The Black Swan of Software Testing
– An experience report on exploratory testing

Main Author & Presenter: Shaham Yusuf
Co-Author: Venkat Moncompu

October 2012
Exploratory testing approach is seldom considered in our test planning as it is the outlier of our scripted testing approach which we take for granted. Exploratory testing carries high impact and can exponentially increase the defect detection rate. Further, with exploratory testing, we get a chance to look at issues during retrospection which does not happen in scripted testing early, thereby reducing defects from leaking to User Acceptance Testing (UAT). For an example, in scripted approach we usually tend to stick to the pre-defined test cases whereas exploratory testing helps to look back and learn from, on-the-fly while testing.

Nassim Nicholas Taleb, the author of the book “The Black Swan: Impact of the highly improbable” says, “one single sighting of black swan can invalidate the belief that all swans are white”.

This presentation is such demonstration of the black swan of testing which might help us to shift our mindset from the granted belief of scripted testing.

- Till 17th century it was believed that all species of swans were white.
- Every bit of empirical evidence confirmed this as true.
- By the discovery of black Swan in Australia destroyed all previous notions that all swans are, and must be, white.
In an industry where prescriptive practice focused on requirements-based testing is prevalent, much attention is paid to scripted test development and verification for conformance. Some people take extra effort to provide visibility of test coverage to leadership that is sufficiently satisfactory yet products/applications continue to fall short of meeting the ‘fitness of use’ criteria of Quality. One reason is the dysfunctional relationship between a design that is system-centric and a solution that ought to be user-centric, which leads to user acceptance issues during software development or defect leakage to production.

Exploratory testing relying on continuous learning, test adaptation and execution, is an effective complementary testing approach which addresses above mentioned gap and helps to detect business critical defects quickly in the testing cycle. However, traditionalists revile at the suggestion of using exploratory testing citing reasons of coverage, test design, traceability, accountability etc. The curious gen Y testers are faced with the daunting task of winning over the skeptics to adopt the practice.

In this presentation we are sharing key factors that have helped us succeed in our efforts to convince stakeholders and to adopt exploratory testing in large scale independent testing engagements. We intend to showcase our blended approach (in other words, Alternative Test Design Techniques) where test designing was done on high level leveraging Scenarios, Mind Maps, Checklist, and Cheat Sheets. Also, the defect metrics presented would intrigue you to give exploratory testing another
chance!
Abstract

Case Study – Traditional Approach

What is the problem in traditional approach?

What do we need to address problems of traditional approach?

What is Exploratory Testing and what are the challenges?

The Solution - Alternative Test Design techniques with exploratory testing

Do it yourself and conclusion
Now look back and think about the testing approaches you have seen or you have followed in the history of your career. We tend more-or-less to follow the same old scripted or documented test case based approach and our belief is firm that it alone works, no matter what type of application we are testing. We bias this belief on the three pillars of ‘Scripted Testing’. First is the execution accountability, second is traceability and third is detailed steps having expected results.

While working on a large and complex project and following the traditional scripted testing approach, we have captured the following metrics along with some key observations.

### Case Study – Traditional Approach

**Test Design**

<table>
<thead>
<tr>
<th>Test Cases Created</th>
<th>Test Case Complexity</th>
<th>Average time spent Creation of each Test Case (hours)</th>
<th>Total time spent on Test Case Creation (hours)</th>
<th>Average time spent Review of each test case (hours)</th>
<th>Total time spent on Test Case Review (hours)</th>
<th>Total Time spent on Test Design (Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>705</td>
<td>Simple</td>
<td>1</td>
<td>706</td>
<td>0.5</td>
<td>353</td>
<td>1059</td>
</tr>
<tr>
<td>1178</td>
<td>Medium</td>
<td>2</td>
<td>2356</td>
<td>0.75</td>
<td>883.5</td>
<td>3239.5</td>
</tr>
<tr>
<td>472</td>
<td>Complex</td>
<td>3</td>
<td>1416</td>
<td>1</td>
<td>472</td>
<td>1888</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td>4478</td>
<td></td>
<td></td>
<td><strong>6186.5</strong></td>
</tr>
</tbody>
</table>

**Key Observation:**

Significant amount of project time, 6186 hours, was spent on Test Design. With the team of 12 resources (including test lead), we spent over 3 months of time in test design phase (considering 45 hours per week per resource).
Case Study – Traditional Approach (contd.)

