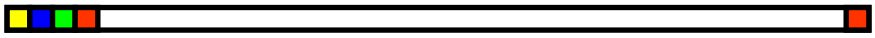
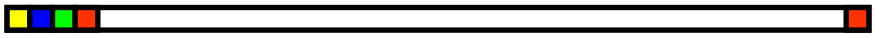


Exploratory Testing: The State of the Art

Michael Bolton
DevelopSense
STeP-IN Conference
Bengalooru, India
January 2007

Who I Am

Michael Bolton
DevelopSense, Toronto
Canada
mb@developsense.com
(416) 992-8378
http://www.developsense.com



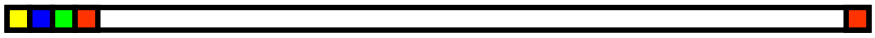
Special Thanks

Thanks to
STeP-IN Forum
Nazreen Vakharia, Coordinator
Arun Ramu, President
and all of the STeP-IN Forum Staff and Sponsors

Exploratory Testing

- What is it?
  - Some suggestions
  - A couple of experience reports
- How do we do it?
  - A few exercises
- How do we do it *better*?
  - A survey of some E.T. dynamics
- How do we make it accountable and repeatable?
  - A look at session-based test management

What is Testing?



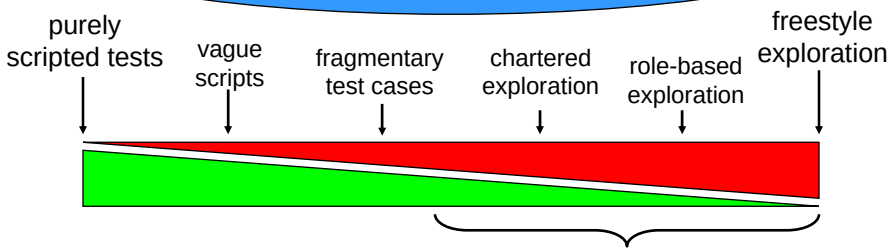
- Questioning a product in order to evaluate it
  - James Bach
- Empirical, technical investigation of the product, done on behalf of stakeholders, intended to reveal quality-related information of the kind that they seek.
  - Cem Kaner

What is *Exploratory* Testing?

- As a term…
  - apparently coined by Cem Kaner in 1983
- As a practice
  - performed by capable testers forever
  - described and elaborated by Cem Kaner, James Bach, and others
  - widely misunderstood and sometimes foolishly disparaged

What is *Exploratory* Testing?

Exploratory testing is simultaneous *learning*, *test design*, and *test execution*.



When we say "exploratory testing" and don't qualify it, we mean anything on the exploratory side of this continuum.

Exploratory Testing

- **The way we practice and teach it…**
  - **IS NOT** "random testing" (or sloppy, or slapdash)
  - **IS NOT** "unstructured testing"
  - **IS NOT** a *technique*, but an *approach*
  - **IS** "ad hoc", in the dictionary sense, "to the purpose"
  - **IS** teachable, repeatable, and manageable
  - **IS** at the heart of excellent testing
  - **IS** the opposite of scripted testing

Contrasting Approaches

Scripted Testing

- Is directed from elsewhere
- Is determined in advance
- Is about confirmation
- Is about controlling tests
- Emphasizes predictability
- Emphasizes decidability
- Like making a speech
- Like playing from a score

Exploratory Testing

- Is directed from within
- Is determined in the moment
- Is about investigation
- Is about improving test design
- Emphasizes adaptability
- Emphasizes learning
- Like having a conversation
- Like playing in a jam session

The E.T. Research Summit, 2006

Participants


James Bach


Jonathan Bach


Scott Barber


Michael Bolton


Elisabeth Hendrickson


Cem Kaner


Mike Kelly


Jonathan Kohl


James Lyndsay


Robert Sabourin

Key Points from The Exploratory Testing Research Summit

- E.T. is a style of testing in which you explore the software while simultaneously designing and executing tests, using feedback from the last test to inform the next. When I offered that definition to an XP programmer recently, he quipped, "It's Test Driven Testing!"
  - Elisabeth Hendrickson
- "Learning" is really key when I think about ET…it's starting to feel like a specialized subset of Exploratory Learning. Test design and execution are things I do when I am learning just about anything, and have the ability to experiment with the subject matter. I am now finding that I apply ET thinking in other areas beyond testing.
  - Jonathan Kohl

