An Exploratory Tester’s Notebook

Michael Bolton
DevelopSense
QUEST Chicago
April 2009

I’m Michael Bolton

Not the singer. Not the guy in Office Space. No relation.
Who I Am

Michael Bolton
(not the singer, not the guy in Office Space)
DevelopSense, Toronto, Canada

mb@developsense.com
+1 (416) 992-8378
http://www.developsense.com

I help solve testing problems
that other people can’t solve.

Acknowledgements

• James Bach
  • some material in this presentation is taken from
    our Rapid Software Testing course

• Cem Kaner

• Jon Bach
  • who introduced me to the Moleskine notebook
    and who, with James, created and documented
    session-based test management—and provides
    exemplary session notes
The First Law of Documentation

“That should be documented.”

“That should be documented if and when and how it serves our purposes.”

Who will read it? Will they understand it? Is there a better way to communicate that information? What does documentation cost you?

• For updated notes AND a more formal paper on notebooks: quest2009@developsense.com
**Documentation: Product or Tool?**

*Paradigm:*
- Tool
- Product

*Audience:*
- Self
- Team
- Customers
- Regulators

*Purpose:*
- Recollection
- Organization
- Communication
- Demonstration

---

**Notebooks: A Personal View**

- Over the last I’ve been keeping a set of notebooks
- This is an experience report on how *one* exploratory tester and consultant (me) has used them
- This is a context-driven talk; this is not a best-practices talk
My Introduction to the Moleskine

- I’ve kept documents (mostly for school or work) all my life
  - scribblers
  - legal pads
  - ASCII text files
  - Word documents
- In January 2004, I noticed Jon Bach’s Moleskine notebook
- In January 2005, James Bach suggested I get one. I did.
- It turns out there’s a something of a cult…
  - http://www.moleskinerie.com/
  - http://www.moleskinecity.com
  - http://en.wikipedia.org/wiki/Moleskine

So What’s the Big Deal?

- Several form factors
  - larger notebook
  - smaller notebook (pocket size)
  - reporter style
  - memo pockets
- Three line styles
  - plain
  - ruled
  - squared
- Page marker
- Elastic closure
- Back pocket
- Sewn binding, lies flat
- Geek-chic-mystique-boutique appeal
  - Well-constructed
  - Durable
  - Somewhat expensive
Who Uses Notebooks?

Exploratory testers are all three, and more.
ALL testers are, at some point, explorers.
Maybe we should learn explorers’ tools.

Paradigmatic Examples

- Leonardo: inventor’s notebook
  - Codex Arundel, Codex Leicester, Codex Foster, etc.
  - contents: sketches, inventions, architecture, elements of mechanics, painting ideas, human anatomy, grocery lists and even people that owed him money (Wikipedia)
Paradigmatic Examples

• Gordon Gould: inventor’s notebook
  • One of the people involved in the invention of the laser.
  • Notes created after meeting with Charles Hard Townes, November 1957 include the acronym "LASER" and several aspects of its design.

Paradigmatic Examples

• William Logan: explorer/geologist’s notebook
  • Written in Stone—geological explorations of Canada
  • http://www.collectionscanada.ca/logan/021014-1000-e.html
Paradigmatic Examples

- William Logan – Explorer’s notebook
  - Written in Stone—geological explorations of Canada
  - http://www.collectionscanada.ca/logan/021014-1000-e.html

Why Notebooks Now?

- In the age of the Palm (I have one) and the smartphone (I have one) and the laptop (I have one), why use notebooks?
  - They’re portable
  - They never crash
  - They never forget to save
  - Battery doesn’t wear out
  - They’re free-form
  - They’re available
  - They’re personal
What I Use Notebooks For

- Brainstorming
- Sketches
- Catalogs of heuristics
- Mind maps
- Diagrams
- Action items and reminders
- “Fieldstones” and blog entries
- Conference or workshop sessions
- Test notes, and practice taking them

My Notebooks

- I thought I lost my notebook once. Now…
A Busy Couple of Days, With Rant

USE THE DAMN THING

- You can take your time, online, automation, or writing scripts, etc. Doing steps diagrams, but you most of the time. What most of us do, wouldn’t it be a good idea to use the damned thing for something like it’s intended purpose? We’re trying not to allow you to do all the other stuff quickly, straight from your head wherever possible, so don’t think that a user of your product is likely

An Exploratory Testing Session

- It’s a cell phone, yeah. In the button - want that. Doesn’t that. It’s a credit card. There’s a little side - won’t test part.
- If you’re thinking about buying or giving money, then the best of luck.
- What I’m on. The button doesn’t work, so can’t go home buying problems.
- At least one of about 10 screen cameras is red, block.
An Exploratory Testing Session

- Created in a very temporary and not "transient" test environment.
- The idea is a mix of the two.
- Turn on: on the unit, and then to "on." On the screen goes completely off.
- Appears, when receiving.
- The button switches on that mode.
- Pressed now, button should flip to home. Click once to return to menu. "On" button doesn’t clear the home button, you have to pick on the home button.