System Integration Testing (SIT) Defects

<table>
<thead>
<tr>
<th>Test Cases Executed</th>
<th>Test Case Complexity</th>
<th>SIT Defects Detected by Test Cases</th>
<th>SIT Defects Detected on Ad-hoc basis (does not maps to a test case directly)</th>
<th>Total SIT Defects Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>706</td>
<td>Simple</td>
<td>69</td>
<td>65</td>
<td>134</td>
</tr>
<tr>
<td>1178</td>
<td>Medium</td>
<td>104</td>
<td>96</td>
<td>200</td>
</tr>
<tr>
<td>472</td>
<td>Complex</td>
<td>24</td>
<td>39</td>
<td>63</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>197</strong></td>
<td><strong>200</strong></td>
<td><strong>397</strong></td>
</tr>
</tbody>
</table>

Key Observation:

After going through robust test design phase, about 50% of the total SIT defects were detected on Ad-Hoc basis, i.e., they were tangential to the test cases.
Case Study – Traditional Approach (contd.)

SIT vs UAT Defects

<table>
<thead>
<tr>
<th>Number of SIT Defects</th>
<th>397</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of UAT Defects</td>
<td>50</td>
</tr>
</tbody>
</table>

Key Observation:

There was significant defect leakage to UAT ((50/397)*100 = 12.5%).
Abstract

Case Study – Traditional Approach

What is the problem in traditional approach?

What do we need to address problems of traditional approach?

What is Exploratory Testing and what are the challenges?

The Solution - Alternative Test Design techniques with exploratory testing

Do it yourself and conclusion
We observed that after spending more than 3 months of time in Test Design phase, we discovered lots of the defects which were valid ones but not directly or explicitly related to the test cases per se. The test cases indeed helped the team to get closer to those defects though not directly. This made us take a step back and see if our Test Design was ineffective though those test cases were reviewed and approved by domain experts!

After doing extended analysis of the designed test cases as well as analysis of those defects, we found that most of the defects being detected were tangential to the pre-designed test cases because the pre-designed test cases are mere interpretation of requirements linguistically; whereas during testing we questioned the product from end user’s expectation or perspective.

On top of the high defect leakage of 12.5% UAT, we realized we were too caught up in the perfection of test design. We were quite taken aback by the large number of so-called ‘non-test cases’ defects which led to defect leakage to UAT. Maybe, we were so confident and comfortable with the detailed approach i.e., the test case based approach that we never felt the need to think in any other manner or of a better approach!
Abstract

Case Study – Traditional Approach

What is the problem in traditional approach?

What do we need to do to address problems of traditional approach?

What is Exploratory Testing and what are the challenges?

The Solution - Alternative Test Design techniques with exploratory testing

Do it yourself and conclusion
In the course of the project many events happened: we missed important bugs, bugs were found during UAT, some in production too and we got bugs at such a later stage that we challenged our so-called robust test design because we seemed to have missed bugs though we spent huge amount of time on creating detailed test cases and up-front planning.

Though these events are something that could not have been predicted in the traditional scripted approach, human nature compels us to create explanations for the occurrence of these events making it explainable and, in retrospect, predictable. In such situations we sometimes may say, it is out of scope of our test case etc.; but the client were not satisfied!

We make 10 years projections on oil prices, economy, social events, political affairs etc., without realizing that that we cannot even predict these for next summer! All our prediction failed terribly on major unpredictable events like financial crisis of 2009 or the 9/11 attack.

On a very similar note, we design amazing test cases with explicit steps prior to the creation of the application. While executing those test cases we are strongly biased by the steps and hence concentrate on the events which are mentioned in the steps or the expected results only. What about millions of other things happening in the application during the same time? Do we focus on them? Do we de-focus from them? These would result in defect leakage. In a traditional approach (i.e., scripted testing), when a tester executes a pre-defined test case, he/she also looks into ‘a day in the
life’ type of tests. Putting the system through these conditions that is not scripted is nothing but *Exploratory Testing*. If these are practiced in a consciously structured manner then the results can be very impressive; as was in our case. The other pitfall of running scripted tests over-and-over is that these tests become ineffective over the repeated runs analogous to following in someone else’ footsteps in a minefield which has a much lower probability of actually finding anything new.
Abstract

Case Study – Traditional Approach

What is the problem in traditional approach?

