Exploratory Testing and Leadership
Michael Bolton
DevelopSense
http://www.developsense.com
November 2009

What IS Exploratory Testing?
• Simultaneous test design, test execution, and learning.
  • James Bach, 1995

But maybe it would be a good idea to underscore why that’s important…

What IS Exploratory Testing?
• Simultaneous test design, test execution, and learning, with an emphasis on learning.
  • Cem Kaner, 2005

But maybe it would be a good idea to be really explicit about what goes on…

What IS Exploratory Testing?
• I follow (and to some degree contributed to) Kaner’s definition, which was refined over several peer conferences through 2007:

  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 his or her work
  • by treating test design, test execution, test result interpretation, and test-related learning
  • as mutually supportive activities
  • that run in parallel
  • throughout the project.


Whoa. Maybe it would be a good idea to keep it brief most of the time…

Testing Isn’t Just Checking
• Checking is a process of confirming and verifying existing beliefs
  • Checking can (and I argue, largely should) be done by automation
  • It is a non-sapient process


Oh no! What Does “Non-Sapient” Mean?
• A non-sapient activity can be performed

by a machine that can’t think (but it’s quick and precise)
by a human who has been instructed NOT to think (and that’s slow and erratic)
What is Checking?

- A check has three attributes
  - It requires an observation
  - The observation is linked to a decision rule
  - The observation and the rule can be applied
  - by a machine
  - by a sufficiently disengaged human

What Is Sapience?

- A sapient activity is one that requires a thinking human to perform
- We test not only for repeatability, but also for adaptability, value, and threats to value

This kind of testing CAN NOT be scripted

Checking IS Important

- Despite what the Agilists might have you believe, checking is not new
  - D. McCracken (1957) refers to “program checkout”
  - Jerry Weinberg: checking was important in the early days because
    - computer time was expensive
    - programmers were cheap
    - the machinery was so unreliable
  - Checking has been rediscovered by the Agilists
    - centrally important to test-driven development, refactoring, continuous integration & deployment
    - excellent checking is surrounded by testing work
  - CHECks are CHange detECtors

But…

- A good tester doesn’t just ask
  - Pass or Fail?
  
- A good tester asks
  
Is there a problem here?

Testing IS Exploring

- Testing, as I see, it is all about exploration, discovery, investigation, and learning
  - Testing can be assisted by machines, but can’t be done by machines alone
  - It is a sapient process

Humans can…

- I can’t do that, but I can help you act on your ideas.

Machines can’t...

- recognize new risks
- investigate
- predict
- speculate
- empathize
- judge
- suggest
- project
- appreciate
- context

- recognize opportunities
- strategize
- learn
- charter
- work around problems
- make conscious decisions
- collaborate
- resource
- troubleshoot
- invent

The Danger of Scripts

- Scripts aren’t necessary for skilled (human) testers
- Script preparation takes away from testing time
- Bugs found and fixed during script prep tend to stay fixed
- Scripts separate design, execution, interpretation, and learning… and thus DE-SKILL
- Scripts drive inattentional blindness

Besides...

- You cannot use a script to
  - investigate a problem you’ve found
  - decide that there’s a problem with a script
  - escape the script problem you’ve identified
  - determine the best way to phrase a report
  - unravel a puzzling situation

Even "scripted" testers explore all the time!

So why don’t we hear more about E.T.?

FEAR

- Maybe managers fear that E.T. depends on skill
  - but who benefits from ANY unskilled testing?
- Maybe managers fear that E.T. is unstructured
  - but it is structured
- Maybe managers fear that E.T. is unaccountable
  - but it can be entirely accountable
- Maybe managers fear that E.T. is unmanageable
  - but you can manage anything if you put your mind to it

Skill vs. Alternatives

Heuristics are applied, not followed.

This...

1. Do this
2. Then do this
3. Then do this
4. Then do this
5. And then this...

The skilled tester remains in control of the process.

...not this.

