INDIUM

LET THE ADVENTURE BEGIN

- THE 80:20 TESTING

TABLE OF CONTENTS

1 Abstract

Gone are the days when we do most of our testing against documents like requirements, use cases, user stories, test cases, etc. Although formal documentation / planning are done in most testing process, how many times we find bugs just clicking at random?

Exploratory testing is a test strategy where the tester has complete freedom over the application where he can chose to navigate at his free will not forgetting the main goal – Quality. Conducting Q&A sessions among peers on the behavior of the product, discussing error handling techniques of the product, learning from RCAs of defects are a few places where one can begin his Exploratory testing. End-to-end scenarios can also be put together for future reference.

Carrying out exploratory testing may not be as easy as it sounds. There are challenges like resource's lack of application knowledge, recording time for this effort, reliability of this testing's outcome, etc.. This white paper brings out the importance of this test approach by performing it in a timed environment and it also highlights the benefits of introducing this testing type in our test life cycle.

The approach in this paper was built to overcome the challenges such as non-coverage of functionalities in documented test cases, underutilization of skilled testers, wide compatibility combinations to cover in a short span of time, etc.

Key Takeaways:

The key takeaways in paper are:

- ✓ How to improve test coverage beyond scripted tests
- ✓ Ideas to kindle your creativity in testing
- ✓ A research oriented approach which will assure good results
- ✓ A cost efficient solution to widen test coverage

2 Testing – The Need to be Quick

In the process of quick adaptation to new technology and resources, the environment is unwilling to take its time for the quality achievements. The blame is not on the inventions, but on the competitive world we are living in. The course of software development has been extremely affected due to this quick turnaround needs. The recent lifecycle has become compact with forceful changes to accommodate completion in intense timelines. With speed being the need of the hour, scripted tests limit the scope of testing. This leads to the evolution of Exploratory testing raising the bar of software quality. Testing is generally exciting and when not restricted, it becomes more fun allowing the tester to uncover the unknown.

3 General Challenges

'Instant' being the word of the era, this section talks about some of the challenges faced in our test process which also lays the basis for this paper:

- **Constant change & Lack of documented requirements:** Dynamically changing requirements with short deadlines affect scripted tests the most. Reusability in such cases is absolutely ruled out.
- **Time constraints & Insufficient Test Coverage:** With short timelines, preparation of test scripts ensuring complete coverage is highly complicated.
- **Early detection of defects:** With short iterations, the need to find bugs early is added pressure.
- **Code breaks & Performance Bottlenecks:** Frequent unstable builds, incompletely developed requirements, lands the tester in a very dicey situation making them loose grip on the scope of testing.

4 Exploratory Testing – An Interesting Usage of Intelligence

Exploratory testing can easily be mixed up with other testing techniques like ad-hoc testing, Monkey testing, etc.. The question is when to use what. It depends on the situation and style of testing required to suite it. The tester lands in situations when requirement changes along with an unsettled process. This requires a well groomed approach yielding maximum test coverage and high defect catch rate which ultimately results in the quality of output.

The key for successful Exploratory testing is to not only to validate randomly, but also involves strategies based on the experience of the product usage, efficiency of the tester, analysis of previous results and thorough knowledge on the concept of test techniques.

The following sections attempt to document some of the best practices that helps get the best out of Exploratory testing. It is not to stress that Exploratory testing is the only option for a quality output but to say that it helps in making testing a complete process with maximum utilization of minimum resources. To understand this best: It is like in the game of Cricket where the batsman can take advantage of the Power Play overs by not having to think twice before hitting hard. With minimal planning, he can be sure to achieve maximum outcome which is a bonus for the score and also adds to the excitement. He is not restricted to a frameset and can relax at playing it safe.

5 Our Derived Exploratory Approach

There were days when a project could be completed with no defined roles or processes and there was no need for a competitive timeline. But lessons learnt from these have now brought us to this stage where we have several methodologies claiming to suite every one of our need. With this spread, which one would suite for a requirement as shown below?

Requirement: A changing combination of environments

Release Frequency: Weekly (sometimes even lesser)

Content Change Rate: Daily

Test Life Cycle: 3 - 4 days

Testing Scope: Functionality; Compatibility

Risks:

- Requirements are just storyboard discussions
- Unrelated weekly changes in requirements
- Short development cycles
- Platform dependent test cases
- Support quick updates throughout the process

Types of Testing: Scripted and Exploratory

5.1 General Exploratory vs. Derived Exploratory

This section details out the difference one follows in general exploratory testing process and how we have tweaked or made additions to it in our derived approach. The following diagram shows the general phases in our derived exploratory process.