What do we need to address problems of traditional approach?

What is Exploratory Testing and what are the challenges?

The Solution - Alternative Test Design techniques with exploratory testing

Do it yourself and conclusion
Exploratory testing is an approach where tester investigates and finds various information about the product, which may matter to stakeholders. The tester is not driven by the pre-defined test cases, rather has control over the test designs during execution and uses the findings to design or vector subsequent tests.

Exploratory Testing is not procedurally structured, rather it is cognitively structured, says Michel Bolton (Bolton, 2008). It is like a chess game. We revise our plans after seeing the move of the opponent (application behavior). Yet all our plans can be changed after one unpredictable move of our opponent. Can we write cases for the chess move beforehand? Even if we write the exhaustive cases (millions of possible cases) then which case to apply when depends on the context and is a cognitive judgment. Exploration is the basic nature of any human being. Human are curious by nature and are interested in investigation, examination and questioning. The dictionary defines exploration as investigation and examination of unknown regions.
Test Design:
Defect leakage metric shows continued bleeding of defects even when test coverage is reportedly 100% of stated requirements. Short-sighted test managers rely excessively only on the stated requirements. This myopic view only adds to the poor quality as requirements are one-dimensional and static whereas user needs are multi-faceted that cannot be represented to meet all those needs. Exploratory Testing gives the tester this degree of freedom to transcend this gap to look at the application in a contextual perspective beyond the stated requirements. Exploratory Testing defines structure to the practice of testing that makes it risk-focused, value-driven and effective. The metrics we have gathered show a reduction in defect escape rate that shows the value-focus of Exploratory Testing.

“Highly repeatable testing can actually minimize the chance of discovering all the important problems, for the same reason stepping in someone else’s footprints minimizes the chance of being blown up by land mine.” – James Bach

Exploratory Testing can overcome the pitfalls of minefield analogy as articulated by James Bach. In feature driven development methodology for e.g., the risk profile keeps changing and test profiles also have to reflect these changes. Through exploratory testing, we design tests dynamically based on the current context continuously as the tester gains insight and feedback from the system. These result in effective testing of the system compared to those which are designed based on requirements linguistically.
Our data demonstrates that the effectiveness of scripted tests decreases dramatically as a function of time (running same tests over and over again) across product releases. And since the adoption of Exploratory Testing in tandem with scripted testing, our data show significant improvements in the number and value of defects reported, which is also appreciated by the stakeholders.

Accountability:
In traditional or scripted testing the accountability is maintained by passing or failing every pre-defined test case. System behavior cannot be just evaluated objectively using pre-defined test cases alone.

In contrast, exploratory testing approach is more of a subjective assessment of the system where the tests are designed and executed based on sapience and learning to use the system (Bolton, 2009). One of the best ways to maintain accountability in exploratory testing is to practice session based testing (Bach, 2000). Disciplined exploratory test sessions are far more effective than a simple execution of pre-defined test case. Nowadays Exploratory sessions are also supported by many tools.

In the traditional approach, in a way testers are forced to follow a premise that ‘this is what we are going to do, irrespective of the way the application behaves at this time’. In contrast the exploratory approach implies ‘This is what we have done’. Now, we leave it on you to decide which is more accountable and effective?

Learning styles:
Scripted approach relies less on the cognitive skills of the testers in terms of continuous discovery and learning whereas Exploratory Testing relies on heuristics and effective learning techniques. The more we rely on the steps of test cases to learn about the system, the more easily we forget about the system. In contrast if we learn the system by process of exploration and discoveries, less is the chance to forget about it – because we discovered about it, steps didn’t tell us!

Negotiated Contract:
It is typical notion in industry that coverage can be shown by number of requirements pass/failed. Well, what about the requirements which are still evolving? What about missed requirements? In Exploratory Testing, we can still show coverage of the documented requirements by means our test execution. This is most favored by agile development teams who adopt just-enough and just-in-time requirements.
Abstract

Case Study – Traditional Approach

What is the problem in traditional approach?

