. Hills, K.S.Layout
Bangalore 560078

ABSTRACT

There exist various approaches in software testing to test software under development. Exploratory Testing is one such important approach where no predefined test cases are used to test the software. It is proven by researchers that exploratory testing is equally effective in detecting defects as the traditional testing where predefined test cases are used to test the software. Hence, this paper brings in deeper insight to gain knowledge of exploratory testing approach. Accordingly, this paper put forth an understanding of the characteristics, benefits, advantages and challenges, techniques, tools and recent advances in exploratory testing. This awareness acts as a travel light for further areas of research and progress in exploratory testing. Further, thus gained understanding enables the test team to formulate strategies towards better modes of developing customer satisfied software products.

Keywords

High Quality Software, Software Engineering, Software Testing, Exploratory Testing

1. INTRODUCTION

Human life today is highly unimaginable without software which has found its use in almost all the fields such as business, medical, education, military and telecommunications. This dependency of human life on software demands IT organizations to develop high quality software [1]. High quality software is defect free, produces predictable results and delivered within time and cost constraints. Also, high quality software is manageable, maintainable, dependable, understandable and efficient. The important factors which affect the quality of a product are quality of the process and the quality of the people which can be represented mathematically as shown below [2].

$$\text{Software Product Quality} = \sum_{i=1}^{n} f(\text{People Quality} + \text{Process Quality}) \quad (1)$$

Where, i = 1 requirement phase, 2 = …, 3= …. n = maintenance phase of software development process

Hence, delivering such high quality software that possesses all the above mentioned characteristics is a challenge in reality faced by IT companies. As a solution, IT industries use several software engineering processes, methods, techniques and standards. One such process is Software Verification and Validation, which detects defects in non-executable and executable software under development and ensures that software which is delivered to the customer, is almost defect free. A defect is any inaccuracy, inadequacy, or undesired behaviour that occurs either in the deliverable or in the product. Software defects have an inherent nature of dwelling, propagating and magnifying with the passage of development. Fixing of these defects at later stages of development becomes costly and time consuming. Thus, it is very much necessary to detect and remove these defects right at the point they are injected [2]. Software Testing, which is a quality control activity, plays a significant role in identifying the presence of defects in executable software. There exists various software testing techniques which can be categorized into Scripted and Unscripted Testing [3]. Exploratory Testing is an unscripted testing where unlike scripted testing, there are no predefined test cases to be executed and is a simultaneous learning, design and execution of test cases. This paper intends to provide an overview of Exploratory Testing in terms of its characteristics, benefits, advantages and drawbacks. The paper also discusses the techniques, models, metrics and tools supporting Exploratory Testing.

2. EXPLORATORY TESTING

Exploratory Testing, introduced by Cem Kaner in 1960’s is a recognized testing approach but has commonly been referred to as ad hoc testing or error guessing [3]. It differs from the conventional test case based testing in that the tests are not based on the predefined test cases. Instead it is a creative, experience based approach in which test design, execution and learning are simultaneous activities and the results of the tests executed are in turn applied for designing further tests. Therefore, exploratory testing is deemed to be a testing activity which involves simultaneous learning in association with test design generation and conducting test execution [3].

There are several scenarios which are considered to be best suitable for Exploratory Testing. For an instance, the following scenario explains a situation where the tester is assigned the task of testing a photo editing program in four hours. The tester's aim here is to assess the program against the standards of the Microsoft Windows Compatibility Certification Program and to report any existing compatibility violations. In order to test this, the tester sets the memory slide bar to 5% and performs memory intensive functions. The tester sets the image size to 100 inches square which is a big canvass, fills the canvass with a color and also tries to add the special graphical effect such as ripple effect to the image. When tester was doing this he immediately gets an error message informing that there is insufficient memory to perform the operation. So this satisfies the desired stability of the program. Next, the tester goes ahead with applying the rest of the graphical effects to the image and to his surprise the program would crank away for five minutes eventually giving an error message: " Error -32: Sorry, this error is fatal" and the application crashed [3].