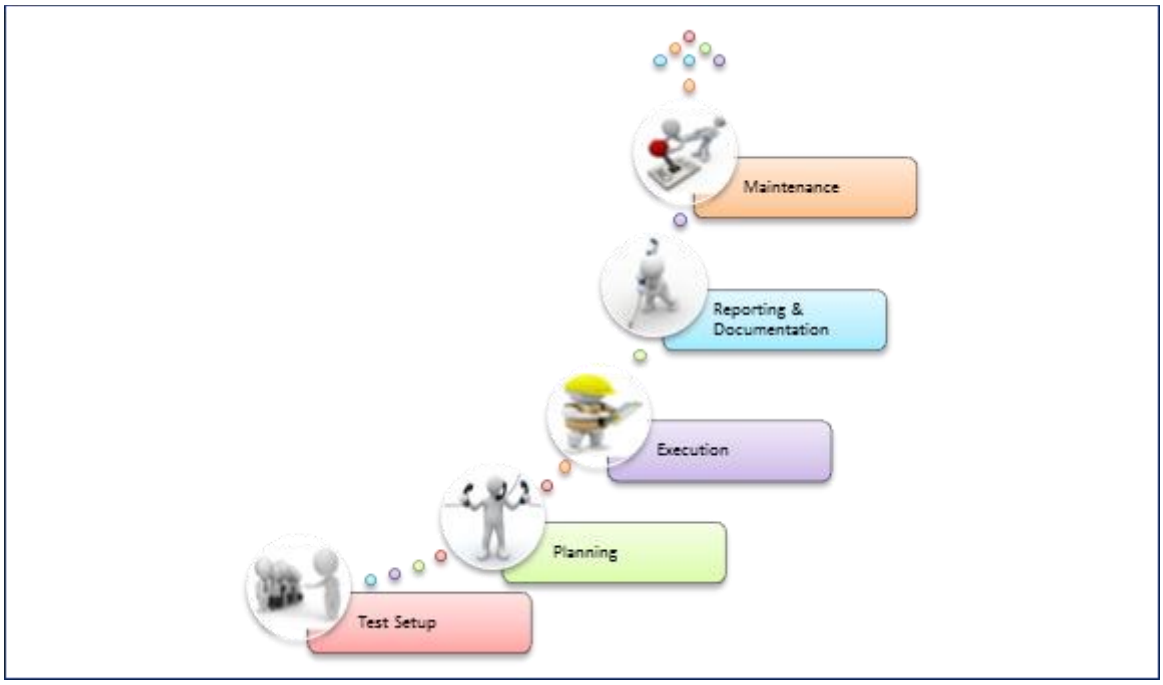

Figure (i) - Derived Exploratory - Process Flow

5.1.1 Team Setup

General:

In the general Exploratory approach, most of the time it is the skilled resources who are sort after. This is to take advantage of their knowledge in the product and their experience. This is a good way to do it but why not use the others in the team who are equally capable testers but may be just a little behind on application know-hows?

Derived:

In our derived approach, we have a few easy to maintain documents (discussed in the Planning and Execution phases) that come in handy here. This will help convert any resource to be able to handle exploratory testing.

5.1.2 Planning

General:

There is a general myth that Exploratory testing is done with no or minimal planning. This has also become the reality in some QA processes. Usually exploratory testing takes the course of error guessing, past defect experience, discussions and KT sessions, etc.

Derived:

Planning goes miles. With this derived approach, we would like to stress on taking a few proactive steps during the planning phase. A few listed here have worked wonders for us.

- **Know your domain and your user:** Access customer logged issues and identify Critical ones for your testing.
- **Data analysis:** Have a data bank with tricky combinations.
- **Areas to focus on:** New feature integrating with old; Values within generated reports; etc.
- **RCA:** Conduct RCA on logged bugs and maintains a tracker. This will help discover the most defect prone areas in the application.
- **Know your environment:** Be updated on plugins and add-ons. A new invention can cause ripples.
- **Identify test cases from the suite:** Pick end to end scenarios that touch various parts of the application and venture around these. Document observations and raise clarifications.

It may seem that planning in our approach is taking up a considerate amount of time but do not go with that wrong notion. The time spent on this phase will help in the long run.

5.1.3 Execution

General:
When there are no planned deadlines and there are moving targets, it is usually difficult to conduct focused testing in general Exploratory. There are chances of missed combinations when it is done totally undocumented and there can also be instances of time spent twice on the same modules.

