Dynamics of Exploratory Testing

(This paper supports Jon Bach's presentation at PNSQC 2006 titled "Exploratory Testing as Competitive Sport.")

Abstract:

Two decades after the term "exploratory testing" was invented, many people don't understand it. Some even call it "monkey testing" -- clicking around the screen with only a primal intelligence, hoping to stumble on to a bug. They decry its efficacy because they say the testing steps aren't precise or repeatable.

And no wonder … to the untrained eye, exploratory testing may not look very understandable or reproducible, especially in relation to scripted tests like test cases. It may look like the sport of curling, for example, where a person at one end of an ice rink slides a rock to the other end, hoping to hit the opponent's rocks in the process. To the untrained eye, curling may look unskilled, and that's at the root of many misunderstandings of exploratory testing. But it's not a question of how it looks, but how skill is cultivated and used.

In this paper, I'll discuss an emerging set of tactics and skills that when understood, can be used to narrate a tester's exploration, just as sports commentators provide play-by-play and color commentary during curling matches. With this evolving language, testers can explain their own exploration so it can be seen not as "monkey testing", but as the thoughtful, "brain-engaged" style of testing that keeps it such a widely used approach, just as Olympic sports – yes, even curling – are a showcase for all kinds of athletic skill.

Bio:

Jon Bach is Manager for Corporate Intellect and Technical Solutions at Quardev Laboratories – an onshore, outsource test lab in Seattle, Washington.

In his ten-year testing career, Jon has worked from contractor to full-time test manager to consultant for many large companies such as Microsoft and Hewlett-Packard. He has written testing articles and presented testing talks, tutorials, and keynotes at over 40 different venues, both domestically and abroad.

In 2000, Jon and his brother James invented "Session-Based Test Management" – a technique for managing (and measuring) exploratory testing. He currently serves as the VP of Conferences for the Association for Software Testing, speaker chairman for Washington Software Alliance's Quality Assurance SIG, and as a regular guest speaker about agile test techniques for Seattle-area testing forums and university classes.

Exploratory Testing, defined

The difference between scripted testing and exploratory testing is as simple as describing the game “20 Questions.” In this game, one person thinks of an object (an animal, vegetable, or mineral) and another person is allowed 20 yes or no questions to discern what the object might be. If the guesser can’t determine what the object is by the 20th question, they lose the game.

The important aspect of the game is that the guesser asks one question at a time, and an answer is given before the guesser decides what their next question will be.

This is exploratory testing in action. The game would not work if a scripted testing paradigm was used, where the guesser had to submit their 20 questions in advance and could not adapt their questions to each answer.

Unlike scripted testing, exploratory testing *relies* on this adaptation. Like the “20 Questions” guesser, the tester is free to design and execute their next test idea based on the results of the previous test. To test wisely, the tester can draw from anything in their cognition – experience, heuristics, biases, observations, conjectures, notions, and hunches, and on and on – but culling through all of that cognition, choosing that next test, playing that hunch and applying that experience – takes skill.

But often, managers treat exploratory testing as if it requires none.

Over 20 years have passed since Cem Kaner (author of the best-selling book "Testing Computer Software") coined the term "exploratory testing."[1] In that time, only a handful of practitioners are considered experts in the approach, even though it's arguably the most widely used testing style in the software testing industry.

Could that be because so many people think they already understand exploratory testing?

Could it be that managers consider it something any untrained tester could do?

Perhaps it's because so many think exploratory testing can't be measured or managed like test cases can. They may associate exploration with randomness, or mindless pounding on the keys, or may have resigned to the fact that, yes, it takes cognition (because even experts have said that exploratory testing involves tester intuition) – but cognition is a mysterious, inexplicable notion that's impossible to teach, so what’s the point of trying?

This paper was written to say that there IS a point in trying. It assumes you want to study what it is about exploratory testing that makes it so inexplicable. It assumes you have tried exploratory testing and found some success with its ability as an approach that helps you find bugs in a way that written test cases cannot. Maybe some of us have seen a tester find severe bugs without a test script and have stood back in awe of how they used their cognition. Maybe that tester was us, but we couldn’t explain how or what we did.

But mainly, this paper is meant to help you set aside a few minutes to think about why exploratory testing works, why it's such a popular approach, and how you might find words to describe the phenomenon of unscripted, unrehearsed testing.

Exploratory testing is like a driving a car – you need to be able to see and react to changing situations all around your vehicle, not just one category of situation. For example, if you wore a neck brace and could only look to the right, it would impair your driving ability.

Analogies like this are useful to help dispel the notion that exploratory testing can be done by any unskilled tester. But to propel critical discussions of what it takes to be a *good* exploratory tester (just like there are good drivers and bad drivers), there have been focused efforts like James Bach's Heuristic and Exploratory Techniques workshops (WHET)[2], James Lyndsay's London Exploratory Workshops on Testing (LEWT)[3], and most recently, the Exploratory Testing Research Summit (ExTRS)[4] in February 2006.

