An Exploratory Tester's Notebook

Michael Bolton, DevelopSense
mb@developsense.com

Biography

Michael Bolton is the co-author (with senior author James Bach) of Rapid Software Testing, a course that presents a methodology and mindset for testing software expertly in uncertain conditions and under extreme time pressure.

A testing trainer and consultant, Michael has over 17 years of experience in the computer industry testing, developing, managing, and writing about software. He is the founder of DevelopSense, a Toronto-based consultancy. He was with Quarterdeck Corporation for eight years, during which he delivered the company's flagship products and directed project and testing teams both in-house and around the world.

Michael has been teaching software testing around the world for eight years. He was an invited participant at the 2003, 2005, 2006, and 2007 Workshops on Teaching Software Testing in Melbourne and Palm Bay, Florida; was a member of the first Exploratory Testing Research Summit in 2006. He is also the Program Chair for TASSQ, the Toronto Association of System and Software Quality, and a co-founder of the Toronto Workshops on Software Testing. He has a regular column in Better Software Magazine, writes for Quality Software (the magazine published by TASSQ), and sporadically produces his own newsletter.

Michael lives in Toronto, Canada, with his wife and two children.

Michael can be reached at mb@developsense.com, or through his Web site, http://www.developsense.com

Abstract: One of the perceived obstacles towards testing using an exploratory testing approach is that exploration is unstructured, unrepeatable, and unaccountable, but a look at history demonstrates that this is clearly not the case. Explorers and investigators throughout history have made plans, kept records, written log books, and drawn maps, and have used these techniques to record information so that they could report to their sponsors and to the world at large. Skilled exploratory testers use similar approaches to describe observations, to record progress, to capture new test ideas, and to relate the testing story and the product story to the project community. By focusing on what actually happens, rather than what we hope will happen, exploratory testing records can tell us even more about the product than traditional pre-scripted approaches do.

In this presentation, Michael Bolton invites you on a tour of his exploratory testing notebook and demonstrates more formal approaches to documenting exploratory testing. The tour includes a look at an informal exploratory testing session, simple mapping and diagramming techniques, and a look at a Session-Based Test Management session sheet. These techniques can help exploratory testers to demonstrate that testing has been performed diligently, thoroughly, and accountably in a way that gets to the heart of what excellent testing is all about: a skilled technical investigation of a product, on behalf of stakeholders, to reveal quality-related information of the kind that they seek.

Documentation Problems

There are many common claims about test documentation: that it's required for new testers or share testing with other testers; that it's needed to deflect legal liability or to keep regulators happy; that it's needed for repeatability, or for accountability; that it forces you to think about test strategy. These claims are typically used to support heavyweight and formalized approaches to test documentation (and to testing itself), but no matter what the motivation, the claims have this in common: they rarely take context, cost, and value into account. Moreover, they often leave out important elements of the story. Novices in any discipline learn not only through documents, but also by observation, participation, practice, coaching, and mentoring; tester may exchange information through conversation, email, and socialization. Lawyers will point out that documentation is only one form of evidence—and that evidence can be used to buttress or to skewer your case—while regulators (for example, the FDA[1]) endorse the principle of the least burdensome approach. Processes can be repeatable without being documented (how do people get to work in the morning?), and auditors are often more interested in the overview of the story than each and every tiny detail. Finally, no document—least of all a template—ever *forced* anyone to think about anything; the thinking part is always up to the reader, never to the document.

Test documentation is often driven by templates in a way that standardizes look and feel without considering content or context. Those who set up the templates may not understand testing outside the context for which the template is set up (or they may not understand testing at all); meanwhile, testers who are required to follow the templates don't own the format. Templates—from the IEEE 829 specification to Fitnesse tests on Agile projects—can standardize and formalize test documentation, but they can also standardize and formalize thinking about testing and our approaches to it. Scripts stand the risk of reducing learning rather than adding to it, because they so frequently leave out the motivation for the test, alternative ways of accomplishing the user's task, and variations that might expose bugs.

