

Exploratory Testing: Evolution or Revolution?

BY SVEN SAMBAER
CTG TEST CONSULTANT

Introduction

Most companies that develop software recognize the crucial importance of a structured test approach and an efficient test organization. Many have invested heavily in test methodologies aligned with their software development practices. In the past, those methodologies usually required testers to create test scripts and test cases using a test basis such as a requirements document, in a process known as 'scripted testing' (ST).

Over time, developers began to adopt more agile, informal, and iterative methods such as eXtreme Programming, DSDM, Crystal, and Scrum. The new methods deemphasized the use of documentation and challenged the reliance by traditional test approaches such as ST on documentation as the test basis. Furthermore, critics claimed that ST led to excessive testing of easy cases, while failing to allow sufficient scope for creative testing of more interesting and complex cases.

A new approach, referred to as 'exploratory testing' (ET), was introduced by Cem Kaner in 1999 and further elaborated by James Bach in recent years.

Defenders of both the ST and ET approaches launched a debate that pitted formality against informality, procedures against freedom, uniformity against creativity, and manageability against efficiency.

This report affirms the value of ET as a technique, while arguing that it must be more than a reaction to the formality of a traditional structured test approach: ET should be an evolution, not a revolution! It proposes a hybrid ST/ET approach as the most effective option, and suggests how and when ET should be included in the testing strategy, in what situations it is most useful, and which aspects of the testing process should never be exploratory or agile.

A SPECIAL CLIENT REPORT

Copyright ©2005 Client Confidential: A special report for clients only. The material covered in this report is for guidance only. Application and implementation guidelines, advice, and consulting are available.

ET encourages the tester to be creative and flexible: to act intuitively. It is investigative rather than confirmative. The tester is free to explore new areas as test ideas arise during test execution. As a result, it significantly increases the chance of finding bugs that might have gone undetected using ST.

What is Exploratory Testing?

ET is a process all software testers know and have been using for decades. James Bach calls it “scientific thinking in real time” and defines it as:

Simultaneous learning test design and test execution; any testing to the extent that the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests.

In ET, the tester explores a piece of the system, thinks about what should be tested and how, and then carries out the appropriate tests. In the process, the tester learns about the system and its behaviour, develops new tests based on what he or she has learned so far, executes these tests, and so on. Test design and execution occur more or less concurrently, often without documentation. The process is a major departure from ST, where tests are first designed and documented and then executed at a later time, not always by the same tester.

ET encourages the tester to be creative and flexible: to act intuitively. It is investigative rather than confirmative. The tester is free to explore new areas as test ideas arise during test execution. As a result, it significantly increases the chance of finding bugs that might have gone undetected using ST.

Is Exploratory Testing a ‘Technique’ or an ‘Approach’?

Despite arguments to the contrary by ET evangelists such as James Bach, this paper defines ET as a technique rather than an approach.

An ‘approach’ must address the key components of ‘process’, ‘people’, and ‘technology.’ But ET represents nothing more or less than a technique for designing and executing tests. It does not specify how the test activities should be planned, how the test strategy is to be defined, how the test activities are to be managed and tracked, how the test organization should be set up, or which tools should be used. In fact, it covers only part of the ‘process’ aspect of testing.

Is Exploratory Testing the same as Agile Testing?

As opposed to ET, which supplies a valuable technique in support of agile testing (AT), AT itself is an approach: a fully elaborated testing philosophy that incorporates principles such as short iterations, continuous integration, extensive communication, ready acceptance of changing requirements, close involvement of business users, and concentration on developing software rather than documenting it. In agile development, “activities that don’t contribute to the end product are considered waste.” (Poppendieck & Poppendieck, 2003). Consider these differences between ET and AT:

- ET offers a useful way to test a specific application feature that can be applied to a wide range of software development projects, from agile development to more traditional code-and-fix efforts. But AT represents a comprehensive testing approach that incorporates an extensive array of features such as daily stand-up meetings, pair testing, location of both testers and developers in a team room, drop-in meetings, test-driven development, and more.