Ad hoc testing, a special case of Exploratory Testing has a sufficiently detailed test notes using which the tests can be rerun by reading them [4]. Exploratory Testing emphasizes learning and adaptability which comprises of four important activities such as learning, design, execution and interpretation activities [5].

Learning, guides what to test, how to test and how to recognize a problem. A tester learns about competitive products, and the history of the product. Also, the tester inspects the product under test, questions, reviews written sources, and experiments with the tools [5].

Design is an activity that enables the testers to create, and construct tests according to the plan. Examples of design activities include mapping of test techniques to test ideas, tools to test techniques, staff skills to tools/techniques, development of supporting test data, development of supporting oracle and so on [5].

Execution, involves test execution and collection of results. Execution is either manual or automated. Configuration of the product under test, pair testing, creating and debugging automated tests are examples of execution activities [5].

Interpretation makes the tester learn from the program in terms of the product and the mode of testing the product. Interpretation activities include determining the pass and fail criteria of test [5].

Robust Characteristics of Exploratory Testing (ET)

Exploratory Testing is popular among testers because of its capability to perform testing in absence of pre-defined test cases and is lead by the results of previously performed tests. The focus in exploratory testing is on finding defects by exploration, thus, several unexpected defects can be predicted and detected which otherwise escapes conventional testing approaches.

The important characteristics of exploratory testing include

1. Test cases are not defined in advance. Instead, exploratory testing is an exploration with a universal mission without step-by-step instructions on how to accomplish the mission.
2. ET is guided by the results of previously performed tests and the gained knowledge from them. An exploratory tester uses any available information such as requirements documents, a user's manual, or even a marketing brochure for testing purposes.
3. ET is highly focused on detecting defects by exploration as opposed to developing test cases for later use.
4. Exploratory testing is simultaneously learning the system under test, designing the test, and execution of the test.
5. Effectiveness of exploratory testing depends on tester's knowledge, skills, and experience [6].

Benefits of Exploratory Testing
The vigorous nature of exploratory testing is useful in various ways during software development process. Some of the benefits of exploratory testing are

1. It increases the defect detection efficiency in terms of defect count, defect severity levels and number of false defect reports.
2. Improves the skills of the tester through simultaneous learning since the tester can learn about the behavior and the failure of the system under test.
3. Pre-defined test cases are not required in exploratory testing which hence leads to reduced documentation.
4. Exploratory testing does not require detailed requirements or specification document.
5. A rapid flow of feedback from testing to developers and testers without the need to hold to organizational obstacles [6].

   Despite of benefits, yet, ET is suitable in few types of applications which are mentioned below.

Areas well suited for Exploratory Testing
Exploratory Testing is applicable in the following situations.

- When a rapid feedback is needed on a new feature or a product.
- When a product needs to be learnt quickly.
- When the tests on the product needs to be expanded and the product is already tested using pre-defined test cases.
- When an important defect needs to be detected in the shortest period of time.
- When a particular needs to be investigated and isolated.
- When the status of a particular risk needs to be investigated.

In addition to the above situations exploratory testing also fits in the following situations.

- To improve scripted tests.
- To interpret imprecise test instructions.
- To perform product analysis and test planning.
- To write new test scripts.
- To perform regression testing based on old bug reports[7].

Exploratory Testing is basically an approach where any testing technique for e.g. scenario-based testing, model-based testing can be performed in an exploratory manner [7].

Structure of ET
Exploratory Testing has a definite structure with external and internal dimensions. External structure consists of elements such as time, tester, product, mission and reporting. A ***tester over a period of time*** interacts with a ***product*** to satisfy a testing ***mission*** and ***reporting*** results. During this process the tester aligns towards the testing mission, the tester imagines a series of questions about the product, designs tests and executes the tests to get the answers for those questions. The tester adjusts the tests and continues exploring if the answers are not satisfactory. The status and the results of the tests are reported by the tester anytime. Internal structure of exploratory testing exists inside the mind of the tester. Therefore, an exploratory tester possesses the following characteristics.