Cem Kaner, who coined the term exploratory testing in 1983, has since defined it as "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."[2] A useful summary is "simultaneous test design, test execution, and learning." In exploratory testing, the result of the last test strongly influences the tester's choices for the next test. This suggests that exploratory testing is incompatible with most formalized approaches to test documentation, since most of them segregate design, execution, and learning; most emphasize scripted actions; and most try to downplay the freedom and responsibility of the individual tester. Faced with this problem, the solution that many people have used is simply to avoid exploratory testing—or at least to avoid admitting that they do it, or to avoid talking about it in reasonable ways. As

---

[1] *The Least Burdensome Provisions of the FDA Modernization Act of 1997; Concept and Principles; Final Guidance for FDA and Industry.* www.fda.gov/cdrh/modact/leastburdensome.html

[2] This definition was arrived at through work done at the 2006 Workshop on Heuristic and Exploratory Testing, which included James Bach, Jonathan Bach, Scott Barber, Michael Bolton, Tim Coulter, Rebecca Fiedler, David Gilbert, Marianne Guntow, James Lyndsay, Robert Sabourin, and Adam White. The definition was used at the November 2006 QAI Conference. Kaner, "Exploratory Testing After 23 Years", www.kaner.com/pdfs/ETat23.pdf

McLuhan said, "We shape our tools; thereafter our tools shape us."[3] Test documentation is a tool that shapes our testing.

Yet exploration is essential to the investigative dimension of software testing. Testing that merely confirms expected behaviour can be expected to suffer from fundamental attribution error ("it works"), confirmation bias ("all the tests pass, so it works"), and anchoring bias ("I *know* it works because all the tests pass, so it works"). Testers who don't explore the software fail to find the bugs that real users find when *they* explore the software. Since any given bug is a surprise, no script is available to tell you how to investigate that bug.

Sometimes documentation is a product, a deliverable of the mission of testing, designed to be produced for and presented to someone else. Sometimes documentation is a tool, something to help keep yourself (or your team) organized, something to help with recollection, but not intended to be presented to anyone[4]. In the former case, presentation and formatting are important; in the latter case, they're much less important. In this paper, I'll introduce (or for some people, revisit) two forms of documentation—one primarily a tool, and the other a product—to support exploratory approaches. The first tends to emphasize the learning dimension, the latter tends to be more applicable to test design and test execution.

This paper and the accompanying presentation represent a highly subjective and personal experience report. While I may offer some things that I've found helpful, this is not intended to be prescriptive, or to offer "best practices"; the whole point of notebooks—for testers, at least—is that they become what you make of them.

An Exploratory Tester's Notebook

Like most of us, I've kept written records, mostly for school or for work, all my life. Among other means of preserving information, I've used scribblers, foolscap paper, legal pads, reporter or steno notepads, pocket notepads, ASCII text files, Word documents, spreadsheets, and probably others.

In 2005, I met Jon Bach for the first time. Jon, brother of James Bach, is an expert exploratory tester (which apparently runs in the family) and a wonderful writer on the subject of E.T., and in particular how to make it accountable. The first thing that I noticed on meeting Jon is that he's an assiduous note-taker—he studied journalism at university—and over the last year, he has inspired me to improve my note-taking processes.

The Moleskine Notebook

One factor in my personal improvement in note-taking was James Bach's recommendation of the Moleskine pocket notebook. I got my first one at the beginning of 2006, and I've been using it ever since. There are several form factors available, with soft or hard covers. The version I have fits in a pocket; it's perfect-bound so it lies flat; it has a fabric bookmark and an elasticized loop that holds the book closed. The pages can be unlined, lined, or squared (graph paper)[5]. I prefer the graph paper; I find that it helps with sketching and with laying out tables of information.

---

[3] Marshall McLuhan, *Understanding Media: The Extensions of Man (Critical Edition).* Gingko Press, Costa Madera, CA, September 2003.

[4] See Kaner, Cem; Bach, James, and Pettichord, Bret, *Lessons Learned in Software Testing*. John Wiley & Sons, New York, 2002.