Derived:
The planning phase feeds into this but it does not stop there. Before we move on to how to expand Exploratory test coverage, the most important thing to remember here is to prioritize what has been planned. In addition to this, here are a few pointers to expand your Exploratory testing skills:

- Standardized charter: Do not fail to note down even the smallest observation. Some of these might not turn to be bugs but once documented, it will help save time in not doing redundant testing around it the next time. Categorize both observations and defects found in this process into something like:
    - Complex – Issues with large impact area
    - New – Issues with no history in the product
    - Changed – Issues with alternate impacts

- Critical – Issues with ultimate damage with no further progress
- Third party – Issues in the product but developed outside the project
- Buggy – Known area with lot of problems
- Recent Failure – Issues with recent history of failures
- Distributed – Issues which will impact a failure in rest of the system
- Strategic – Issues in areas with special importance with respect to the business

- Soap opera testing: This taking a step back and going to fix our scripted regression suites. Make the suite cover exaggerated activity in condensed time. These can be ideal test scenarios to go back if there is a boon of an extra day of testing.
- Negative testing: Be cynical in your testing approach. Always see the glass half empty. Imagine being a customer who believes in breaking the system.

5.1.4 Reporting and Documentation

General:
In the general approach it gets tedious to collate all that was experience because it was not done in a streamlined manner. Observations / defects not formally documented may lead to two things i. missed scenarios and ii. Redundant testing which require the same effort.

Derived:
The best part of the derived approach is that most documentation is done in parallel and not much time is spent here is specific. All one has to do is consolidate results to be used for future execution. The very first time the derived approach is applied a tester will be able to see results with the growing documentation that actually did not take up time but grew along with the testing process.

When the Exploratory procedure is properly charted, reviewed and supervised, all one has to do is pick a tester who is available not really worrying about his knowledge in the application.

5.1.5 Maintenance

General:
In most cases, this phase is given the least priority as the testing part is considered done and the application seems to behave as it should. Usually this phase does very little for Exploratory. Concentration is on updating the scripted tests to match Exploratory testing results / observations. While this is still important, we have to take the road where maintaining the scripted tests will become minimal.

Derived:
As the derived process matures, the steps taken in the above phases will ultimately lead to clarity in process and documentation. Practicing test techniques like the soap opera testing, etc. and maintaining the various trackers discussed will not just help in fine tuning the Exploratory process but also the scripted tests.

6 Results & Metrics

- Highlights Defects found in Scripted vs. Exploratory
- Introduction of refined exploratory testing contributed 20% more defects in High and Medium category
- Refined Exploratory testing helped to uncover affected component under UI, Display and other hidden functionality

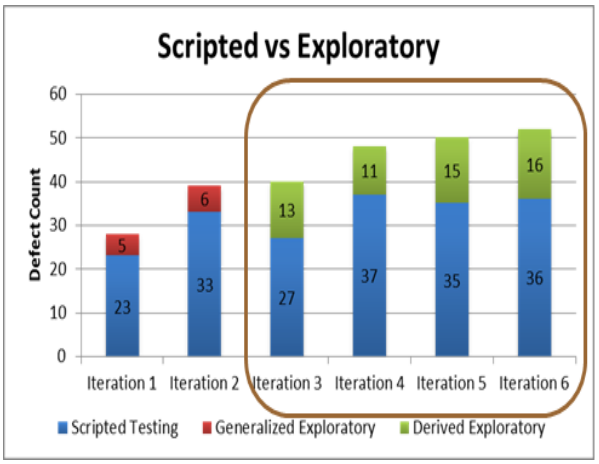

Figure (ii) – Scripted vs. Exploratory

- Shows the Test case defects
- Projects with consecutive iterations need lots of effort in script maintenance, which helps in hands-on testing
- Integrated Exploratory testing approach with scenarios, helped to make the documentation more stable therefore reducing unnecessary time spent in test script fixes.

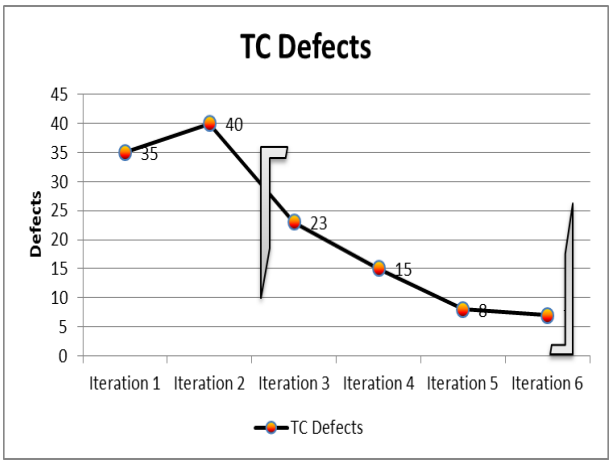

Figure (iii) – TC Defects

- Highlights the Distribution of defects based on the testing type
- Defects with Derived approach is 20% more – Older is 7%
- This 20% raised the confidence in the apps by 80%
- Proves Derived ET approach is a best practice in faster iterations / builds