Key Points from The Exploratory Testing Research Summit

- Testing can be a little exploratory or very exploratory. The opposite of exploratory testing is testing where the learning, design, and execution are partitioned from each other and do not interact.
  - James Bach
- My definition of testing is "technical investigation of a product, on behalf of stakeholders, with the objective of exposing quality-related information of the kind they seek". This definition is inherently exploratory. My core definition is "brain-engaged testing". My public definition is simultaneous learning, design and execution, *with an emphasis on learning*.
  - Cem Kaner

Workshop on Heuristic and Exploratory Testing, May 2006

Participants

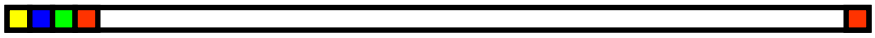
 James Bach
 Jonathan Bach
 Scott Barber
 Michael Bolton
 Tim Coulter
 Rebecca Fiedler

 David Gilbert
 Marianne Guntow
 Cem Kaner
 James Lyndsay
 Robert Sabourin
 Adam White

Workshop on Heuristic and Exploratory Testing, May 2006

- At this workshop, we did some more definitional work, and performed some task analyses
- Some maintained that, for exploratory testing, we need an artifact to test
- Others held that any activity that we perform where the intent is to inform exploration counts as E.T.
  - I think this is a disagreement between *test activities* (which include learning) and *performing a test* (execution and evaluation)
  - It may also reflect a focus on “simultaneous”, which prompted a suggestion to say “parallel” instead

Conclusions from the Workshops



- (These are personal views)
- Cem's emphasis on learning was provocative
- Exploration *is of the essence* of learning
  - cf. Play as Exploratory Learning
- Exploratory testing principally serves the investigative goals of software testing
- Confirmatory goals are important, but testing that is solely confirmatory is likely to be weak
- Exploratory testing is a critical means of checking and improving our test design
- Developing insight into our cognitive processes is key to improving E.T.

What is Exploratory Testing?

- I follow (and to some degree contributed to) Kaner's definition, which was refined in the peer conferences:

Exploratory software testing

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

See Kaner, "Exploratory Testing After 23 Years", www.kaner.com/pdfs/ETat23.pdf

How Can We Justify Exploratory Testing?

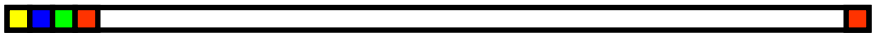
Start by recognizing that competent testers do exploratory testing all the time.

Paradigmatic Examples of E.T.

Test Design Focus

- Interviewing a candidate (or a program) a question; get an interesting answer that sparks a new question; ask that question, get another interesting answer; repeat
- Evaluating of a feature that is not understood or specified
- Testing a product that does not yet exist
- Using feedback from the last test to inform the next
- Asking “does this risk warrant further testing?”
- High-volume test automation
  - reveals new problems that the team didn’t anticipate or know how to look for in a more systematic way
  - the *exploratory* work here involves continually revising the test series based on what is being learned

Paradigmatic Examples of E.T.



Test Execution Focus

- Retesting and testing around a defect
- Investigating a puzzling situation
- Retesting an old or fixed product
- Pair testing
- Scenario testing, using personae
- Interactive automated testing
  - a.k.a. “computer-assisted testing”
- play testing: customers using loosely scripted usability "charters" as they evaluate video games

Paradigmatic Examples of E.T.

Learning Focus

- Testing a new product
- Improving your a model of product by investigating its elements
- Using and operating a product, and searching for bugs while also searching for new testing ideas
- Scanning or mapping a delivered artifact with focus on potential exploitation, unexpected interaction, or emergent behaviour)
- Interacting with a product to test your model of it

Getting Better at E.T.

- Identifying what we're looking for
- Developing skills and tactics
- Recognizing polarities
- Using heuristics
  - Project Environment
  - Product Elements
  - Quality Criteria
  - Test Techniques
  - Consistency Oracles
- Developing and refining E.T. work products

What we may seek and learn in exploration

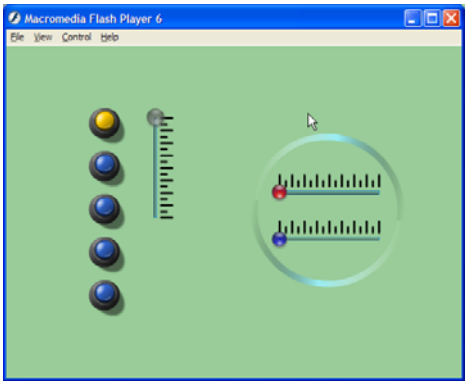
- **Composition**
  - **Affordances:** ways in which the product can be used
  - **Dimensions & Variables:** the product space and the things that can change within it
  - **Relationships & Interactions:** functions that may cooperate or interfere with each other
  - **Navigation:** where things are and how to get to them
- **Conformance**
  - **Benefits:** What the product is good for
  - **Consistencies:** Fulfillment of logical, factual, and cultural expectations
  - **Oracles:** Mechanisms or principles by which we can spot bugs
  - **Bugs and Risks:** Potential problems that could matter to some person or agency