[5] They can also be lined with five-line staff paper for musicians.

The Moleskine has a certain kind of chic/geek/boutique/mystique kind of appeal; it turns out that there's something of a cult around them, no doubt influenced by their marketing. Each notebook comes with a page of history in several languages, which adds to the European cachet. The page includes the claim that the Moleskine was used by Bruce Chatwin, Pablo Picasso, Ernest Hemingway, Henri Mattisse, Andre Breton, and others who are reputed to have used the Moleskine. The claim is fictitious[6], although these artists did use books of the same colour, form factor, with sewn bindings and other features that the new books reproduce. The appeal, for me, is that the books are well-constructed, beautiful, and inviting. This reminds me of Cem Kaner's advice to his students: "Use a good pen. Lawyers and others who do lots of handwriting buy expensive fountain pens for a reason. The pen glides across the page, requiring minimal pressure to leave ink."[7] A good tool asks to be used.

Why Use Notebooks?

In the age of the personal digital assistant (I have one), the laptop computer, (I have one), and the desktop computer (I have one), and the smart phone (I don't have one), why use notebooks?

- They're portable, and thus easy to have consistently available.
- They never crash.
- They never forget to auto-save.
- The batteries don't wear out, they don't have to be recharged—and they're never AA when you need AAA or AAA when you need AA.
- You don't have to turn them off with your other portable electronic devices when the plane is taking off or landing.

Most importantly, notebooks are free-form and personal in ways that the "personal" computer cannot be. Notebooks afford diversity of approaches, sketching and drawing, different thinking styles, different note-taking styles. All Windows text editors, irrespective of their features, still look like Windows programs at some level. In a notebook, there's little to no reformatting; "undo" consists of crossing out a line or a page and starting over or, perhaps more appropriately, of tolerating imperfection. When it's a paper notebook, and it's your own, there's a little less pressure to make things look good. For me, this allows for a more free flow of ideas.

Figure 1: Page from Michael Bolton's Notebook #2

In 2005, James and Jonathan Bach presented a paper at the STAR West conference on exploratory

[6] http://www.iht.com/articles/2004/10/16/mmole_ed3_.php
[7] http://www.testingeducation.org/BBST/exams/NotesForStudents.htm

dynamics, skills and tactics. Michael Kelly led a session in which we further developed this list at Consultants' Camp 2006.

Several of the points in this list—especially modeling, questioning, chartering, observing, generating and elaborating, abandoning and recovering, conjecturing, and of course recording and reporting—can be aided by the kinds of things that we do in notebooks: writing, sketching, listing, speculating, brainstorming, and journaling. Much of what we think of as history or scientific discovery was first recorded in notebooks. We see a pattern of writing and keeping notes in situations and disciplines where learning and discovery are involved. A variety of models helps us to appreciate a problem (and potentially its solution) from more angles. Thinking about a problem is different from uttering it, which is still different from sketching it or writing prose about it. The direct interaction with the ink and the paper gives us a tactile mode to supplement the visual, and the fact that handwriting is, for many people, slower than typing, may slow down our thought processes in beneficial ways. A notebook gives us a medium in which to record, re-model, and reflect. These are, in my view, essential testing skills and tactics.

From a historical perspective, we are aware that Leonardo was a great thinker because he left notebooks, but it's also reasonable to consider that Leonardo may have been a great thinker at least in part because he *used* notebooks.

Who Uses Notebooks?

Inventors, scientists, explorers, artists, writers, and students have made notebook work part of their creative process, leaving both themselves and us with records of their thought processes.

Leonardo da Vinci's notebooks are among the most famous books in history, and also at this writing the most expensive; one of them, the Codex Leicester, was purchased in 1994 for $30.8 million by a certain ex-programmer from the Pacific Northwest[8]. Leonardo left approximately 13,000 pages of daily notes and drawings. I was lucky enough to see one recently—the Codex Foster, from the collection of the Victoria and Albert Museum.