In his recent keynote at the First Annual Conference for the Association for Software Testing (June 2006)[5], Kaner stated a definition of software testing:

"…an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test."[6]

As he focused on *exploratory testing*, he delivered a definition in one slide consisting of ellipses – perhaps a deliberate attempt to emphasize the key components of the richly cognitive approach:

"Exploratory testing is

… a style of software testing
… that emphasizes personal freedom and responsibility
… of the tester
… to continually optimize the value of their work
… by treating test-related learning, test design, and test execution
… as mutually supportive activities
… that run in parallel throughout the project."[7]

I understood why Kaner would take such care with his definition. As a practitioner and trainer of exploratory testers since 2000[8], I know the importance of slowing down and emphasizing exploratory testing dynamics. I have even struggled with how to describe it – as did my colleagues at the Exploratory Testing Research Summit.

So, what about a demonstration to help people understand?

For the Software Testing and Review (STAR) Conference in 2005[9], my brother James (perhaps the most famous exploratory testing expert in our industry) and I were asked to present a demonstration of exploratory testing in action.

In preparing our presentation, we wanted to distill our demo into separate behaviors we could show in a two-hour timeframe. As we sat to prepare our strategy, we were creating the beginnings of a set of skills and tactics to help testers not just learn how to do exploratory testing, but do it well. We were creating a vocabulary – a language to help testers explain it their exploration to stakeholders, just as we had done on many projects.

Our aims were:

- to give testers a language to explain their work to project stakeholders
- to suggest a set of behaviors that can help them be more effective explorers
- to help testers understand how much cognition is involved in something that seems to be so taken for granted
- to dispel the notion that exploratory testing is mysterious and unstructured

That last point is a good place to start.

Is it mysterious?

No more mysterious than understanding the Olympic sport of curling. To understand the game (and ultimately to appreciate it), you need to know what to look for. It's more than just one person throwing a rock down a sheet of ice in an attempt to hit opponent rocks. It is a game of skill and strategy, with its own vocabulary. And hearing the commentators from this past Olympics, I found that I was impressed at how boring a sport could look but how interesting it was once hearing its language described in play-by-play and color commentary.

Consider the following a vocabulary that a play-by-play or color commentator would use if they were seeing you test and describing it to spectators or a list of drills to practice as if testers were in an exploratory testing pre-season training camp. I also hope it will take away the mystery of exploratory testing, and to give it a structure you can understand.

These skills comprise professional exploration of technology: [10]

Chartering: Making your own decisions about what you will work on and how you will work. Understanding your client's needs, the problems you must solve, and assuring all stakeholders that your work is on target.

Manipulating: Making and managing contact with the object of your study; configuring and interacting with it. Designing experiments and establishing lab procedures.

Observing: Gathering empirical data about the object of your study; collecting different kinds of data, or data about different aspects of the object. Designing experiments and establishing lab procedures.

Modeling: Composing, describing, and working with mental models of the things you are exploring; identifying dimensions, variables, and dynamics that are relevant.

Conjecturing: Considering possibilities and probabilities. Considering multiple, incompatible explanations that account for the same facts.

Questioning: Identifying missing information, conceiving of questions, and asking questions in a way that elicits the information that you seek.

Recording: Preserving information about your process, progress, and findings; taking notes, categorizing, sorting.

Reporting: Making a credible, professional report of your work to your clients in oral and written form.

Resourcing: Obtaining tools and information to support your effort; exploring sources of such tools and information; getting people to help you.

Refocusing: Managing the scope and depth and nature of your attention; looking at different things, looking for different things, in different ways.

Branching/Backtracking: Allowing yourself to be productively distracted from one course of action in order to explore a new idea that arises in the moment. Identifying opportunities and pursuing them without losing track of the process.

Generating/Elaborating: Working quickly in a manner good enough for the circumstances. Revisiting the solution later to extend, refine, refactor or correct it.

Roughing/Refining: Working quickly in a manner good enough for the circumstances. Revisiting the solution later to extend, elaborate, refactor or correct it.

Alternating: Switching among or contrasting different activities or perspectives so as to create or relieve productive tension and make faster progress.

Exploratory Testing Polarities

To develop ideas or search a complex space quickly yet thoroughly, not only must you look at the world from many points of view and perform many kinds of activities (which may be dualities), but your mind may get sharper from the very act of switching from one kind of activity to another.
Here is a partial list of polarities:

Warming up vs. cruising (or cooling down)
Doing vs. describing
Careful vs. quick
Data gathering vs. data analysis
Working with the product vs. reading about the product
Working with the product vs. working with the developer
Product vs. project
Solo work vs. team effort
Your ideas vs. others' ideas
Lab conditions vs. field conditions
Current version vs. old versions
Feature vs. feature
Requirement vs. requirement
Test design vs. execution
Testing vs. touring
Individual tests vs. general lab procedures and infrastructure
Testing vs. resting

