Exploratory Risk-Based Testing Approach

MAHMOUD SABRA
Information Systems Department, Faculty of Computers & Information Sciences
Mansoura University, EGYPT
m.sabra@egyptnetwork.com

HAITHAM EL-GHAREEB
Information Systems Department, Faculty of Computers & Information Sciences
Mansoura University, EGYPT
helghareeb@mans.edu.eg

TAHA EL-ARIF
Computer Science Department, Faculty of Computer & Information Sciences
Ain-Shams University, EGYPT
taha_elarif@yahoo.com

Abstract: - Software testing is an integral part of software engineering. During the software development life cycle, testing is highly needed to assure the quality of the software process and product. The software development industry spends more than half of its budget on maintenance related activities. Software testing provides a means to reduce errors, cut maintenance and overall software costs. This thesis discusses a new approach to software testing that enhances testing process using a hybrid approach of exploratory testing and risk-based testing. Early in the history of software development, testing was confined to testing the finished code, but, testing is more of a quality control mechanism. If software cannot be tested exhaustively, it must be tested selectively. A risk-based testing approach provides proactive opportunities to reduce the levels of product risk, starting in the initial stages of a project.

Key-Words: - Exploratory Testing; Quality Control; Risk-Based Testing; Software Testing.

1 Introduction
Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance software quality. Software testing is a trade-off between budget, time and quality. However, as the practice of software development has evolved, there has been increasing interest in expanding the role of testing upwards in the SDLC stages, embedding testing throughout the systems development process. The rapid change in the software development process brings a lot of challenges to this field. In order to encounter these challenges, the companies search for more agile and cost effective methods [1]. This attitude is visible in all phases of software development process. The methods, approaches and techniques of software testing have developed to adapt this change.

Exploratory Testing (ET) has been used by the testers knowingly or unknowingly. But still it faces difficulties in being realized as an approach for performing effective software testing [1] [2]. One of the major reasons of ET not being realized – although practiced – is lack of scientific researches. Therefore, ET seems to be a victim of a number of misconceptions [3] [4]. The results also showed that ET approach was preferred to be used in combination with other testing techniques.

In recent years, ET approach has gained focus and popularity especially among professional testing practitioners. Exploratory testing is done with a view that software is bound to have defects in it. The tester has to somehow identify these defects using any possible combination of techniques that forces the software to behave in an undesirable manner [5].
There is no single technique is sufficient for defect detection problem (Houdek et al., 2001). Instead, the combination seems to be right approach [6]. Kasurinen et al. (2010) observed exploratory testing as part of a more generic risk-based approach to testing (RBT) [7]. Bhatti and Ghazi (Bhatti and Ghazi, 2010) asked the interviewees about the techniques and approaches that they use in combination with ET mind set. Risk based testing was mostly used by four interviewees, because, it at the first hand tests the most critical features [5].

2 MANUAL TESTING

In Manual Testing, a Human will interact with a system in order to attempt to uncover bugs. Manual Testing is generally understood as scripted manual testing, in which testers follows a predefined script when interacting with the system. These scripts are usually created by test engineers, and then run later by software tester. Scripts for manual testing can be viewed as test programs written to be executed by humans rather than by machines. Scripted manual testing is a tedious process when done repetitively (as for regression testing).

The graph shown in Figure 1 shows a hypothetical example of the amount of effort required to perform this degree of testing over several iterations of software development [4].

3 EXPLORATORY TESTING

Exploratory Testing is an approach that does not rely on the documentation of test cases prior to test execution. This approach has been acknowledged in software testing books since the 1970’s [8]. ET keeps testing software after executing the scripted tests and avoid investing effort to carefully designing and documenting tests when the software is in an unstable state and could be redesigned soon [9].

As discussed in (Naseer and Zulfiqar, 2010), ET is an approach to software testing that emphasizes the freedom and responsibility of each tester to continually optimize the value of his work by treating learning, test design and test execution as mutually supportive activities that run in parallel throughout the project [3].