As opposed to ET, which supplies a valuable technique in support of agile testing (AT), AT itself is an approach: a fully elaborated testing philosophy that incorporates principles such as short iterations, continuous integration, extensive communication, ready acceptance of changing requirements, close involvement of business users, and concentration on developing software rather than documenting it.

- Unlike ET, AT is always iterative, and can also be used as a technique for testing specific features in traditional ‘big-bang’, waterfall-like projects.
- ET requires very little documentation prior to test execution. In AT projects, this is not always true. In test-driven eXtreme programming, for example, acceptance tests are documented prior to development that summarize preliminary discussions and decisions and serve as basic requirements documents for the development team. As Crispin notes, “Acceptance tests are designed to tell us when we’ve successfully completed the iterations functionality.”
- Finally, the AT community promotes the use of test automation (for regression testing), while the freedom and creativity characteristic of ET make it an unsuitable technique for test automation.

Making Exploratory Testing Manageable

Although ET is by no means synonymous with ‘unstructured’ and ‘unplanned’, some test managers complained that it was not an adequately transparent or manageable process. They found it difficult to monitor their teams’ activities or to accurately gauge the status of the test object. In response, test managers have developed techniques to gain better control of the ET process and explain it more clearly to project stakeholders.

The resulting variations on ET ranged from informal ‘freestyle ET’, where the tester produces only bug reports, to the more formal ‘chartered ET’ where the tester works according to a specific test assignment, but with no designated procedure. In another variation, ‘session-based test management’, the test effort is divided into manageable time segments.

Confronted with the limitations of freestyle ET, greater structure and manageability were sought, resulting in methods like those below:

- *Sessions*: 90-minute time boxes for ET
- *Charters*: A clear mission for the session describing what to test, how to test, what bugs to look for, what risks are involved, and what documents to examine
- *Session sheets*: Reviewable results of a session, including notes, bugs, issues, and basic metrics such as time spent on set-up, test execution, and bug reporting
- *Session logs*: Hansel and Gretel-like ‘breadcrumb trails’ used during test execution
- *Debriefings*: Meetings with the test manager
- *Dashboards* for reporting purposes

Variations on ET ranged from informal 'freestyle ET', where the tester produces only bug reports, to the more formal 'chartered ET' where the tester works according to a specific test assignment, but with no designated procedure. In another variation, 'session-based test management', the test effort is divided into manageable time segments.

Exploratory Testing: Pro and Con

ET is clearly a valuable component of a modern test approach. Among its many important advantages:

- It encourages creativity.
- It increases the chance of finding new and otherwise undetectable bugs.
- It allows more time for testing interesting and complex cases.
- Its greater efficiency enables the tester to find more bugs in a shorter time period and make a faster assessment.
- It demonstrates an application's ease of use.
- It is adaptable and flexible.
- It's more fun than ST.

At the same time, ET also has disadvantages that must be taken into account:

- Its limited manageability makes it difficult to coordinate.
- It provides only limited project intelligence, with little insight into risks, test coverage, or test depth.
- It offers limited test reusability or reproducibility of failures.
- It depends heavily on the testing skills and domain knowledge of the tester.
- When combined with ST, it involves a risk of redundant tests.
- It provides low accountability, making it unsuitable for a regulatory compliance context.
- It does not provide absolute assurance that the most important bugs have been found.
- It is inappropriate for security testing, performance testing, or some other highly specialized test types.
- It is not suitable for tests that are complex to set up or those with a slow feedback loop, such as batch programs running at night.
- It can start only when the test object is available—that is, situated directly on the critical path of the project.