Scripted procedures give the illusion of control over unskilled testers.
**Exploratory Testing IS Structured**

- We’ve studied the structure of ET, we’ve written about it, and we know how to teach it.
- The structure of ET comes from many sources:
  - Test design heuristics
  - Chartering
  - Time boxing
  - Perceived product risks
  - The nature of specific tests
  - The structure of the product being tested
  - The process of learning the product
  - Development activities
  - Constraints and resources afforded by the project
  - The skills, talents, and interests of the tester
  - The overall mission of testing

In other words, it’s not “random”, but systematic. Not procedurally structured, but cognitively structured.

**A Heuristic Test Strategy Model**

- **Project Environment**
- **Tests**
- **Quality Criteria**
- **Product Elements**

**Oracles**

An oracle is a heuristic principle or mechanism by which someone might recognize a problem.

(usually works, might fail)

(but not decide conclusively)

Bug (n): Something that bugs someone who matters

**Consistency** (“this agrees with that”) an important theme in oracles

- History
- Image
- Comparable Products
- Claims
- User Expectations
- Purpose
- Product Standards

Consistency heuristics rely on the quality of your models of the product and its context.

**Test Coverage Isn’t Just Code Coverage**

Test coverage is the amount of the system space that has been tested.

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

**Product Elements**

- Capability
- Reliability
- Usability
- Security
- Scalability
- Performance
- Installability
- Compatibility
- Supportability
- Testability
- Maintainability
- Portability
- Localizability

**Quality Criteria**

- Purpose
- Structure
- Functional
- Data
- Platform
- Operations
- Time
Cost as a Simplifying Factor
Try quick tests as well as careful tests

In my travels, I’ve seen extraordinary emphasis on long cycles of planning without feedback. This makes testing ineffective and slow.

A quick test is a cheap test that has some value, gives fast feedback, but requires little preparation, knowledge, or time to perform.

Bursts of quick tests represent a great way to discover risks upon which careful testing can be better focused.

What Does Rapid ET Look Like?
Concise Documentation Minimizes Waste

General
- Testing Heuristics
- Risk Catalog

Project-Specific
- Coverage Model
- Risk Model
- Test Strategy
- Reference
- Schedule
- Issues
- Bugs
- Status
- Dashboard

How To Measure Test Coverage
(it’s not merely code coverage)

- Identify quality criteria
- Identify session time focused on each criterion
- Consider product elements (structure, function, data, platform, operations, and time)
- Break them down into coverage areas
- Assess test coverage in terms of
  - Level 1: Smoke and sanity
  - Level 2: Common, core, critical aspects
  - Level 3: Complex, challenging, harsh, extreme, exceptional

How To Measure ET Efficiency
Track rough percentage of time spent on
- Test design and execution
- Bug investigation and reporting
- Setup

Ask why time was spent on each:
- Lots on T might indicate great code, but might indicate poor bug-finding skill
- Lots on B might mean code quality problems, but might suggest inefficiency in reporting
- Lots on S might mean testability or configuration problems for customers, or it might mean early days of testing

How To Manage Exploratory Testing
Achieve excellent test design by exploring different test designs while actually testing and interacting with the system

Guide testers with personal supervision and concise documentation of test ideas. Meanwhile, train them so that they can guide themselves and be accountable for increasingly challenging work.

Accountability for Exploratory Testing:
Session-Based Test Management

- Charter
  - A clear, concise mission for a test session
- Time Box
  - 90-minutes (+/- 45)
- Reviewable Results
  - a session sheet—a test report whose raw data can be scanned, parsed and compiled by a tool
- Debriefing
  - a conversation between tester and manager or test lead

For more info, see http://www.satisfice.com/sbtm
What Is Leadership?

- "Leadership is the process of creating an environment in which everyone is empowered."
  - Gerald M. Weinberg, *Becoming a Technical Leader*
- Leaders require freedom and responsibility to optimize the quality of their work, while granting freedom and responsibility to others to do the same.

What Does A Leader Do?