Finding defects is the primary purpose of ET and documenting the results of the testing is found more important than planning and scripting the test execution paths beforehand. However, a certain degree of planning is needed for ET. Copeland (Copeland, 2004) suggests performing “chartered exploratory testing” where the charter may define what to test, what documents are available, what tactics to use, what defect types to look for, and what risks are involved. The charter defines a mission for testing and works as a guideline [9].

Scripted testing has advantage of consistency, while exploratory testing benefits from its reliance on human ingenuity. Both avoid the difficulties with creation of an automated test oracle, by relying on a human’s ability to judge whether or not a system meets expectations.

Despite its unplanned, freeform nature, exploratory testing become accepted in industry, and is felt to be an effective way of finding defects. Practitioner literature argue that exploratory testing reduces overhead in creating, maintaining documentation, helps team members understand the features and behavior of the application under development, and allows testers to immediately focus in productive areas during testing [4].

This approach adapts well to rapidly dynamic product requirements. ET brings more freedom and fun along with responsibility to their work, because it challenges the intellect of tester. While performing during time constrained situations, ET increases the sense of accomplishment in testers. It will be more helpful to report every issue and defects with details about how they were generated or found. This information will later help to reproduce the bugs [5]. Further, a recent academic study shows the exploratory testing is at least as effective at catching bugs as scripted manual testing, and is less likely to report that the system is broken when it is actually functioning correctly.
3.1 Misconceptions

There are number of misconceptions related to ET including the understanding of ET as a completely and ad-hoc approach to test. This is considered to be one of its major disadvantages. General and prime misconception is being considered as a technique. Kaner (kaner, 2006) state that everyone does ET to some degree and it is an approach not a technique. ET being an approach and relatively a new phenomenon has created commotion for its adaptability.

Exploratory testing or its results are itself unable to gratify the manager or customer. Manager and customer think it as playing with system, unstructured, unrepeatable, unaccountable and the results are not considered reliable (Naseer and Zulfiqar, 2010; Bolton, 2007). It is the biggest reason that ET is not practiced in industry. But according to (Naseer and Zulfiqar, 2010) it is due to lack of knowledge and experience in ET. Being a new concept it requires a lot of experience and skill for producing valuable results. But also when customer observes that ET has identified critical bugs they start accepting the results produced by ET.

Naseer’s (Naseer and Zulfiqar, 2010) study shows that the biggest disadvantage of exploratory testing is that ET is more prone to human error. But if we analyse, then systematic testing is also prone to human errors, since test design is done by human and there exists equal chances to committing some sort of mistake. Test case design can miss out some important aspect of system which might be considered small or unimportant to be included in test design. Whereas, during system exploration in ET approach, it is more likely to find such defects [5].

3.2 Knowledge as a Test Oracle

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. Failure recognition 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 [10].

The challenge of finding a reliable oracle is referred to as “the oracle problem”. The Test-Case-Based Testing (TCBT) paradigm aims to solve the oracle problem by predefining the expected result in detail. In practice, requirements, specifications, and thus test cases are seldom perfect in terms of comprehensiveness and accuracy. From the empirical research on real-world testing activities, it seems that a human oracle is in many cases the way test results are evaluated in practice. In industrial practice, the recognition of failures is left as a human decision. Thus, the oracle problem is highly relevant in manual testing, and typically solved using the personal knowledge of testers and varying types of documentation.

The challenge of using a human oracle is that humans are fallible, i.e., testers do not always recognize a failure, even when a test case reveals it. In an experiment by Basili and Selby [11], subjects recognized only 70% of observable failures. On the other hand, human testers are able to detect incorrect results using experience based partial oracles, even when they cannot know the exact correct result.

The core idea in ET is that the tester can and should use any available sources of information in the testing, which suggest that the oracle in ET can be any knowledge, documentation, model, or software available to the tester [12].