What we may seek and learn in exploration

- **Context**
  - **History:** where the product has been, and how it came to be
  - **Operations:** inferences about its users and the conditions under which it will be used
- **Conditions**
  - **Attitudes:** what our clients care about and what they want from us
  - **Complexities & Challenges:** the hardest things to test.
  - **Resources:** tools and information that might help us test.

An E.T. Exercise

Black Box Test Machine, from James Lyndsay
http://www.workroom-productions.com



We use this odd, mysterious program to give testers experience in understanding and exercising their exploratory skills and tactics.

An E.T. Exercise: "Plenty Questions"

- Like the traditional game of twenty questions, but with a more oblique problem and unlimited yes-or-no questions
  - "A woman falls in love with a man, which causes her to lose her hair."
  - "A person saves his life by holding a newspaper in front of him."
- The purpose is to exercise skills in framing useful, targeted questions in an open-ended search for explanations

An E.T. Exercise: Exploratory Skills and Tactics

1. Team up!
2. Within your group, choose something that you have explored, or could imagine exploring; or an exploration from history
   - A city
   - A country
   - A building
   - A new subject or field
3. Identify the activities, skills, and tactics that happen during exploration

Exploratory Skills and Tactics: One Evolving List

- Modeling
- Resourcing
- Questioning
- Chartering
- Observing
- Manipulating
- Collaborating
- Generating & Elaborating
- Overproduction & Abandonment
- Abandonment & Recovery
- Refocusing
- Alternating
- Branching & Backtracking
- Conjecturing
- Recording
- Reporting
- Evaluating
- Tooling

This list was developed by James and Jonathan Bach, and elaborated by participants at Mike Kelly's session at Consultants' Camp 2006

Exploratory Testing Polarities

- James and Jonathan Bach identify a number contrasting motivations, behaviours and perspectives
- Examples
  - Testing to learn vs. testing to search
  - Warming up vs. cruising vs. cooling down
  - Doing vs. describing
  - Doing vs. thinking
  - Careful vs. quick
  - Data gathering vs. data analysis
- To get better at E.T., recognize these polarities and learn to control and vary them

To Learn Exploratory Testing We Must Learn To *Test*

- Learning how to test in an exploratory way can be challenging, because:

Nobody ever taught us how to test.

- **WHEREAS…**
  - Almost nobody enjoys reviewing written test procedures.
  - Almost nobody knows how to evaluate the quality of written test procedures.
  - Almost every manager seems to think that written tests are a Good Thing.
- **THEREFORE**
  - Writing *awful* test procedures won't get us fired. Some companies will even *reward us* for the poor quality of our test procedures.
- **and**
  - That means there is little pressure on us to become excellent testers.

Heuristics: Generating Solutions Quickly

- **adjective:**

  "serving to discover."

- **noun:**

  "a fallible method for solving a problem."

"Heuristic reasoning is not regarded as final and strict but as provisional and plausible only, whose purpose is to discover the solution to the present problem."
- George Polya, *How to Solve It*

Oracles

An oracle is the principle or mechanism by which you recognize a problem.

"..it works"

"...it appeared at least once to meet some requirement to some degree."

All Oracles Are Heuristic

- There is no single oracle that can tell us whether a program (or a feature) is working correctly at all times and in all circumstances.
- Oracles are fallible and context-dependent.
- Oracles can be contradicted by other oracles.
- Multiple oracles may increase our confidence, but even combinations of oracles are fallible.
- Recognizing a different problem usually requires a different oracle.
- A test designer doesn't need to be aware of an oracle in advance of the observation, unless the test is designed to be run by rote.
- Any time you see a problem, you *must* be using an oracle… so *what is it*?