Test Design: Being a test designer an exploratory tester designs the test which systematically explores the product.

Careful Observation: *Being a* cautious observer an exploratory tester observes *anything* unusual or mysterious and must be able to distinguish between observation and inference.

Critical Thinking: Exploratory testers have the ability to assess and describe their logic that helps in reporting the status of the testing.

Diverse Ideas: Exploratory testers create new by making use of heuristics such as guidelines, generic checklists, mnemonics, or rules of thumb.

Rich Resources: A list of tools, information sources and test data are prepared by exploratory testers so that they can be applied appropriately during testing [7].

Exploratory Testing Techniques
Important exploratory testing techniques are

Freestyle Exploratory Testing
This technique does not include specific charters. The tester freely explores the product by learning, designing and executing the tests. Defect Reports are the only official result obtained from this technique. Freestyle exploratory testing is managed in two ways namely by delegation and by participation. In exploratory testing managed by delegation, the test lead specifies the charters and the testers continue designing and executing the tests to achieve the charters, and report back. The test reports in this technique may be written or oral. In Exploratory Testing managed by participation the test lead also performs testing along with other testers. This participation eliminates potential confusions in the testing team. Team exploratory testing is a technique where the collective effort of the people in detecting the defects leads to better ideas than if they worked individually [7].

Session Based Test Management
A session being the basic testing work unit, is an uninterrupted block of reviewble, chartered test effort. By "Uninterrupted" indicates no significant interruptions such as no telephone calls, emails, chatting, meetings. "Reviewable" indicates producing a session report that is reviewed by test manager thus helping the manager in taking decisions. The meaning of "chartered" is having a mission for testing activity. A session normally lasts for 90 minutes, but there might be short and long sessions which lasts for 45 minutes to 2 hours.

Each session in turn consists of three kinds of tasks namely, test design and execution, bug investigation and reporting and session setup. These are together called "TBS" metrics. Test design and execution deals with scanning the product and looking for problems. Bug investigation and reporting is the process of detecting and the reporting of a bug. Certain activities such as configuring equipment, studying manuals, locating materials or writing a session report are required for setting up a session. Session based test management is also accompanied "opportunity" testing. Opportunity testing is any testing which does not match with the charter of testing. In addition to task breakdown metrics a session sheet also contains three important parts such as bugs, issues and notes. Bugs determine the quality of the product, issues are the questions or problems related to the testing process, notes consists of test case ideas, lists of functions, risks etc. At the end of each session, debriefing of the session takes place. Debriefing gives an idea of the progress done in a test session.

A session report consists of the following sections

1. **Session Charter(areas to be tested)**
2. **Tester name**
3. **Date and time started**
4. **Task breakdown(TBS metrics)**
5. **Data files**
6. **Test notes**
7. **Issues**
8. **Bugs**

These reports after the session are stored with other reports in the database and are then scanned by a tool, which breaks them into the basic elements, normalizes them and summarizes them into tables and metrics. The scanning tool makes about 80 syntax and consistency checks on each sheet. Output of the scanner is a group of text tables. Each text table is in a delimited format suitable for importing to MS Excel for analysis purposes. The text tables are as follows

- Test Notes (test notes sections by session ID)Bugs(bug records, by bug ID and session ID)
- Issues(issue records, by issue ID and session ID)
- Charters(charter statements and area keywords, by session ID)
- Date Files(data file names by session ID)
- Session Breakdowns(Session metrics, by session ID)
- Coverage Breakdowns(session metrics, by area keywords)
- Tester Breakdowns(session metrics, by tester name)
- Day Breakdowns(session metrics, by day)
- ToDo sessions(Incomplete session sheets) [7]

Some of the metrics obtained from session based test management method are

- Number of sessions completed.
- Number of problems found
- Function areas covered
- Percentage of session time spent setting up for testing
- percentage of session time spent on testing
- percentage of time spent investigating problems
- velocity of charter
- average session execution Progress of testing can then be measured using these metrics which helps the management to take decisions.