[8] Incidentally, the exhibit notes and catalog suggested that Leonardo didn't intend to encrypt his work via the mirror writing for which he was so famous; he wrote backwards because he was left-handed, and writing normally would smudge the ink.

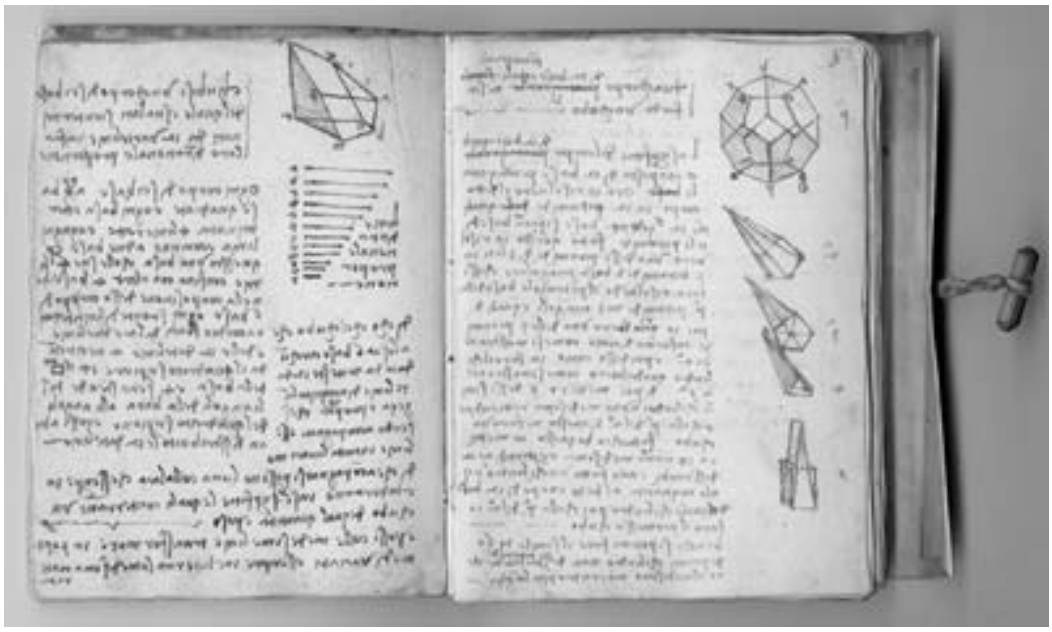


Figure 2: Leonardo da Vinci, The Codex Foster

As a man of the Renaissance, Leonardo blurred the lines between artist, scientist, engineer, and inventor[9], and his notebooks reflect this. Leonardo collects ideas and drawings, but also puzzles, aphorisms, plans, observations. They are enormously eclectic, reflecting an exploratory outlook on the world. As such, his notebooks are surprisingly similar to the notebook patterns of exploratory testers described below, though none has consciously followed Leonardo's paradigms or principles, so far as I know. The form factor is also startlingly similar to the smaller Moleskine notebooks. Obviously, the significance of our work pales next to Leonardo's, but is there some intrinsic relationship between exploratory thinking and the notebook as a medium?

What Do I Use My Notebook For?

I've been keeping three separate notebooks. My large-format book contains notes that I take during sessions at conferences and workshops. It tends to be tidier and better-organized. My small-format book is a ready place to record pretty much anything that I find interesting. Here are some examples:

Lists of things, as brainstorms or catalogs. My current lists include testing heuristics; reifications; and test ideas. These lists are accessible and can be added to or referenced at any time. This is my favorite use of the Moleskine—as a portable thinking and storage tool.

"Fieldstones" and blog entries. Collections of observations; the odd rant; memorable quotes; aphorisms. The term "fieldstone" is taken from Gerald M. Weinberg's book Weinberg on Writing: The Fieldstone Method. In the book, Jerry uses the metaphor of the pile of stones that are pulled from the field as you clear it; then you assemble a wall or a building from the fieldstones.[10] I collect ideas for articles and blog entries and develop them later.