- Performs complex cognitive tasks
- Has access to a large number of models
- Applies the models to absorb, process, and respond to whatever information is available
- Responds, flexibly and adaptably, to whatever complications the situation presents
- Empowers (teams of skilled technical) people
- Learns rapidly and observes keenly
- Is introspective and self-critical
- Motivates, organizes, and innovates

Key Ideas

- All managers should be leaders, but managers are not the only leaders
- Managers who relinquish control foster environments in which leadership can blossom
- Managers who seize control and won’t let go destroy leadership

Motivation: How To Kill It

- Make people feel that change will not be appreciated
- Do everything for them so they won’t feel the need to do things themselves
- Discourage anything that people might enjoy doing for its own sake

Organization: How To Foster Chaos

- Encourage such high competition that co-operation will be unthinkable
- Keep resources slightly below the necessary minimum
- Suppress information of general value, or bury it in an avalanche of meaningless words and paper

Ideas: How To Suppress the Flow

- Don’t listen when you can criticize
- Give your own ideas first, and loudest
- Punish those who offer suggestions
- Keep people from working together
- Above all, tolerate no laughter
Leadership Is Exploratory!

- A leader
  - both grants and receives freedom with responsibility
  - doesn’t follow a script
  - fosters fault-tolerant environments
  - fosters and practices learning
  - practices critical thinking

In other words, it’s not “random”, but systematic.

People I DO See As Leaders

- People who question what they see and hear
  - like participants in Edista’s Test Republic
- People who exchange their ideas
  - like participants in The Bangalore Workshops on Software Testing
- People who practice their craft
  - like the Weekend Testers
  - (see the presentation this afternoon!)

People I DO NOT See As Leaders

- People who are afraid to speak truth to power
- Those who do not actively question the outdated testing methodologies
- Those who disempower other people
  - those (including, alas, Indian managers) who see testers (and especially Indian testers) as hopelessly unskilled
  - anyone involved with scripted testing (unless the script is for a machine)
  - certificationists; people who participate in or promote the empty certifications that we currently have
  - Western organizations that help promote this stuff

How Do Programmers Program?

- Do we use programming cases?
- Do we follow programming scripts?
  - Is there a step-by-step procedure for the development of every program?
  - Does each programming task have an expected, predicted result?
- Do we evaluate programmers by counting the lines of code they write?
- Do we evaluate programmer performance by “coding error escape rates”?
- Do we aspire to reduce the cost of programming by bringing in development automation?

How Do Managers Manage?

- Do we use management cases?
- Do we follow management scripts?
  - Is there a step-by-step procedure for every management action?
  - Does each management action have an expected, predicted result?
- Do we evaluate managers by counting their decisions?
- Do we evaluate management performance by “bad decision escape rates”?
- Do we aspire to reduce the cost of management by bringing in management automation?

So… What Do We Want?

- If we want to miss important problems slowly
  - emphasize confirmation
  - emphasize repetition
  - then complain about how little time we have
- If we want to find important problems quickly
  - reduce wasted time and wasted effort
  - prevent regression problems
  - emphasize exploration, discovery, investigation
  - train and empower testers
  - grant them freedom and responsibility for the quality of their work
What if we train our people and they leave?

What if you don't train them and they stay?

Author unknown, but I'm envious of him/her.

Acknowledgements

- James Bach (http://satisfice.com)
- Cem Kaner (http://www.kaner.com)
- Jerry Weinberg (http://www.geraldmweinberg.com)

Questions? More information?
Michael Bolton
http://www.developsense.com
michael@developsense.com

Readings

- Perfect Software and Other Illusions About Testing
- Becoming a Technical Leader
- Quality Software Management, Vol. 1: Systems Thinking
  - Gerald M. Weinberg
- Lessons Learned in Software Testing
  - Kaner, Bach, and Pettichord
- DevelopSense Web Site (and blog), http://www.developsense.com
  - Michael Bolton
- Satisifice Web Site (and blog), http://www.satisfice.com
  - James Bach
  - Jonathan Kohl
  - Elisabeth Hendrickson