A sample session sheet can be referenced in [8]

Thread Based Test Management
This method was introduced by James Bach in 2010. A thread is a test idea or a test activity. Thread Based Test Management is a generalization of Session Based Test Management. Thread Based test Management is an easy and quick method to start and is not limited by time. The method starts by listing the test ideas as threads and then arranging these threads using mind map tool [9].

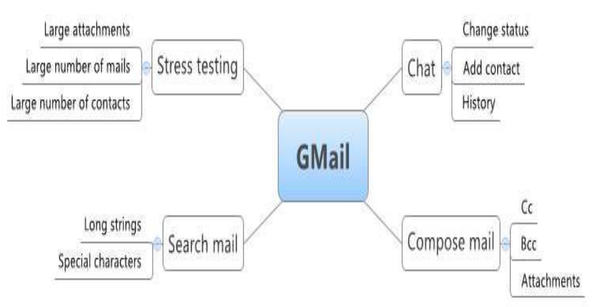

Figure 1: Thread Based Test Management

3. xBTM

xBTM was first introduced by Michael Albrecht and Christin Wiedemann of AddQ Consulting in 2011. xBTM is the combination of Session Based Test Management and Thread Based Test Management where "x" represents number of sessions or threads. xBTM starts by using session based test management or thread based management. A mind map is designed based on the context by listing test ideas for test activities, estimating the number of charters needed and updating the mind map as and when the test progresses [9]**.**

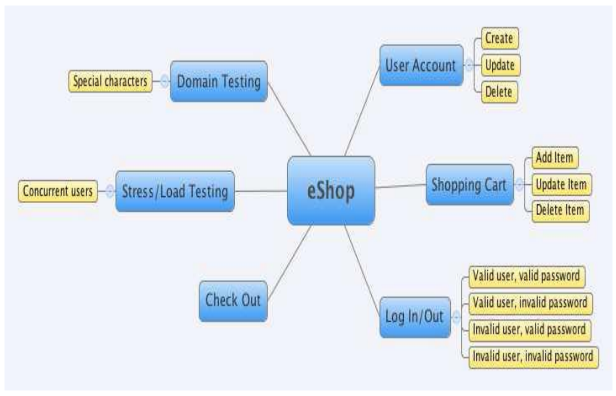

Figure 2: xBTM

Tour Based Exploratory Testing

An exploratory tester justifies for five specific properties such as user inputs, state, code-paths, user data and execution environment while performing exploratory testing. During this process, the tester would be taking decisions regarding small things such as choosing among atomic inputs, arranging atomic inputs in combination or in sequence. The tester does this with the use of input filters, use of input checks and exception handlers etc. For taking large decisions concerning feature interaction, data flows and choosing the path through UI exploratory testers use tourism metaphor where the exploration of the software is done using the tools such as organized tours, guidebooks, maps and local information. This helps the tester in setting goals during testing.

"Tourist " metaphor is suitable for exploratory testing where a tester tries to explore a new destination. An exploratory tester selects a combination of features while performing exploratory testing similar to a mix of landmarks and sites as selected by tourists on a tour. As tourists partition the destination into physical boundaries or districts such as business district, entertainment district, theatre district etc an exploratory tester partitions the features of an application.

1. Business District - It contains the features and functions on which the business of an organization mainly depends. It consists of starting and the shutdown code. Tours associated with the district are

i) The guidebook Tour

ii) The Money Tour

iii) The Landmark tour

iv) The Intellectual Tour

v) The FedEx Tour

vi) The After-Hours Tour

vii) The Garbage Collector's Tour

2. Historical District - The main aim in this district is to test the functionality of the legacy code and verify the bug fixes. The tours associated with the district are

i) The Bad Neighborhood Tour

ii) The Museum Tour

iii) The Prior Version Tour

3. Tourist District - This is district where new testers are always attracted than the experienced. The tours associated with the district are