3.3 Session Based Test Management

Exploratory testing should not be unplanned, unstructured, or careless testing without any strategy or goals. Exploratory testing can be as disciplined as any other intellectual activity. It can be structured, managed and planned as long as the planning is not extended to describe detailed tests on the level of test cases.

Jonathan Bach (J. Itkonen and K. Rautiainen, 2005) has published an approach to ET called Session-Based Test Management (SBTM). This session-based approach brings a clear structure to loosely defined exploratory testing. It is an approach to planning, managing, and controlling ET in short (almost) fixed-length sessions. Each of these sessions is planned in advance on a charter sheet, which is a high level plan for test session [9]. The detail of test cases executed in a session is not pre-specified but a clearly defined report and metrics are collected during the session. The results are briefed afterwards between the tester and the test manager [5].

4 Risk-Based Testing

A risk based approach is introduced for testing. Risk based testing techniques is widely used in where the risk involvement is higher. It is more perceptive than calculative. RBT is significant if it can determine and ensure the quality of product quantitatively using metrics.
Objective of Risk Based Testing is to minimize the risk associated with program or project. This makes it crucial to apply RBT while doing testing. Finally the risk outcome will be multiple of business impact, probability or estimated risk [13].

Risk-based testing is systematic. It takes a logical, orderly, and consistent approach to prioritizing requirements based on business and technical risk.

### 4.1 Risk Analysis

Risk analysis systematically examines the business and technical risk of the requirements to determine priorities that provide the basis for application design, development, and the testing strategy for a project. From a testing perspective, the priorities resulting from a risk analysis provide the primary input to a testing strategy for the project. When the risks, priorities, and testing strategy are all aligned, the testing team can perform the right tests and find the defects that, if undetected, could cause serious damage to the delivered system.

Final decision will be based on outcome of risk assess. From the risk identification and analysis it will become more feasible to prioritize the each component of system on the basis of risk. Once the risk has been identified, risk avoidance plan is generated to minimize or to reduce the risk. Method of assessment of risk given in risk analysis model given in figure 2 [14]:

![Risk Analysis Model](image)

**Fig.2. Risk Analysis Model.**

### 4.2 Test Case Prioritization using Risk Exposure

Test Case Prioritization (TCP) schedules test cases in an order that increases the overall effectiveness of the testing process in meeting a performance goal. Employing the proposed method, test cases are scheduled in an order that achieves a greater rate of fault detection, in as short time as possible. We prioritize test cases based upon the risk exposure associated with individual risks [15].

### 5 Hybrid Approach

Test cased based testing approach is mostly followed by ET. In TCBT; testers can miss out serious bugs, because they don’t see the things around it. So if we have blend of both scripted and exploratory testing then ET generates more valuable information, better test results and test coverage, which ultimately identifies more bugs, because they are hidden. Test cases are written to meet the functionality of the requirements, but it became invalid when functionality is developed [5].

ET is one of the most widely used approaches in the field of software testing. It is also the least understood approach despite being used by most testers as part of their day-today activities. For example, any tester who deviates from a scripted process to characterize a defect he found or to verify a bug fix is doing ET. However, some testers are reluctant to describe their work as exploratory because they think the term connotes random work or that it is merely an excuse for testers to avoid the documentation and planning.

Competently done, exploratory testing is neither an excuse to avoid work nor a waste of time due to randomness. The essence of exploration is an active, risk-focused investigation of the product. Rather than designing tests early, when tester is just beginning to learn what the product is, how it can fail, and who will use it to do what, on what platforms, the exploratory tester might use any test technique, and will probably use several different ones, in his quest to learn more about the product and its weaknesses and designs tests to exploit the knowledge he has just gained [16].