What do we need to address problems of traditional approach?

What is Exploratory Testing and what are the challenges?

The Solution - Alternative Test Design techniques with exploratory testing

Do it yourself and conclusion
We like stories, detailed steps and documentation and we like to simplify them, i.e., *to reduce the dimension of matters!* It is because all of these look very good. There is a flip side of these though. One of the ways in which human beings are blind to the occurrence of unpredictable has to do with something called the “narrative fallacy,” a problem that illustrates our need to turn abstract concepts into understandable stories. We write detailed test cases from the abstract application i.e., the requirement documents and we expect to find all the vital defects of the application from those test cases though the context during execution may be different. In short, how it is possible to understand the properties of the unknown (the application), based on what is known theoretically (the test cases and the requirements)?

After extensive analysis, thought and brainstorming, we realized that test design has to be light weight so as to complement user expectation as well as evolving requirement specifications.

We toyed with several ideas, like,

**Writing high level test cases**

**Not writing test cases at all and just relying on session based exploratory testing**

**Using requirement documents only as our guide while testing**

**Leveraging use cases and user stories to derive ad-hoc test cases while testing**

**Stop toying and get back to the traditional approach**

Soon we realized that there could not be a single alternative; we have to come up with an approach which leverages best of all approaches and should be in-line with

---

### The Solution - Alternative Test Design techniques with exploratory testing

**The Solution:**

Leverage Exploratory Testing approach with Alternative Test Design Techniques.

**Alternative Test Design techniques :**

**Scenarios – scenario based testing complemented by test execution narration**

- We used high level test scenarios for designing complex business rule type of tests

**Flow Diagrams**

- We used flow charts for designing end-to-end type of tests

**Mind Maps**

- Mind Maps were used to jot down spontaneous ideas which flow during JAD (Joint Analysis & Design) sessions and requirement analysis phase.

**Cheat Sheets**

- We used cheat sheets for existing/old modules, primarily to regress the system.

**Checklists**

- Checklists were extensively used to design nitty-gritty types of test e.g., Date should be in GMT, Text should be Arial Black, size 14 etc.
our project – which means it should suit the context of our project.

We need something in the form of guidance heuristics for our test cases. We knew that some high level test design is required for our needs. But the nature of our project demanded mix of things. We boiled them down to the following set of light weight test design techniques,

Scenarios – scenario based testing complemented by test execution narration
- We used high level test scenarios for designing complex business rule type of tests

Flow Diagrams
- We used flow charts for designing end-to-end type of tests

Mind Maps
- Mind Maps were used to jot down spontaneous ideas which flow during JAD (Joint Analysis & Design) sessions and requirement analysis phase.

Cheat Sheets
- We used cheat sheets for existing/old modules, primarily to regress the system.

Checklists
- Checklists were extensively used to design nitty-gritty types of test e.g., Date should be in GMT, Text should be Arial Black, size 14 etc.

We have not coined these test techniques but have merely leveraged them in our project. There can be other test design techniques as well, but the above five worked well in the context and nature of our application. The key idea is that test design should be at a High-Level rather than detailed step-by-step so that testers get a freedom to explore and at the same time they are more responsible for testing as opposed to just following the steps.

All of the 5 test design techniques which we have used in our project encourages testers to document tests at a high level. They have different purpose in the context to type of test, for example; ‘Flow Diagrams’ were used for end-to-end type of tests whereas checklists were leveraged to ensure that we are missing nitty-gritty type of requirements.

We highly encourage you to choose your high level test design techniques after careful analysis in context to the nature of your project.
A scenario is a hypothetical story, used to help a person think through a complex problem or system. "Scenarios" gained popularity in military planning in the United States in the 1950's. Scenario-based planning gained wide commercial popularity after a spectacular success at Royal Dutch/Shell in the early 1970's.

A kid explores the world by tasting everything (Putting things in mouth) and learns which is what. But we sometimes instruct them what to do and what not to. We provide some direction to them so that they can be guided toward a better understanding.

It seems that Scenario based testing exists in the normal testing life cycle processes and maps Use Case to Scenario and then to Test Cases. In our approach we stop at Use case to Scenario and move towards execution.