Example of Heuristic Oracles

- We suspect a problem if a product is *inconsistent* with…
  - History (the past behaviour of the program
  - Image (of the company or organization)
  - Comparable products
  - Claims (made by anyone who matters)
  - User Needs (as we understand them)
  - Product (internally, within itself)
  - Purpose (the product's intended use, implicit or explicit)
  - Statutes (laws or external standards)
- We suspect a problem if a product *is consistent* with a previously observed problem pattern

Use consistency heuristics to support inferences about possible bugs

Tests Are Oracles PLUS Coverage

According to Jerry Weinberg, "it works"

"We haven't tried very hard to make it fail, and we haven't been running it very long or under very diverse conditions, but so far we haven't seen any failures, though we haven't been looking too closely, either."

Coverage

Product coverage is the proportion of the product that has been tested.

There are as many kinds of coverage as there are ways to model the product.

Product Elements

- Structure
- Functional
- Data
- Platform
- Operations
- Time

Capability
Reliability
Usability
Security
Scalability

Performance
Installability
Compatibility
Supportability
Testability

Maintainability
Portability
Localizability

Quality Criteria

Sometimes your coverage is disputed…

"No user would do that."

really means...

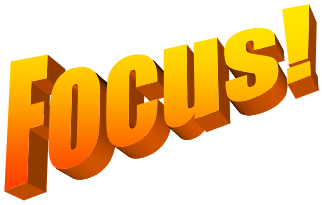
"No user I can *think of*, who I *like*, would do that *on purpose*."

Who aren't you thinking of?
Who don't you like who might really use this product?
What might good users do by accident?

How to Find Elusive Bugs

1. Look over your recent tests.
2. Find some pattern there.
3. With your next few tests, *violate* the old pattern and follow a *new* pattern.
4. Prefer the *MFAT* heuristic (**m**ultiple **f**actors **a**t a **t**ime).

What to Do if You are Confused



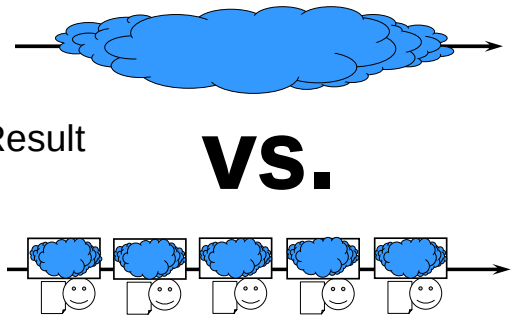
1. Return frequently to a *known* state.
2. Prefer the *OFAT* heuristic (**o**ne **f**actor **a**t a **t**ime).
3. Make *precise* observations.

Taking Notes

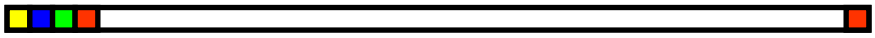
- Test Execution Log (automatic, if possible)
- Test Coverage Outline/Matrix
- Oracle Notes
- Risk/Strategy List
- Issues, Questions & Anomalies
  - It would be easier to test if you changed/added…
  - How does … work?
  - Is this important to test? How should I test it?
  - I saw something strange…

Accountability for E.T.: Session-Based Test Management

1) Charter
2) Time Box
3) Reviewable Result
4) Debriefing



Charter



- ***A clear mission for the session***
- A charter may suggest what should be tested, how it should be tested, and what problems to look for.
- A charter is not meant to be a detailed plan.
- General charters may be necessary at first:
  - "Analyze the Insert Picture function"
- Specific charters provide better focus, but take more effort to design:
  - "Test clip art insertion. Focus on stress and flow techniques, and make sure to insert into a variety of documents. We're concerned about resource leaks or anything else that might degrade performance over time."

Time Box

Focused test effort of fixed duration

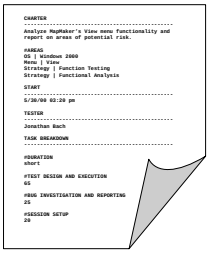
Short: 60 minutes (+-15)
Normal: 90 minutes (+-15)
Long: 120 minutes (+-15)

- Brief enough for accurate reporting.
- Brief enough to allow flexible scheduling.
- Brief enough to allow course correction.
- Long enough to get solid testing done.
- Long enough for efficient debriefings.
- Beware of overly precise timing.

Reviewable Results

A scannable session sheet
(a text file formatted for handling by a Perl script)

- Charter
  - #AREAS
- Start Time
- Tester Name(s)
- Timing
  - Session Duration
  - Percentage spent on
    - Test design and execution
    - Bug investigation and reporting
    - Session setup
  - Proportion of Charter vs. Opportunity (non-charter) work
- Data file names (references)
- Test Notes
- Bugs
  - # BUG
- Issues
  - # ISSUE



Debriefing

Assessment begins with observation

- The manager reviews session sheet to assure that (s)he understands it and that it follows the protocol.
- The tester answers any questions.
- Session metrics are checked.
- Charter may be adjusted.
- Session may be extended.
- New sessions may be chartered.
- Coaching happens.