[9] How To Think Like Leonardo da Vinci
[10] Weinberg, Gerald M., *Weinberg on Writing: The Fieldstone Method.*

Logs of testing sessions. These are often impromptu, used primarily to practice testing and reporting, to reflect and learn later, and to teach the process. A couple of examples follow below.

Meeting notes. He who controls the minutes controls history, and he who controls history controls the world.

Ultra-Portable PowerPoints. These are one-page presentations that typically involve a table or a diagram. This is handy for the cases in which I'd like to make a point to a colleague or client. Since the listener focuses on the data and on my story, and not on what Edward Tufte[11] calls "chartjunk", the portable PowerPoints may be more compelling than the real thing.

Mind maps and diagrams. I use these for planning and visualization purposes. I need to practice them more. I did use a mind map to prepare this presentation.

Notes collected as I'm teaching. When a student does something clever during a testing exercise, I don't want to interrupt the flow, but I do want to keep track of it so that I can recount it to the class and give recognition and appreciation to the person who did it. Moreover, about half the time this results in some improvement to our course materials[12], so a notebook entry is very handy.

Action items, reminders, and random notes. Sometimes the notebook is the handiest piece of paper around, so I scribble something down on a free page—contact names (for entry later), reminders to send something to someone; shopping lists.

Stuff in the pocket. I keep receipts and business cards (so I don't lose them). I also have a magic trick that I use as a testing exercise that fits perfectly into the pocket.

I try to remember to put a title and date on each page. Lately I've been slipping somewhat, especially on the random notes pages.

I've been using a second large-format notebook for notes on books that I'm studying. I haven't kept this up so well. It's better organized than my small format book, but my small format book is handy more often, so notes about books—and quotes from them—tend to go in that.

I'm not doing journaling, but the notebooks seem to remind me that, some day, I will. Our society doesn't seem to have the same diary tradition as it used to; web logs retrieve this idea. Several of my colleagues do keep personal journals.

How Do Other Exploratory Testers Use Notebooks?

I've done a very informal and decidedly unscientific survey of some of my colleagues, especially those who are exploratory testers.

---

[11] Tufte, Edward, *Envisioning Information*. Graphics Press, Chesire, Connecticut, 1990.
[12] Bach, James, and Bolton, Michael, *Rapid Software Testing*. http://www.satisfice.com/rst.pdf.

Adam White reports, “My notebook is my *life*. It's how I keep track of things I have to do. It supplements my memory so that I don't waste brain power on remembering to remember something. I just record it and move on.

“I have found a method of taking notes that brings my attention to things. If someone tells me about a book then I will write "Book" and underline it twice. Then when flipping back through my notes I can see that I have a reference to a book that I thought was interesting at some point in time. I use this process for other things like blogs, websites, key ideas, quotes etc. It makes organizing information after the fact very easy.”

Adam reports similar experiences to my own in how he came to use Moleskines. He too observed Jon Bach and James Bach using Moleskine notebooks; he too uses a selection of books—one large-form for work, one large-form for personal journaling, and a small one for portability and availability. He also says that the elastic helps to prevent him from losing pens.

Jonathan Kohl also reports that he uses notebooks constantly. “My favorite is my Moleskine, but I also use other things for taking notes. With my Moleskine, I capture test ideas; article ideas; diagrams or models I am working on for articles; teaching materials, or some other reason for an explanation to others; and testing notes[13]. I have a couple of notes to help focus me, and the rest are ideas, impressions, and the starred items are bugs. I translated the bugs into bug reports in a fault tracking system, and the other notes into a document on risk areas. For client work, I don't usually use my Moleskine for testing, since they may want my notes.” This is an important point for contractors and full-time employees; your notebook may be considered a work product—and therefore the property of your company—if you use it at work, or for work.

“I also use index cards (preferably post-it note index cards), primarily for bug reports,” continues Jonathan. “My test area is often full of post-its, each a bug, at the end of a morning or afternoon testing session. Over time, I arrange the post-its according to groups, and log them into a bug tracker or on story cards (if doing XP.) When I am doing test automation/test toolsmith work, I use story cards for features or other tasks, and others for bugs.”