When to Apply Exploratory Testing

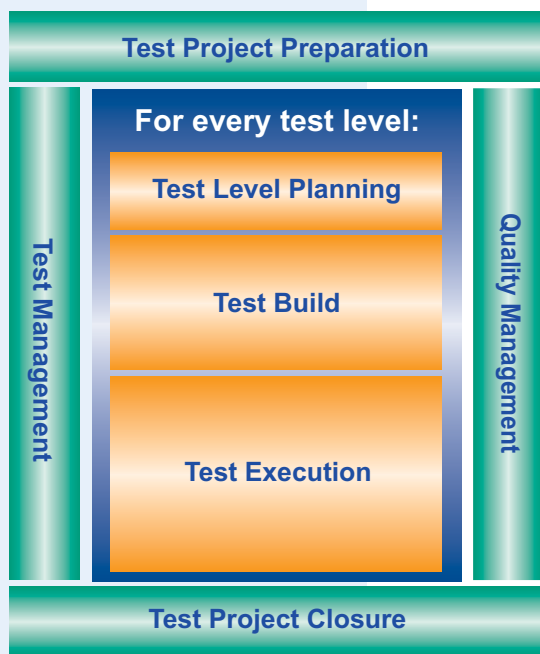
Taking ET's advantages and disadvantages into account, there are some situations where it is ideally useful. While the following list is not exhaustive, it suggests several ideas as to where ET can best be incorporated into the overall test approach:

- *A new tester enters the team:* ET can make the learning phase an active testing and exploration experience.
- *A quick assessment is needed:* ET offers fast insight into product quality in the short term when there is no time for test preparation.

A test plan describes the 'what', 'where', 'when', 'who', and 'how' of testing in a project. A key element of the test plan is the test strategy. The test strategy ensures that the test effort is correctly focused, and that the best use is being made of the available resources.

- *New information comes to light during the execution of scripted tests:* The new information might suggest another test strategy that would warrant switching to an exploratory mode.
- *You want to validate the work of another tester:* ET lets you explore the feature that he/she has tested.
- *Your team includes testers with high domain knowledge:* Those individuals can be trusted to perform effective testing using ET.
- *An intake test (a.k.a. smoke test) is desired:* ET lets you determine the maturity and stability of the test object before progressing to large-scale test execution.
- *There is no test basis:* ET is useful when there is no documentation or other sources that can define clear, expected results for the tests.
- *It's an agile project* (eXtreme Programming, DSDM, Crystal, Scrum, XBreed, etc.)
- *You want to isolate and investigate a particular defect.* (Bach)
- *You want to determine the status of a particular risk* in order to evaluate the need for ST in that area. (Bach)
- *It's an early iteration* in which the product is not stable enough for ST.
- *It's a beta test*, where users are invited to provide early feedback on a prototype or a preliminary test version.
- *You want to augment ST to diversify testing:* The combination of ST and ET is appropriate for testing features with a high risk and/or priority profile.

How to include ET in your Testing Project



A typical, straightforward testing process features these steps:

- The test project starts with *preparation and planning* activities, where the approach and test strategy are defined, the test planning is specified, and test plans are written.
- In the *test build* phase, the test specifications are documented and the test infrastructure is set up.
- The tests that were prepared in the previous step are run in the *test execution* phase, which usually consists of multiple test cycles.
- At the end of the project, *test project closure* takes place: the test process is evaluated, the test deliverables are consolidated, and the test team is released from the assignment.
- In parallel with these activities, *test management* ensures that the test process proceeds in a professional way using accepted management tools such as follow-up, status reporting, and defect management.
- *Quality management* is performed to manage, guide, and improve project execution and assure the quality of the deliverables.

It's important to note that test management activities—preparation, planning, management, and closure—cannot be conducted in an exploratory way. These activities need the support of traditional techniques such as documenting test plans, test strategies, test status, and test summary reports.

The figure on the preceding page illustrates the typical test process. The following sections describe ET's optimal role in various testing phases.