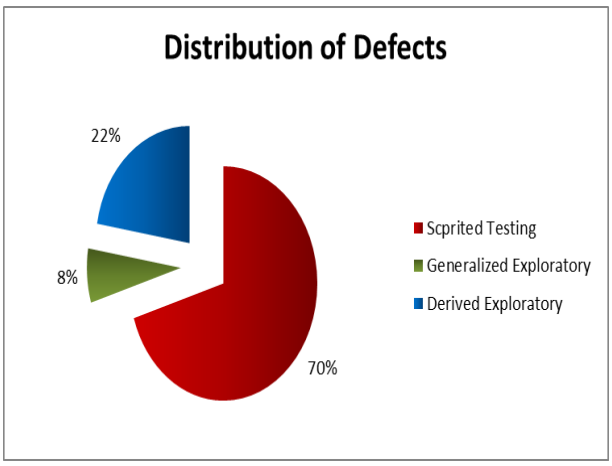

Figure (iv) – Distribution of Defects

Key Levers	With Derived Exploratory Testing	Value Additions
Test Scripting	Reduced by 50%	• Advanced strategy for bug detection • More coverage area • Easier to maintain platform dependent scripts • Reduced timelines in test scripting
TC Defects	Reduced by 80%	• Time saved in test cases fixes
Defect Detection	Increased by 25%	• Defect pattern identified easily • Early detection of bugs
Exploratory Issues	Increased by 20%	• Logged high severity issues • Increased confidence in product for the business team
Reporting	Better by 40%	• Proactive monitoring • Effective documentation for future reference

Table (i) - Derived Exploratory – Value additions

7 Conclusion

Testing is an interesting game of skill and adventure. One may choose from different avenues to reach the next stage in the game of quality and the tester's role is to break through each one of these to ensure it leads to the correct destination. Our derived approach with its information flow from one phase to the next not only helps save time but also helps uncover 20% more defects. Once you take control of the reigns to this derived game of adventure, you will be sure to hook on to it.

About Indium Software

At Indium Software, we've been entrenched in the world of software testing since 1999. We've built a team of 450+ software and test professionals in our offices in Chennai, Bengaluru, New Jersey, Sunnyvale, London and Kuala Lumpur.

The core of Indium's objective to servicing our global customers can be explained with this simple line: "We're small enough to care, large enough to deliver." We are a preferred testing vendor for enterprise and ISV customers ranging from Fortune 100 to 5000 companies and small to medium enterprises.

Till date, we've served over 250 clients in the U.S., and Rest of the World.

Contact Us

USA

SUNNYVALE
Suite 210,
1250 Oakmead Parkway
Sunnyvale, CA – 94085.
Phone: +1(408) 501-8844
Fax: +1(408) 501-8808

ATLANTA
Crown Office Suites
1870 The Exchange
Suite 100
Atlanta, GA – 30339.
Phone: +1 (770) 989-7302

PRINCETON
Carnegie Center
Suite 150,
300 Carnegie Center
Princeton, NJ – 08540.
Phone: +1 (609) 786-2423

UNITED KINGDOM

LONDON
Indium Software
71-75 Shelton Street
London, WC2H 9JQ.

INDIA

CHENNAI
No.64 (Old N.143), Eldams Road
Ganesh Chambers Teynampet,
Chennai – 600 018.
Phone: +91-44-6606 9100

BENGALURU
No.100, Kay ARR Royal Stone
Tech Park, 5th Floor, Pai layout,
Benniganahalli, Bengaluru,
Karnataka – 560016.
Phone: +91-80-4645 7777

MALAYSIA

KAULA LUMPUR
Suite 8-1 & 8-2, Level 8, Menara
CIMB, No.1, Jalan Stesen Sentral 2
Kuala Lumpur – 50470.
Phone: +60 (3) 2298 8465
Fax: +60 (3) 2298 8201

SALES INQUIRIES

americas.sales@indiumsoft.com
apac.sales@indiumsoft.com
emea.sales@indiumsoft.com
india.sales@indiumsoft.com
sales@indiumsoft.com

GENERAL ENQUIRIES

careers@indiumsoft.com
info@indiumsoft.com