Jonathan also uses graph-paper pads for notes that he doesn't need to keep. They contain rough session and testing notes; diagrams, scrawls, models, or things that he is trying to understand better; analysis notes, interview points, and anything else he’s interested in capturing. “These notes are illegible to most people other than me, and I summarize them and put what is needed into something more permanent.” This is also an important point about documentation in general: sometimes documentation is a product—a deliverable, or something that you show to or share with someone else. At other times, documentation is a tool—a personal aid to memory or thought processes.

“I worked with engineers a lot starting out, so I have a black notebook that I use to record my time and tasks each day. I started doing this as an employee, and do it as a consultant now as well.”

Fiona Charles also keeps a project-specific notebook. She uses a large form factor, so that it can accommodate 8½ x11 pages pasted into it. She also pastes a plastic pocket, a calendar, and loose notes from pre-kickoff meetings—she says that a glue stick is an essential part of the kit. In the

[13]Jonathan provides an example at http://www.kohl.ca/articles/ExploratoryTesting_MusicofInvestigation.pdf

notebook, she records conversations with clients and others in the project community. She uses clear termination line for dates, sets of notes, and "think pages."

Jerry Weinberg also uses project notebooks. On the first page, he places his name, his contact information, and offer of a reward for the safe return of the book. On the facing page, he keeps a list of contact info for important people to the project. On the subsequent pages, he keeps a daily log from the front of the book forwards. He keeps a separate list of learnings from the back of the book backward, until the two sections collide somewhere in the middle; then he starts a new book. "I always date the learnings," he says. "In fact, I date everything. You never know when this will be useful data." Like me, he never tears a page out.

Jerry is also a strong advocate of journaling[14]. For one thing, he treats starting journaling—and the reader's reaction to it—as an exercise in learning about effecting change in ourselves and in other people. "One great advantage of the journal method," he says, "is that unlike a book or a lecture, everything in it is relevant to *you*. Because each person's learning is personal, I can't you what you'll learn, but I can guarantee that you'll learn *something*." That's been my experience; the notebook reflects me and what I'm learning. It's also interesting to ask myself about the things, or kinds of things, that I *haven't* put it.

Jon Bach reports that he uses his notebooks in several modes. "'Log file', to capture the flow of my testing; 'epiphany trap', to capture "a ha!" moments (denoted by a star with a circle around it); diagrams and models—for example, the squiggle diagram when James and I first roughed out Session-Based Test Management; to-do lists—lots and of lots them, which eventually get put into Microsoft Outlook's Task Manager with a date and deadline—reminders, flight, hotel, taxi info when traveling, and phone numbers; quotes from colleagues, book references, URLs; blog ideas, brainstorms, ideas for classes, abstracts for new talks I want to do; heuristics, mnemonics; puzzles and their solutions (like on a math exam that says "show your work"); personal journal entries (especially on a plane); letters to my wife and child -- to clear my head after some heinous testing problem I might need a break from."

Jon also identifies as significant the paradigm "'NTSB Investigator.' I'll look back on my old notes for lost items to rescue—things that are may have become more important than when I first captured them because of emergent context. You would never crack open the black box of an airplane after a successful flight, but what if there was a systemic pattern of silent failures just waiting to culminate in a HUGE failure? *Then* you might look at data for a successful flight and be on the lookout for pathologies."