While we do agree to this it does not adhere to the processes that are required and defined for test case based testing which are very much essential while delivering large complex projects. Here we have taken steps to make sure people endorsing quality believe in our “black swan” of testing. In this approach we have designed processes and documentation which will support answers to all stakeholders who define and endorse quality.

Here are some characteristics of an ideal scenario:

- The test is based on a story about how the application is used, including information about the motivations of the people involved.
- The story is motivating. A stakeholder with influence would push to fix a problem that fails this test.
The story is credible. It not only could happen in the real world; stakeholders would believe that something like it probably will happen.

The story involves complex use of the application or a complex environment or a complex set of data.

The test results are easy to evaluate. This is valuable for all tests, but is especially important for scenarios because they are complex.

In our case we have customized it to suit our project needs and context. We highly encourage you to customize it to suit your projects. No matter which format you follow the basic idea remains the same – to have a short story or objective of the scenario followed by a one line process and an open ended output.

In our case we refer to the objective as the story of the scenario. We make the objective motivating and credible based on the knowledge of the application and the business. We document complex situations and state transitions by matrix of combinations and other accelerators which suits the context.

Below is a typical test case having detailed steps and is followed by a typical scenario for the same test condition. The below comparison shows how a high level scenario handles the test condition in few lines whereas the test case has more than 8 detailed steps for the same situation.
We created flow diagrams using MS Visio to design the end-to-end types of tests. These were the test where we are NOT concerned about a module/functionality working in silo but multiple functionalities/modules should work in conjunction.

Mind maps are great aid to spontaneous test idea generation. The open ended nature of Mind Map allows to club ideas in logical manner. The best times to create Mind Maps are during JAD (Joint Analysis & Design) sessions or during team brainstorming sessions. We have gathered some great ideas using Mind Maps and leveraged them during testing.

They are unique to each tester as it is for any user of mind maps. It helps organize the thought process that is proven to be non-linear and therefore not friendly to the way languages are written. Mind maps provide the spatial dimension to facilitate the thought process. Ideas by association and spatial reference are the keys to assimilating thoughts which are otherwise not comprehensible. Mind maps provide just the right framework for exploration.

Depending on the nature of your project, you may create Mind Maps functionality/module wise or testing type wise.
Different types of checklist can be used depending on the nature of the project. We use a checklist which covers the explicit requirements, and we need to ‘check’ that they work in every iteration / release. We wanted to check this in every iteration as they were working before and we want to confirm it newer iteration(s). Here is a typical Checklist,

<table>
<thead>
<tr>
<th>Check List</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td></td>
</tr>
<tr>
<td>1. Project ‘A’, ‘A’ is true</td>
<td>✓</td>
</tr>
<tr>
<td>2. Test whether ‘A’ is true</td>
<td>✓</td>
</tr>
<tr>
<td>3. Test whether ‘A’ is true</td>
<td>✓</td>
</tr>
<tr>
<td>4. Test whether ‘A’ is true</td>
<td>✓</td>
</tr>
<tr>
<td>5. Test whether ‘A’ is true</td>
<td>✓</td>
</tr>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>1. Date is 2017</td>
<td>✓</td>
</tr>
<tr>
<td>2. Date is 2017</td>
<td>✓</td>
</tr>
<tr>
<td>3. Date is 2017</td>
<td>✓</td>
</tr>
<tr>
<td>4. Date is 2017</td>
<td>✓</td>
</tr>
</tbody>
</table>

General

<table>
<thead>
<tr>
<th>Check List</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Check whether a variable is defined</td>
<td>✓</td>
</tr>
<tr>
<td>4. Check whether a variable is defined</td>
<td>✓</td>
</tr>
<tr>
<td>5. Check whether a variable is defined</td>
<td>✓</td>
</tr>
<tr>
<td>6. Check whether a variable is defined</td>
<td>✓</td>
</tr>
<tr>
<td>7. Check whether a variable is defined</td>
<td>✓</td>
</tr>
</tbody>
</table>

Many people say that checklist and scripts are more or less same. But it is not. A checklist is an open ended item, whereas a script is a close ended item. You may validate an item of a checklist in any number of ways. But in scripts, you would follow the specified steps. The details can be referred in “value of Checklist by Cem Kaner” (Kaner, 2008) in which Cem Kaner describes how checklist differs from scripts.
# Case Study – Alternative Test Design Approach