i) The Collector's Tour

ii) The Lonely Businessman Tour

iii) The Supermodel Tour

iv) The TOGOF Tour

v) The Scottish Pub Tour

4. Entertainment District - This district involves supporting features such as formatting texts, modifying backgrounds in a word processor than the main features. In a word processor, the supporting features are formatting texts, modifying backgrounds and templates etc. The associated tours are

i) The Supporting Actor Tour

ii) The Back Alley Tour

iii) The All-Nighter Tour

5. Hotel District - This district often involves tours where a tester tests the secondary and supporting functions. Associated tours are

i) The Rained Out Tour

ii) The Couch Potato Tour

6- Seedy District - In this district the testers test the sections of software that are vulnerable. The tours associated are

i) The Saboteur

ii) The Antisocial Tour

iii) The Obsessive-Compulsive Tour

Hybrid Exploratory Testing Techniques

Scripted and Exploratory Testing

Scripted and Exploratory Testing coexists where the scripts present a structure and exploratory testing adds variation to increase the effectiveness of the combined approach. This approach begins with formal scripted testing, which is later continued by exploratory testing..

Scenario Based Exploratory Testing
End-to-End Scenario Testing is performed by testers when they perform manual testing. The popularity of scenario testing is because of the confidence it gives to the users about the product that it will work as expected. A Scenario performs the following functions

- tells a user story
- describes a requirement
- demonstrates how a feature works
- demonstrates an integration scenario
- describes setup and installation
- describes cautions and things that could go wrong

Scenario Based Exploration is performed using existing scenarios and variations are injected as and when required, thus translating a single scenario into many test cases by considering choices in input selection, data usage and environmental conditions. There are two ways through which variations are injected into a scenario testing: by scenario operators and tours [10].

Exploratory Testers

Exploratory testing is based on the knowledge of the testers that varies from person to person just as any other kind of knowledge. The knowledge that the testers possess may be obtained from previous projects, or drawn on the experience of other people. Exploratory testers may also have on the knowledge from the training they received or from the published sources [11].

Exploratory testers use heuristics to make decisions. Heuristics is defined as "Of or relating to a usually speculative formulation serving as the guide in the investigation or solution of a problem"(American Heritage Dictionary of the English Language 3rd edition). James Bach has developed a model named Satisfice Heuristic Test Strategy Model that shows the types of the knowledge used by the explorers. In this model, environment of the project, quality criteria defined on the project and elements of the product being tested combine with test techniques to affect the quality of the product. Each of these elements in turn consists of several components that help the testers to determine the information they need for the project [11].

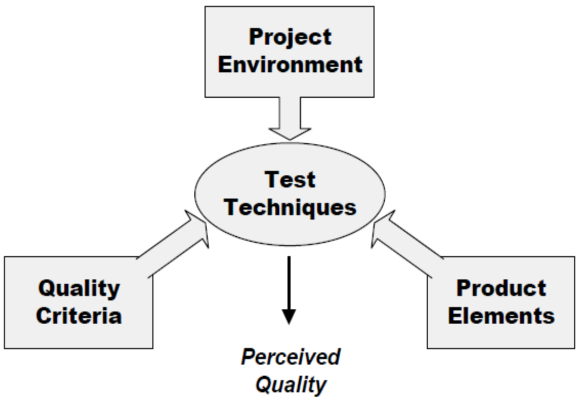

Figure 3: Satisfice Heuristic Test Strategy Model

Learning Styles and their applications to Exploratory Testing
An important activity of exploratory testing is learning about the software, including its weaknesses, potential failure modes, potential applications, market, configuration variability. Several models of learning styles have been proposed. A learning style is a person’s “characteristic strengths and preferences in the ways they take in and process information In 1988 Felder-Silverman proposed a model of learning styles that indicates a person's predilections on five continua: Sensory/ Intuitive, Visual/Verbal, Inductive/Deductive, Active/Reflective and Sequential/Global [12].