Evolving Work Products

Exploratory testing spirals upward toward a complete and professional set of test artifacts. Look for any of the following to be created or refined during an exploratory test session:

Test Ideas: Tests, test cases, test procedures, or fragments thereof.
Testability Ideas: How can the product be made easier to test?
Bugs: Anything about the product that threatens its value.
Risks: Any potential areas of bugginess or types of bug.
Issues: Any questions regarding the test project, or matters to be escalated.
Test Coverage Outline: Aspects of the product we might want to test.
Test Data: Any data developed for use in tests.
Test Tools: Any tools acquired or developed to aid testing.
Test Strategy: The set of ideas that guide our test design.
Test Infrastructure and Lab Procedures: General practices or systems that provide a basis for excellent testing.
Test Estimation: Ideas about what we need and how much time we need.
Test Process Assessment: Our own assessment of the quality of our test process.
Testing Narrative: The story of our testing so far.
Tester: Experience and skill set evolves over the course of the project.
Test Team: The test team gets better, too. Relationships evolve and mature.
Developer Relations: As you test, you also get to know the developer.

Testing Considerations

This is a compressed version of the Satisfice Heuristic Test Strategy model[11]. It's a set of considerations designed to help you test robustly or evaluate someone else's testing.

Project Environment

Customers: Anyone who is a client of the test project.
Information: Information about the product or project that is needed for testing.
Developer Relations: How you get along with the programmers.
Test Team: Anyone who will perform or support testing.
Equipment & Tools: Hardware, software, or documents required to administer testing.
Schedules: The sequence, duration, and synchronization of project events.
Test Items: The product to be tested.
Deliverables: The observable products of the test project.

Product Elements

Structure: Everything that comprises the physical product.
Functions: Everything that the product does.
Data: Everything that the product processes.
Platform: Everything on which the product depends (and that is outside your project).
Operations: How the product will be used.

Quality Criteria Categories

Capability: Can it perform the required functions?
Reliability: Will it work well and resist failure in all required situations?
Usability: How easy is it for a real user to use the product?
Scalability: How well does the deployment of the product scale up or down?
Performance: How speedy and responsive is it?
Installability: How easily can it be installed onto it target platform?
Compatibility: How well does it work with external components & configurations?
Supportability: How economical will it be to provide support to users of the product?
Testability: How effectively can the product be tested?
Maintainability: How economical is it to build, fix or enhance the product?
Portability: How economical will it be to port or reuse the technology elsewhere?
Localizability: How economical will it be to publish the product in another language?

General Test Techniques

Function Testing: Test what it can do.
Domain Testing: Divide and conquer the data.
Stress Testing: Overwhelm the product.
Flow Testing: Do one thing after another.
Scenario Testing: Test to a compelling story.
Claims Testing: Verify every claim.
User Testing: Involve the users.
Risk Testing: Imagine a problem, then find it.
Automatic Testing: Write a program to generate and run a zillion tests.

Conclusion

If testers want to win respect and credibility as they perform exploratory testing, they have to be able to describe their work to a stakeholder's satisfaction. They have to be able to use a language to define and report their "off-road" testing. But they also have to be committed to using this language as a foundation to cultivate their skill as effective explorers.

Perhaps if exploratory testing was viewed as a spectator sport that could be practiced like Spring Training to prepare for the Major League Baseball season or Pre-Season Camp to prepare for the National Football League season, it could be commented on (literally) as it unfolded and stakeholders would see exploration not as unskilled "monkey testing", but *"… a style of software testing that emphasizes personal freedom and responsibility of the tester to continually optimize the value of their work by treating test-related learning, test design, and test execution as mutually supportive activities that run in parallel throughout the project."*[12]

Bibliography

[1] Cem Kaner stated his definition of exploratory testing in a keynote speech at the first annual conference of the Association for Software Testing in June 2006. See http://www.associationforsoftwaretesting.org

[2] http://serl.cs.colorado.edu/~serl/seworld/database/5899.html

[3] http://www.workroom-productions.com/LEWT.html

[4] The Exploratory Testing Research Summit was a meeting of speakers and thinkers on exploratory testing in Melbourne, Florida on January 30, 2006.

[5] See http://www.associationforsoftwaretesting.org

[6] Kaner, keynote, CAST 2006

[7] Kaner, keynote, CAST 2006

[8] Presentation: “Measuring Ad Hoc Testing”, Software Testing Analysis and Review Conference (West) 2000

[9] Presentation: “Testing Outside the Bachs”, Software Testing Analysis and Review Conference (West) 2005, http://www.sqe.com/archive/sw2005/sessions2.html?from=glance&dg=date&dgd=thu&ac=t4#T4

[10] From a handout titled “Exploratory Testing Dynamics”, first published and presented at STARWest 2005 by James and Jonathan Bach

[11] See http://www.satisfice.com/tools/satisfice-tsm-4p.pdf

[12] Kaner, keynote, CAST 2006