## Test Design

<table>
<thead>
<tr>
<th>Test Design - Artifacts</th>
<th>Number of Test Artifacts</th>
<th>Average time spent Creation of each Test Artifact (hours)</th>
<th>Total time spent on Test Artifact Creation (hours)</th>
<th>Average time spent Review of each test case (hours)</th>
<th>Total time spent on Test Case Review (hours)</th>
<th>Total Time spent on Test Design (Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenarios</td>
<td>481</td>
<td>1.5</td>
<td>721.5</td>
<td>1</td>
<td>481</td>
<td>1202.5</td>
</tr>
<tr>
<td>Flow Diagrams</td>
<td>55</td>
<td>1.5</td>
<td>82.5</td>
<td>0.5</td>
<td>27.5</td>
<td>110</td>
</tr>
<tr>
<td>Mind Maps</td>
<td>20</td>
<td>2</td>
<td>40</td>
<td>1</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>Cheat Sheets</td>
<td>15</td>
<td>1</td>
<td>15</td>
<td>0.5</td>
<td>7.5</td>
<td>22.5</td>
</tr>
<tr>
<td>Checklists</td>
<td>27</td>
<td>1</td>
<td>27</td>
<td>0.5</td>
<td>13.5</td>
<td>40.5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>598</strong></td>
<td><strong>886</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>1435.5</strong></td>
</tr>
</tbody>
</table>

**Key Observation:**

Considerable amount of Test Design time is saved due to light weight test designing.
Case Study – Alternative Test Design Approach (contd.)

System Integration Testing (SIT) Defects

<table>
<thead>
<tr>
<th>Test Artifacts Executed</th>
<th>Test Artifacts Executed</th>
<th>Defects Detected by Test Artifacts</th>
<th>Defects Detected on Ad-hoc basis (does not map to any of the test artifact directly)</th>
<th>Total Defects Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenarios</td>
<td>481</td>
<td>289</td>
<td>11</td>
<td>300</td>
</tr>
<tr>
<td>Flow Diagrams</td>
<td>55</td>
<td>23</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>Mind Maps</td>
<td>20</td>
<td>34</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>Cheat Sheets</td>
<td>15</td>
<td>3</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>Checklists</td>
<td>27</td>
<td>52</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>598</strong></td>
<td><strong>401</strong></td>
<td><strong>26</strong></td>
<td><strong>427</strong></td>
</tr>
</tbody>
</table>

Key Observation:

Most of the detected defects were the outcome of the execution of the high level test artifacts. Only 6% of the total defects were detected on Ad-Hoc basis; thus the designed artifacts were leveraged completely and effectively.
### Case Study – Alternative Test Design Approach (contd.)

**SIT vs UAT Defects**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of SIT Defects</td>
<td>427</td>
</tr>
<tr>
<td>Number of UAT Defects</td>
<td>10</td>
</tr>
</tbody>
</table>

**Key Observation:**

Defect leakage was on 2.3% as the high level test artifacts allowed testers to explore like end users.

**Note:** Client used pre-defined test scripts for UAT prior to implementation of our alternative Test Design Techniques as well as after the implementation.
**Conclusion:**

Our findings of improved defect detection by as much as 12% on our project compares well with the gains other studies reported (Itkonen, Mantyla & Lassenius, 2007). Some other projects adopting exploratory testing that have reported savings of up to 30% in the testing budget over their scripted counterparts. In lean practice of software development one of the ways of identifying waste is to use a value mapping of all the process activities. If the same is done for testing processes, the scripted testing shows little value being added from the up-front activities of documentation and review whereas exploratory testing shortens the time to adding value to the quality. These findings provide compelling evidence that testing processes can only improve with the adoption of exploratory testing. We urge the skeptics out there to give exploratory testing a second chance!
We thank you for spending your time in reading the benefits of exploratory testing. We highly encourage you to pilot it in your organization / project and see the benefits for yourself. A few things to keep in mind which will help your organization become more successful.