Test Project Preparation and Planning

| ID | Prio | Features To Test |
|-----------|------|---------------------------|
| 1 | | FUNCTIONAL TESTING |
| 1.1 | | NOTEPAD ENGLISH |
| 1.1.1 | | MENU FEATURES |
| 1.1.1.1 | | FILE |
| 1.1.1.1.1 | M | NEW |
| 1.1.1.1.2 | H | OPEN |
| 1.1.1.1.3 | M | SAVE |
| 1.1.1.1.4 | L | SAVE AS |
| 1.1.1.1.5 | M | PAGE SETUP |
| 1.1.1.1.6 | H | PRINT |
| 1.1.1.1.7 | M | EXIT |
| 1.1.1.2 | | EDIT |
| 1.1.1.2.1 | L | UNDO |
| 1.1.1.2.2 | M | CUT / COPY / PASTE |
| 1.1.1.2.3 | H | DELETE |
| 1.1.1.2.4 | M | FIND / FIND NEXT |
| 1.1.1.2.5 | L | REPLACE |
| 1.1.1.2.6 | L | GO TO |
| 1.1.1.2.7 | L | SELECT ALL |
| 1.1.1.2.8 | L | TIME DATE |
| 1.1.1.3 | | Etc. |

A test plan describes the ‘what’, ‘where’, ‘when’, ‘who’, and ‘how’ of testing in a project. A key element of the test plan is the test strategy. The test strategy ensures that the test effort is correctly focused, and that the best use is being made of the available resources.

As required by CTG's STBox™ methodology, one of the best ways to define a test strategy is to complete a ‘Features to Test’ (FTT) list, like the one at left. The FTT list provides an overview of the product risks, the FTTs themselves, and the priority/risk of each FTT.

The priority and the characteristics of each FTT indicate which technique(s) should be used to test that particular feature. Low priority/risk features or features that are poorly documented are good candidates for ET. High priority/risk features should be covered to a greater or lesser degree by some form of ST.

Bach says that “Most situations benefit from a mix of scripted and exploratory testing”, an opinion I strongly second. In fact, I'd advise test managers to apply ET to every FTT in their

test strategies. Even for FTTs that will undergo intensive scripted and formal testing (such as the ‘Print’ functionality in the our example), I'd prescribe, for example, “four hours of additional ET” to increase the chance of finding bugs in that area. Some test managers I've encountered even reserve a fixed day each week for their teams to perform ET (e.g., ‘Friday is ET day’).

Once it's clear what needs to be tested and how, the test manager can start planning the test activities. As Bach points out, it should be clear that ET is a planned activity, even though it's not scripted in detail. Lee Copeland has written about what he calls “exploratory planning”: that is, planning as much as is practicable, learning from the planning and from the project, and then adjusting the planning based on the findings. While I don't necessarily agree with Copeland's terminology, I do believe that planning is always a dynamic process consisting of planning, re-planning, revised re-planning, and so on. Even older and more traditional project management methodologies recognize the value of that cycle.

It's important to note that test management activities—preparation, planning, management, and closure—cannot be conducted in an exploratory way. These activities need the support of traditional techniques such as documenting test plans, test strategies, test status, and test summary reports.

Test Build and Execution

As noted earlier, ET combines test building and test execution as simultaneous activities.

The use of an FTT list is consistent with the concept of *charter-based ET*, as invented by Bach, although what Bach calls “charters”, we call “FTT’s”. Charters also have a unique ID, a risk level, a title, and a short description. Once the test manager specifies the charters, the testers work independently to build and execute the tests to fulfill the charters, and then report back to the manager in either written or oral form. The test manager should use an instrument such as an FTT list to steer the testing process. Therefore, ET should always be at least charter-based.

When the exploratory test effort is divided into test sessions (e.g., 90-minute time boxes), ET is taken a step further, to *session-based test management* (also introduced by Bach). The charters then describe in one or two sentences how testers will spend the next 90 minutes of testing. A typical exploratory test session looks like this:

- The tester reads the charter (or FTT list).
- The tester tries to find out more about the FTT, either through documentation or talking to people, and writes down some test ideas. A good set of testing heuristics can help the testers to generate ideas.
- The tester begins executing the tests and exploring the test object. As new test ideas continually emerge during test execution, the tester is free to be distracted and to zoom in on areas of doubt or instability.
- When an exact description of the expected result is not available, Oracle heuristics, such as Bach’s “HICCUPP” may be helpful. HICCUPP indicates that exploratory testers need to check whether the test object is consistent with past behavior (**H**istory), with the **I**mage of the company and the product, with **C**omparable products, with the **C**laims that have been made for it, with **U**sers’ expectations, with the functions within the **P**roduct, and with its **P**urpose.
- Bugs or suspected bugs must be entered into a bug tracking system. Bug administration is the only clerical activity that is absolutely mandatory in basic ET.
- At the end of a session, the tester provides feedback to the test manager, either in a debriefing or by means of a session sheet.