- Picture:
  - Option "go" or option "stop".
  - Note 1: support for not second.

- The screen helps from main.
- This doesn’t provide help, takes me to language selection.

- Home:
  - When 3, very sudden.

- Minutes:
  - Your latest update (15 minutes).
  - Dose option: "sync".
  - Works:

- Pressed help at this hour.
- Being me only to a lame illustration of how to use...
An Exploratory Testing Session

Observation on Building Skill
Diagramming

• “The diagram is nothing; the diagramming is everything.”
  • Jerry Weinberg

James Bach on White-Box Risk-Based Analysis, with Diagrams

  • [pointing at a box] What if the function in this box fails?
  • Can this function ever be invoked at the wrong time?
  • [pointing at any part of the diagram] What error checking do you do here?
  • [pointing at an arrow] What exactly does this arrow mean? What would happen if it was broken?

Guideword Heuristics for Diagram Analysis

- Boxes
  • Missing/Drop-out
  • Extra/Interfering
  • Incorrect
  • Timing/Sequencing
  • Contents/Algorithms
  • Conditional behavior
  • Limitations
  • Error Handling

- Lines
  • Missing/Drop-out
  • Extra/Forking
  • Incorrect
  • Timing/Sequencing
  • Status Communication
  • Data Structures

- Paths
  • Simplest
  • Popular
  • Critical
  • Complex
  • Pathological
  • Challenging
  • Error Handling
  • Periodic

- James Bach, from our Rapid Software Testing course.
Incremental Catalogs

• As ideas occur to me, I might reserve a single page or two to consolidate them.

An Ongoing Bug Catalog
Portable Presentations!

• Easier than booting the laptop!

<table>
<thead>
<tr>
<th>Bugs Found</th>
<th>Test Design</th>
<th>Test Execution</th>
<th>Test Investigation</th>
<th>Test Reporting</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>90 mins</td>
<td>0 mins</td>
<td>0 mins</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>90 mins</td>
<td>10 mins</td>
<td>1 min</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10 mins</td>
<td>90 mins</td>
<td>0 mins</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

When more bugs are found, testing slows or coverage worsens or both.

E.T. Skills and Tactics

• Mike Kelly elaborated on this list of exploratory skills and tactics, which was originally written by James and Jon Bach.

• In writing down the list, I reckoned that tooling (distinct from resourcing) and evaluating were (for me) missing.
How do you record your work?

Use concise, modular documents that help tell the testing story.

How Might We Organize, Record, and Report Coverage?

- annotated diagrams (see earlier slides)
- coverage outlines and risk lists
  - plentiful examples in the Rapid Software Testing notes
  - requirement / risk vs. coverage matrices
    - (see subsequent slides)
- bug taxonomies (external and in-house)
  - example: appendix to Testing Computer Software
  - example: “Bugs in your Shopping Cart”,
    www.kaner.com/pdfs/BugsInYourShoppingCart.pdf
- summarized log files
- automated tools (e.g. profilers, coverage tools)
### Quality Criteria Coverage Matrix

![Image of Quality Criteria Coverage Matrix](image1)

### Product Element Coverage Matrix

![Image of Product Element Coverage Matrix](image2)
E.T. Notetaking Online:
Session-Based Test Management

- Charter
  - A clear, concise mission for a session of testing
- Time Box
  - 90-minute (+/- 30), long enough for setup and focused work; short enough to make sure things don’t get off track
- Reviewable Result
  - next slide!
- Debriefing
  - conversation between tester and manager
  - problems, bugs and issues can be discussed
  - new risks can be identified
  - coaching and mentoring can happen

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 test session sheet that can be scanned by a Perl script for compilation elsewhere

• Charter
  • #AREAS
• Start Time
• Tester Name(s)
• Breakdown
  • DURATION
  • TEST DESIGN AND EXECUTION
  • BUG INVESTIGATION AND REPORTING
  • SESSION SETUP
  • CHARTER/OPPORTUNITY
• Data Files
• Test Notes
• Bugs
  • BUG
• Issues
  • ISSUE
**Debriefing**

*Assessment begins with observation*

- The manager or test lead reviews the 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 and mentoring happens.