ET can be used in different manners i.e. ET solely, in combination with other techniques, techniques being used in exploratory fashion and complementing other techniques. It is believed by most of the practitioners that exploratory testing is way of thinking about testing and can be used with any method of testing [3]. According to testers in industry have following preferences showed in table 1.
### Table 1. Preference of Using ET Approach in Industry.

<table>
<thead>
<tr>
<th>Preference</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>In combination with other testing technique</td>
<td>76%</td>
</tr>
<tr>
<td>As Complementary</td>
<td>48%</td>
</tr>
<tr>
<td>Using other technique in an exploratory manner</td>
<td>30%</td>
</tr>
<tr>
<td>Solely exploratory</td>
<td>22%</td>
</tr>
</tbody>
</table>

All companies had found ET practical for giving fast feedback to developers regarding newly developed features. When the developer(s) had completed a feature, a tester would quickly explore the new feature and give feedback to the developer(s). The feedback could range from reporting defects to pointing out usability problems or misconceptions regarding the customer requirements. This information could then be used to steer the development, if necessary [9].

Business, technical, and testing stakeholders align their priorities at the beginning of a project. After every stakeholder agrees to common priorities, the testing team can determine the focus of its testing activities in every project phase. Getting all project stakeholders to agree on and follow priorities that are aligned to business needs significantly increases the likelihood that the delivered system will meet the needs of its users.

The risk-based testing process is not tied to any particular development methodology. The risk-based testing process can also be adapted to meet the needs of any development methodology, including waterfall, iterative, and agile. Ultimately, this process is versatile enough to apply to any application written in any programming language running on any machine with any operating system.

Figure 3 provides an example of incorporating risk-based testing into iterative development at HP.

Iteration 0 focuses on developing an initial testing strategy from equally initial requirements.

![Fig. 3. Risk-based testing and iterative development.](image)

Iterations 1—5 focus on test planning, preparation, and execution by iteration, including, where necessary, updates and refinements to the testing strategy to correspond with updates and refinements to the requirements. During each iteration, the testing team uses a risk-based approach to systematically designing tests and maintaining a requirements traceability matrix to ensure prioritized coverage of requirements.

Iteration 6, the final iteration, provides an overall test results report that supports a well-informed decision about the readiness of the application for release based on outstanding risk. It also provides valuable lessons learned that can deliver tangible benefit to future projects [14].

### 6 Conclusion

Historically, Software testing has been widely used as a way to help engineers develop high-quality systems. However, pressure to produce higher-quality software at lower cost is increasing. Existing traditional techniques used in practice are not sufficient alone for this purpose.
This paper relies mainly on the combination of an ET and RBT Approach.

The literature presents some misconceptions about exploratory testing, and I proposed solutions through new exploratory techniques, Activities, and Approaches for managing exploratory testing.

RBT is intended to focus testing upon more risky aspects of the system. The results of RBT are assessed in terms of how severe the risks are. Also, we can prioritize our test cases according to the continuously assessment on each iteration.

I further investigated how the RBT and ET integrated in software development process. And how to use them throughout the SDLC in balanced way as a hybrid approach in order to making testing better and improve customer satisfaction by delivering a more error free product.

References:

Mahmoud I. Sabra, Software Test Engineer at EgyptNetwork Company, Mansoura city, outsourced to ITS Company, Cairo city, Egypt. He was graduated from Faculty of Computers and Information Sciences, Information Systems Department, Mansoura University, Egypt, 2010. He is certified from Information Technology Institute (ITI), Egypt.

Haitham A. El-Ghareeb, Associate Professor in Information Systems Department, Faculty of Computers and Information Sciences, Mansoura University, Egypt. He has received Bachelor, Masters, and Ph.D. in Information Systems in 2004, 2008, and 2011 respectively from the same faculty currently working at. He is a member of different distinguishable computer societies. Haitham has contributed to many research papers, edited two books, and supervising masters and Ph.D. He is contributing in different funded research projects. Visit Haitham’s Website: www.helghareeb.me