Accounting for Test/Bug/Setup

- Test, Bug, and Setup are orthogonal categories.
- Estimate the percentage of on-charter work that fell into each category.
- Nearest 5% or 10% is good enough.
- If activities are done simultaneously, report the highest precedence activity.
- Precedence goes in order: T, B, then S.
- All we really want is to track interruptions to testing.
- Don't include Opportunity Testing in the estimate.

Reporting Test Effectiveness

- A "perfect" session is one entirely dedicated to test design and execution
- Setup, and bug investigation and reporting, take time away from test design and execution
- Suppose that a testing a feature takes two minutes
  - this is a highly arbitrary and artificial assumption, but we use it to make a point
- Suppose also that it takes ten minutes to investigate and report a bug
  - another stupid, sweeping generalization in service of the point
- In a 90-minute session, we can run 45 feature tests—*as long as we don't find any bugs*

Reporting Effectiveness

Bugs found	Test design and execution	Bug reporting and investigation	Number of tests
0	90 (45 tests)	0	45
1	80 (40 tests)	10 (1 test)	41
8	10 (5 tests)	80 (8 tests)	13

Investigating and reporting bugs means….

SLOWER TESTING **or...**
REDUCED COVERAGE **...or both.**

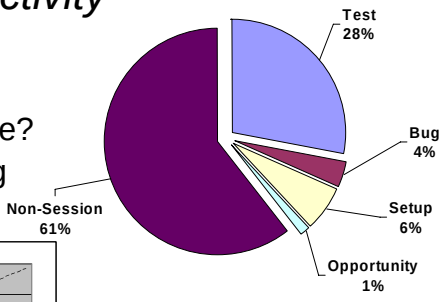
Coverage

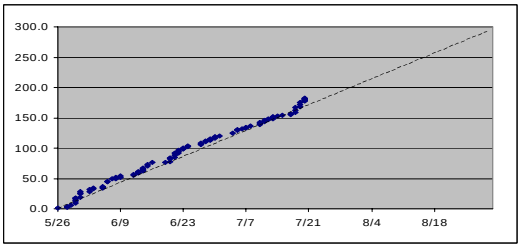
- These are text labels listed in the Charter section of the session sheet. (e.g. “insert picture”)
- Coverage areas can include any way of modeling the product or the test effort
  - functional areas of the product
  - product elements
  - quality criteria
  - test strategies
  - test configuration
  - system configuration parameters
- Use the debriefings to check the validity of the specified coverage areas.

Work Breakdown

Diagnosing the productivity

- Do these proportions make sense?
- How do they change over time?
- Is the reporting protocol being followed?





Estimating a Test Cycle with SBTM

1. How many *perfect* sessions (100% on-charter test design and execution) does it take to obtain the coverage that we seek? *(let's say 40)*
2. How many sessions can the team (of 4 testers) do per day? *(let's say 3 per day, per tester, so 12 sessions)*
3. Sessions contain some setup, and some bug investigation and reporting, so we can't count that. How productive are the sessions *(let's say 66% is on-charter test design and execution)*
4. Estimate: 40 / (12 * .66) = **5 days**
5. We base the estimate on the data we've collected. When any conditions or assumptions behind this estimate change, we will update the estimate.

Challenges of High Accountability E.T. / SBTM

- Architecting the system of charters (test planning)
- Making time for debriefings
- Getting the metrics right
- Creating good test notes
- Keeping the technique from dominating the testing
- Maintaining commitment to the approach

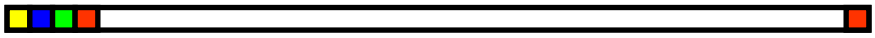
For example session sheets and metrics see
http://www.satisfice.com/sbtm

E.T. as a Pragmatic Practice

- Why invest lots of
  - time
  - effort
  - documentation
  - resources
  - planning

  at the *beginning* of the project, when we know the *least* about it?
- Among other advantages, the learning and looping in exploratory testing make it an important way to *discover* a plan

Resources



- http://www.satisfice.com
  - James Bach's site
- http://www.satisifice.com/moodle
  - free Black Box Software Testing course, written by Kaner and Bach, and presented (on streaming video) by Kaner
- http://www.testingeducation.org
  - lots of testing resources, hosted by Florida Institute of Technology