Some test managers organize ‘ET in pairs’. This tactic has several advantages. For instance, it allows one person to test while the others make creative suggestions and register bugs, and it enables team members to learn from each other’s insights.

“A project is like driving at night on a dark road. Testing provides the headlights for the journey.” Translating the feedback from exploratory testers into objective project intelligence is the art that every test manager must master.

Test Management and Test Project Closure

The real added value of testing is the generation of what P. Gerrard has called “project intelligence”. Project intelligence is the information provided to project managers that allows them to make informed, rational decisions (e.g., “What are the risks if we release today?”) Gerrard used this simile: “A project is like driving at night on a dark road. Testing provides the headlights for the journey.” Translating the feedback from exploratory testers into objective project intelligence is the art that every test manager must master. The resulting project intelligence and/or test status should be delivered via online, simple-to-read dashboards.

An important element of the project closure phase is the test evaluation report, which assesses the test project and provides clear advice from the test manager to project management.

Conclusion

ET is now recognized as a valuable technique in modern test approaches, representing an evolution towards more complete and efficient testing. The most effective test strategies are a combination of ST and ET.

While ET is a useful tool, it does not provide a ‘free pass’ to unstructured testing practices. Some activities should never have an exploratory character, and should always be formal and structured in nature. Test strategy, test planning, and all other test management activities must be extremely well structured, because they serve as the basis for delivering high-quality project intelligence to management. And as projects become more agile and exploratory, these activities—and hence the test manager’s job—become even more crucial.

Resources

- James Bach, 2003, “Tutorial Exploratory Testing,” Eurostar 2003
- James Bach, 2003, “Heuristic Test Strategy Model”
- Stefan Steurs, “Exploratory Testing – De kip of het Ei,” TI-KVIV 2004
- Tim Koomen, 2005, “ET – Knuffelen of knevelen?”
- James Bach, 2003, “Scripted versus Exploratory Testing,” Eurostar 2003
- James Bach, 2001, “What is Exploratory Testing”
- Tinkham & Caner, “Exploring Exploratory Testing”
- James Bach, 2003, “Exploratory Testing Explained”
- James Bach, 2003, “Inside the mind of an Exploratory Tester”
- James Bach, 2001, “Where does Exploratory Testing fit?”
- James Bach, 2001, “Exploratory Testing and the Planning Myth”
- Lee Copeland, 2001, “Exploratory Planning”
- Elisabeth Hendrickson, 2004, “Agility for Testers”
- Rex Black, 2005, “The importance of the Right Technique”
- William Rollinson, 2005, “Face-Off: Structured vs Exploratory Testing and Error”
- Andrew Thompson, 2005, “How to Choose between Exploratory and Scripted Testing”
- James Lyndsay, 2004, “Getting a grip on Exploratory Testing”
- Jon Bach, 2004, “Testing Sessions: Making Exploratory Testing Accountable”
- Lisa Crispin, 2001, “eXtreme Rules of the Road”
- Brian Marick, 2003, “Testing in the Agile World”
- Brian Marick, “Agile Methods and Agile Testing”
- Tim Koomen, 2004, “Exploratory Testing & TMap”
- Kaner, Bach, Pettichord, *Lessons Learned in Software Testing*, ISBN 0-471-08112-4

Backed by 39 years' experience, CTG provides IT staffing, application management outsourcing, consulting, and software development and integration solutions to help clients focus on their core businesses and use IT as a competitive advantage to excel in their markets. CTG combines in-depth understanding of our clients' business with a full range of integrated services and proprietary methodologies supported by an ISO 9001:2000-certified management system. Our 3,500 IT professionals, based in an international network of offices in North America and Europe, have a proven track record of delivering solutions that work.

More information about CTG is available on the web at www.ctg.com.