**The Breakdown Metrics**

*Testing is like looking for worms*

- Test Design and Execution
- Session Setup
- Bug Investigation and Reporting
Reporting the TBS Breakdown
A guess is okay, but follow the protocol

- Test, Bug, and Setup are orthogonal categories.
- Estimate the percentage of 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.

Test Session Effectiveness

- A "perfectly effective" testing session is one entirely dedicated to test design, test execution, and learning
  - a "perfect" session is the exception, not the rule
- Test design and execution tend to contribute to test coverage
  - varied tests tend to provide more coverage than repeated tests
- Setup, bug investigation, and reporting take time away from test design and execution
- Suppose that testing a feature takes two minutes
  - this is a highly arbitrary and artificial assumption—that is, it’s wrong, but we use it to model an issue and make a point
- Suppose also that it takes eight extra 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
### How Do We Spend Time?
*(assuming all tests below are *good* tests)*

<table>
<thead>
<tr>
<th>Module</th>
<th>Bug reporting/investigation (time spent on tests that find bugs)</th>
<th>Test design and execution (time spent on tests that find no bugs)</th>
<th>Number of tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (good)</td>
<td>0 minutes (no bugs found)</td>
<td>90 minutes (45 tests)</td>
<td>45</td>
</tr>
<tr>
<td>B (okay)</td>
<td>10 minutes (1 bug, 1 test)</td>
<td>80 minutes (40 tests)</td>
<td>41</td>
</tr>
<tr>
<td>C (bad)</td>
<td>80 minutes (8 bugs, 8 tests)</td>
<td>10 minutes (5 tests)</td>
<td>13</td>
</tr>
</tbody>
</table>

**Investigating and reporting bugs means…**

**SLOWER TESTING** or…

**REDUCED COVERAGE** …or both.

- In the first instance, our *coverage* is great—but if we’re being assessed on the number of bugs we’re finding, we look bad.
- In the second instance, coverage looks good, and we found a bug, too.
- In the third instance, we look good because we’re finding and reporting lots of *bugs*—but our *coverage* is suffering severely. A system that rewards us or increases confidence based on the number of bugs we find might mislead us into believing that our product is well tested.

### What Happens The Next Day?
*(assume 6 minutes per bug fix verification)*

<table>
<thead>
<tr>
<th>Fix verifications</th>
<th>Bug reporting and investigation today</th>
<th>Test design and execution today</th>
<th>New tests today</th>
<th>Total over two days</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 min</td>
<td>0</td>
<td>45</td>
<td>45</td>
<td>90</td>
</tr>
<tr>
<td>6 min</td>
<td>10 min (1 new bug)</td>
<td>74 min (37 tests)</td>
<td>38</td>
<td>79</td>
</tr>
<tr>
<td>48 min</td>
<td>40 min (4 new bugs)</td>
<td>2 min (1 test)</td>
<td>5</td>
<td>18</td>
</tr>
</tbody>
</table>

**Finding bugs today means…**

**VERIFYING FIXES LATER** …which means…

**EVEN SLOOWER TESTING** or…

**EVEN LESS COVERAGE** …or both.

…and note the optimistic assumption that all of our fixed verifications worked, and that we found no new bugs while running them. Has this ever happened for you?
Testing vs. Investigation

• Note that I just gave you a compelling-looking table, using simple measures, but notice that we still don’t know anything about…
  • the quality and relevance of the tests
  • the quality and relevance of the bug reports
  • the skill of the testers in finding and reporting bugs
  • the complexity of the respective modules
  • luck

…but if we ask better questions, instead of letting data make our decisions, we’re more likely to make progress.

Session-Based Test Management

For more information on SBTM, see http://www.satisfice.com/sbtm
How do you effectively report your work?

Learn to tell a compelling story that provokes the right questions.