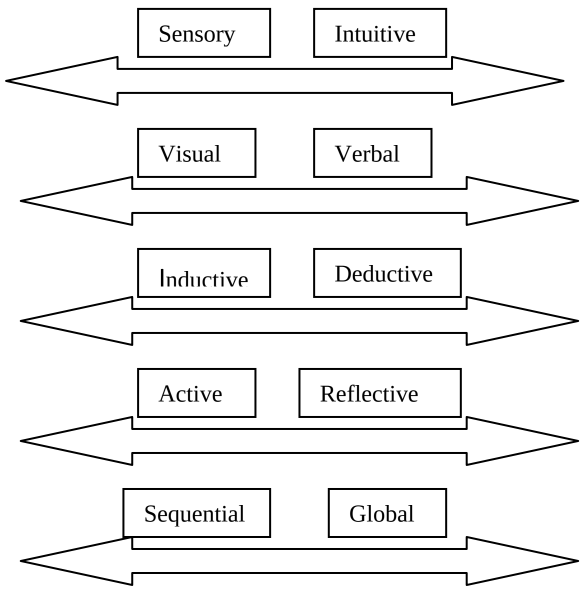

Figure 4: Felder-Silverman Learning Styles

Sensory/Intuitive
According to this model, a person with a preference to sensory information is one who relies more on the information he receives through his external senses, while a person with a preference for intuitive information relies on his internal information (generated from memory, conjecture, and interpretation) and intuition. The sensory-based person focuses on his actual observations of the software. The intuitor will focus on the internal model of the software that is under test.

Visual/Verbal
Visual learners retain information they get from visual images such as pictures, movies, diagrams or demonstrations. Verbal learners retain information they hear (or read) such as lectures, written words, and mathematical formulas.

Visual learners will tend to work with an internal model that is picture-based such as a set of UML diagrams, flowcharts, or even mental screenshots. Verbal learners would use a textual model such as textual description of the system for testing.

Inductive/Deductive
An inductive learner prefers to work from specifics and derive the generalities, while a deductive learner starts with the generalities and applies them to the specific situations.

An inductive learner collects as many specifics as possible and generalizes them to the application. A deductive learner does testing by keeping a collection of general principles and heuristics and looks for ways to specifically and applies these generalities.

Active/Reflective
Active learners discuss with others or may do experiment with information as soon as they get it. Reflective learners think about information before they use it. They prefer to work alone.

An active tester often executes many test cases rapidly and views each test case as an experiment. A reflective tester, on the other hand, executes few test cases.

Sequential/Global
Sequential learners begin learning in small bits incrementally building on the knowledge they have already learned. Global learners, however, tend to learn in large portions.

A sequential learner builds information and knowledge in a logical progression, while a global learner needs critical pieces of information in order to get the understanding of the subject [12].

Other Styles of Explorations
There exist other styles of explorations for real-time and embedded systems to uncover uncertainties [13].

Environmental explorations
Environmental explorations simulate uncertainty in the environment in which the system is operating. These uncertainties may arise from operating system anomalies or from operational domain disturbances, such as a power surge or a violent storm.

Input explorations
Input explorations simulate uncertainties such as false or missed interrupts, anomalous data, and deliberately poisoned data which further leads to series of failures that overload the system.

Output explorations
Output explorations simulate gross or subtle defective output from a software control system which further perturbs the system response.

State explorations
State explorations simulate internal faults, such as jumped program counters, which further can lead to uncertainty of program state that is difficult to diagnose, and nearly impossible to recover from.

Behavioral explorations
Behavioral explorations simulate a wide class of timing and scheduling problems that are the characteristic of real-time systems.

Language explorations
A set of explorations are needed to test the compiler, and other systems programs involved in the production of the executable code (debuggers, linkers, loaders, etc.).

COTS explorations
These explorations uncover problems in software furnished by third parties such as commercial vendors, or open source software.

These explorations are extensively used in testing various real-time embedded systems for avionics applications, including the Space Shuttle Inertial Measurement Unit, satellite systems, and other navigation systems [13].