Ensure that the stakeholders are well informed and know the pros and cons of the effort:

We suggest you conduct some competency development sessions for your team members to enhance their exploratory testing skills. This can be done by organizing few time bound testing sessions and giving them some open source applications to test and discuss their findings.

Proper context based test plan should be prepared and should be updated regularly based on the dynamics of the project

While creating / designing scenarios the checklist concept should be used extensively so that you do not miss any requirements of the negotiated contract

Ensure to maintain proper execution narration for the test execution accountability and share the narration to the stakeholders on daily basis along with the daily status report

Conduct some short sessions within your team to explain effective narration writing technique, prior to the execution.

As you understand the benefits of Exploratory Testing, please be aware of some of the cons too:

---

### Do it yourself and conclusion

**Some Challenges during implementation of Exploratory testing using Alternative test design techniques:**

Some of the challenges during implementation were related to management, team members and efficiency of execution. We have described below those challenges and their possible solutions.

<table>
<thead>
<tr>
<th>Leadership &amp; Management</th>
<th>Test Team</th>
<th>Effectiveness &amp; Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>- You need to convince your project stakeholders by showcasing the benefits of this approach</td>
<td>- Change in mindset of the resource is required as they would be transmitting to high level execution from an approach which was having detailed steps</td>
<td>- During initial phase effectiveness may be bit less but it would increase drastically once the process is matured</td>
</tr>
<tr>
<td>- A pilot should be conducted prior to implementation in large scale projects</td>
<td>- Team members need to sharpen exploratory testing skills by regular practice</td>
<td>- The lessons learnt and good practices can be leveraged to other projects of the organization for efficient execution of the later projects</td>
</tr>
<tr>
<td>- Design Test Strategy and Test Plan to suite the context of Scenario Based approach</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Do it yourself:
We thank you for spending your time in reading the benefits of exploratory testing. We highly encourage you to pilot it in your organization / project and see the benefits for yourself. A few things to keep in mind which will help your organization become more successful.

Ensure that the stakeholders are well informed and know the pros and cons of the effort:
Exploratory tests call for testers to be willing and open to learning from the system as opposed to requirements-based expectation from the system.

There is an expectation that testers are familiar with the business function and what the user wants to achieve with the task being implemented using the system.

Exploratory testing calls for clear oversight and definition of objectives from the test. The retrospective from the test sessions are very important to ensure these tests are effective.

The tester is expected to understand the role of each user of the system and the tasks that are needed to be performed by the system under test.
We thank you for spending your time in reading the benefits of exploratory testing. We highly encourage you to pilot it in your organization / project and see the benefits for yourself. A few things to keep in mind which will help your organization become more successful.

Ensure that the stakeholders are well informed and know the pros and cons of the effort:

We suggest you conduct some competency development sessions for your team members to enhance their exploratory testing skills. This can be done by organizing few time bound testing sessions and giving them some open source applications to test and discuss their findings.

Proper context based test plan should be prepared and should be updated regularly based on the dynamics of the project.

While creating / designing scenarios the checklist concept should be used extensively so that you do not miss any requirements of the negotiated contract.

Ensure to maintain proper execution narration for the test execution accountability and share the narration to the stakeholders on daily basis along with the daily status report.

Conduct some short sessions within your team to explain effective narration writing technique, prior to the execution.

As you understand the benefits of Exploratory Testing, please be aware of some of the cons too:

References:

The Black Swan by Nassim Nicholas Taleb, Random House, 2007
An Introduction to Scenario Testing by Cem Kaner
Scenarios: The Art of Strategic Conversation by Kees van der Heijden, Royal Dutch/Shell’s former head of scenario planning, Wiley, 1999
Value Of Checklists by Cem Kaner, CAST 2008 presentation
A tutorial in Exploratory Testing by Cem Kaner
The Nature of Exploratory Testing by Cem Kaner
Exploratory tests call for testers to be willing and open to learning from the system as opposed to requirements-based expectation from the system.

There is an expectation that testers are familiar with the business function and what the user wants to achieve with the task being implemented using the system.

Exploratory testing calls for clear oversight and definition of objectives from the test. The retrospective from the test sessions are very important to ensure these tests are effective.

The tester is expected to understand the role of each user of the system and the tasks that are needed to be performed by the system under test.