The Dashboard Concept

Large dedicated whiteboard “Do Not Erase”

Project conference room

Project status meeting
### Testing Dashboard

<table>
<thead>
<tr>
<th>Area</th>
<th>Effort</th>
<th>C.</th>
<th>Q</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>file/edit</td>
<td>high</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>view</td>
<td>low</td>
<td>1+</td>
<td></td>
<td>1345, 1363, 1401</td>
</tr>
<tr>
<td>insert</td>
<td>low</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>low</td>
<td>2+</td>
<td></td>
<td>automation broken</td>
</tr>
<tr>
<td>tools</td>
<td>blocked</td>
<td>1</td>
<td></td>
<td>crashes: 1406, 1407</td>
</tr>
<tr>
<td>slideshow</td>
<td>low</td>
<td>2</td>
<td></td>
<td>animation memory leak</td>
</tr>
<tr>
<td>online help</td>
<td>blocked</td>
<td>0</td>
<td></td>
<td>new files not delivered</td>
</tr>
<tr>
<td>clipart</td>
<td>none</td>
<td>1</td>
<td></td>
<td>need help to test...</td>
</tr>
<tr>
<td>converters</td>
<td>none</td>
<td>1</td>
<td></td>
<td>need help to test...</td>
</tr>
<tr>
<td>install</td>
<td>start 3/17</td>
<td>0</td>
<td></td>
<td>lab time is scheduled</td>
</tr>
<tr>
<td>compatibility</td>
<td>start 3/17</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>general GUI</td>
<td>low</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Updated:** 2/21
**Build:** 38

### Product Area

- 15-30 areas (keep it simple)
- Avoid sub-areas: they’re confusing.
- Areas should have roughly equal value.
- Areas together should be inclusive of everything reasonably testable.
- “Product areas” can include tasks or risks- but put them at the end.
- Minimize overlap between areas.
- Areas must "make sense" to your clients, or they’ll ignore the board.
## Test Effort

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Not testing; not planning to test.</td>
</tr>
<tr>
<td>Start</td>
<td>No testing yet, but expect to start soon.</td>
</tr>
<tr>
<td>Low</td>
<td>Regression or spot testing only; maintaining coverage.</td>
</tr>
<tr>
<td>High</td>
<td>Focused testing effort; increasing coverage.</td>
</tr>
<tr>
<td>Pause</td>
<td>Temporarily ceased testing, though area is testable.</td>
</tr>
<tr>
<td>Blocked</td>
<td>Can’t effectively test, due to blocking problem.</td>
</tr>
<tr>
<td>Ship</td>
<td>Going through final tests and signoff procedure.</td>
</tr>
</tbody>
</table>

- Use red to denote significant problems or stoppages, as in `blocked`, `none`, or `pause`.
- Color `ship` green once the final tests are complete and everything else on that row is green.
- Use neutral color (such as black or blue, but pick only one) for others, as in `start`, `low`, or `high`. 
We don’t have good information about this area.

**Sanity Check:** major functions & simple data.

More than sanity, but many functions not tested.

**Common & Critical:** all functions touched; common & critical tests executed.

Some data, state, or error coverage beyond level 2.

**Complex Cases:** strong data, state, exceptional, error, extreme, stress or long-sequence testing.

- Color green if coverage level is acceptable for ship, otherwise color black.
- Level 1 and 2 focus on functional requirements and capabilities: *can* this product work at all?
- Level 2 may span 50%-90% code coverage.
- Level 2+ and 3 focus on information to judge performance, reliability, compatibility, and other “ilities”: *will* this product work under realistic usage?
- Level 3 or 3+ implies “if there were a bad bug in this area, we would probably know about it.”
### Quality Assessment

| | “We know of no problems in this area that threaten to stop ship or interrupt testing, nor do we have any definite suspicions about any.” |
| | “We know of problems that are possible showstoppers, or we suspect that there are important problems not yet discovered.” |
| | “We know of problems in this area that definitely stop ship or interrupt testing.” |

### Comments

Use the comment field to explain anything colored red, or any non-green quality indicator.

- Problem ID numbers.
- Reasons for pausing, or delayed start.
- Nature of blocking problems.
- Why area is unstaffed.
Using the Dashboard

- **Updates**: 2-5/week, or at each build, or prior to each project meeting.
- **Progress**: Set expectation about the duration of the “Testing Clock” and how new builds reset it.
- **Justification**: Be ready to justify the contents of any cell in the dashboard. The authority of the board depends upon meaningful, actionable content.
- **Going High Tech**: Sure, you can put this on the web, but will anyone actually look at it???

“We shape our tools; thereafter they shape us.”
- Marshall McLuhan