Tools supporting Exploratory Testing
Tools are infrequently used in Exploratory Testing. Following table gives the list of the tools supporting exploratory testing [14].

Table 1: Tools Supporting Exploratory Testing

	Tools Supporting Exploratory Testing
	Mind Maps(e.g.XMind)
	Custom made tool
	Rapid Reporter
	Evernote
	Excel
	qTrace
	Vim-Editor
Software	Jira Test Sessions
	One Note
	Perclip
	IETester
	BB Flashback
Non-Software	Literature
	PostIts
	Checklists
	Paper & Pen

Challenges of Exploratory Testing
Following challenges faced by conventional test case based testing are also applicable for Exploratory Testing.

- Learning challenge is about knowing the program.
- Visibility is about determining the progress of the testing process
- Control is about setting internal data values
- Risk/Selection is about determining the best tests to run
- Execution is about the most efficient way to run the tests
- Logistics is about determining the environment is needed to support test execution?
- The Oracle problem is about determining if the test result is correct or not.
- The Reporting is about replicating a defect and effectively reporting it.
- Documentation challenge is about determining the test documentation required.
- Measurement is about deciding the appropriate metrics.
- Stopping challenge is about deciding when to stop testing?

Exploratory testing is an experience based testing and differs highly from the document driven Case Based Testing. There are only few research articles and books published on exploratory testing. Practitioner reports on exploratory testing assert ET is both effective in detecting defects and cost efficient. More research is required to better understand all the aspects of exploratory testing.

In conclusion, exploratory testing is encouraged in the practitioner literature and scientific studies of exploratory testing are yet an emerging technique. Effectiveness and efficiency of ET approach are supported by studies comparing

ET with other testing approaches. This research paper attempts to help interested testing and research community to gain more knowledge about exploratory testing, its various techniques, metrics, tools and challenges.

4. REFERENCES

[1] Humphrey, Watts S. “The Software Quality Challenge.” CROSSTALK: The journal of Defense Software Engineering (June 2008).

[2] V. Suma, T. R. Gopalakrishnan Nair, “Defect Management Strategies in Software Development”, Book on Recent Advances in Technologies", ISBN 978-953-307-017-9, pp 379-404, Intec web Publishers, Vienna, Austria, November 2009.

[3] Juha Itkonen, Mika V. Mantyla and Casper Lassenius, " The Role of Knowledge in Failure Detection During Exploratory Software Testing", IEEE Transactions on Software Engineering, May 2011,

[4] Chris Agruss & Bob Johnson, " Ad Hoc Software Testing: A perspective on exploration and improvisation", 2000.

[5] Cem Kaner, "A Tutorial in Exploratory Testing", April 2008.

[6] Itkonen, J. and K. Rautiainen, “Exploratory testing: amultiple case study,” Proceedings of InternationalSymposium on Empirical Software Engineering, 2005, pp. 84-93.

[7] James Bach, “Exploratory testing,” in The Testing Practitioner, 2nd ed., E. van Veenendaal, Ed. Den Bosch: UTN Publishers, 2004, pp. 253–265.

[8] Jonathan Bach, "Session-based test management." Software and Quality Engineering Magazine 2.6 (2000).

[9] Christin Wiedemann, "Exploratory Testing on Agile Projects Effective, Efficient and Engaging”, SQDG, Calgary, January 15th, 2013.

[10] James A Whittaker, "Exploratory Software Testing - Tips, Tricks , Tours and Techniques to guide test design", Pearson Education, 2009.

[11] Andy Tinkham, Cem Kaner, " Exploring Exploratory Testing", 2003.

[12] Andy Tinkham, Cem Kaner, "Learning Styles and Exploratory Testing", 2003.

[13] Phil Laplante, "Exploratory Testing for Mission Critical, Real-Time and Embedded Systems", IEEE Transactions on Reliability, pp. 449-482, Volume 59, Number 3, September 2010.

[14] Pfahl, Dietmar, et al. "How is exploratory testing used?: a state-of-practice survey." Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering , Measurement. ACM, 2014.