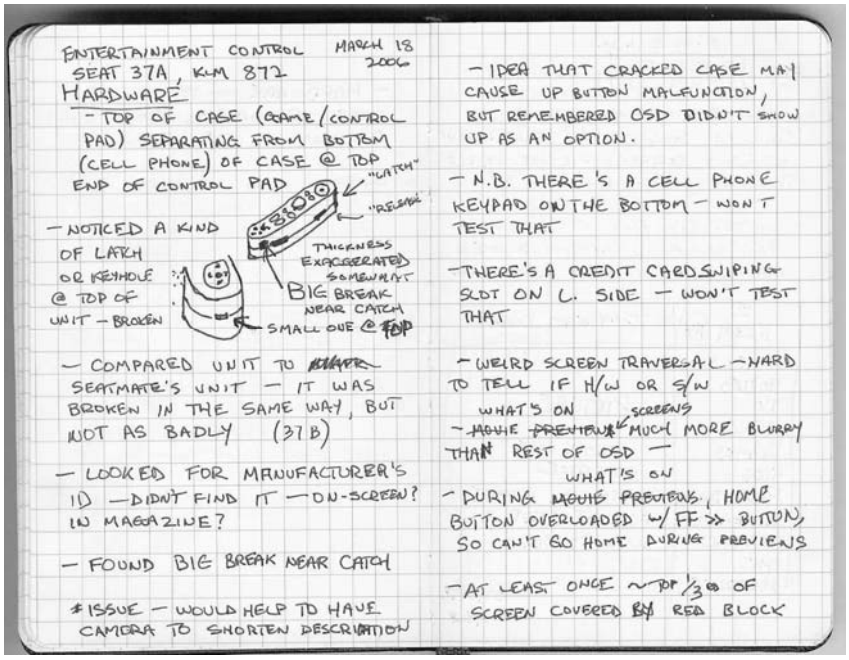
Example: An Impromptu Exploratory Testing Session

I flew from Delhi to Amsterdam. I was delighted to see that the plane was equipped with a personal in-flight entertainment system, which meant that I could choose my own movies or TV to watch. As it happened, I got other entertainment from the system that I wouldn't have predicted.

The system was menu-driven. I went to the page that listed the movies that were available, and after scrolling around a bit, I found that the "Up" button on the controller didn't work. I then inspected the controller unit, and found that it was cracked in a couple of places. Both of the

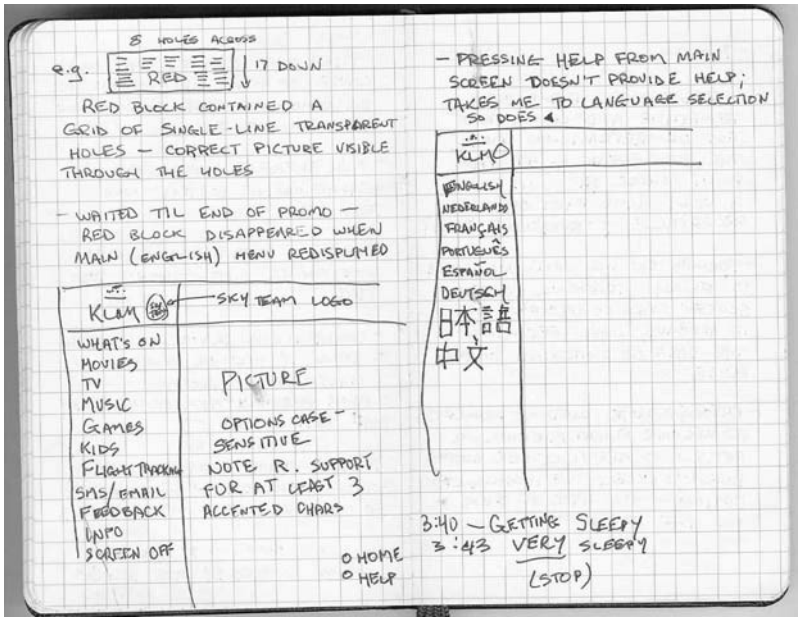
[14] Becoming a Technical Leader, pp. 80-85

cracks were associated with the mechanism that returned the unit, via a retractable cord, to a receptacle in the side of the seat. I found that if I held the controller just so, then I could get around the hardware—but the software failed me. That is, I found lots of bugs. I realized that this was an opportunity to collect, exercise, and demonstrate the sorts of note-taking that I might perform when I'm testing a product for the first time. Here are the entries from my Moleskine, and some notes about my notes.



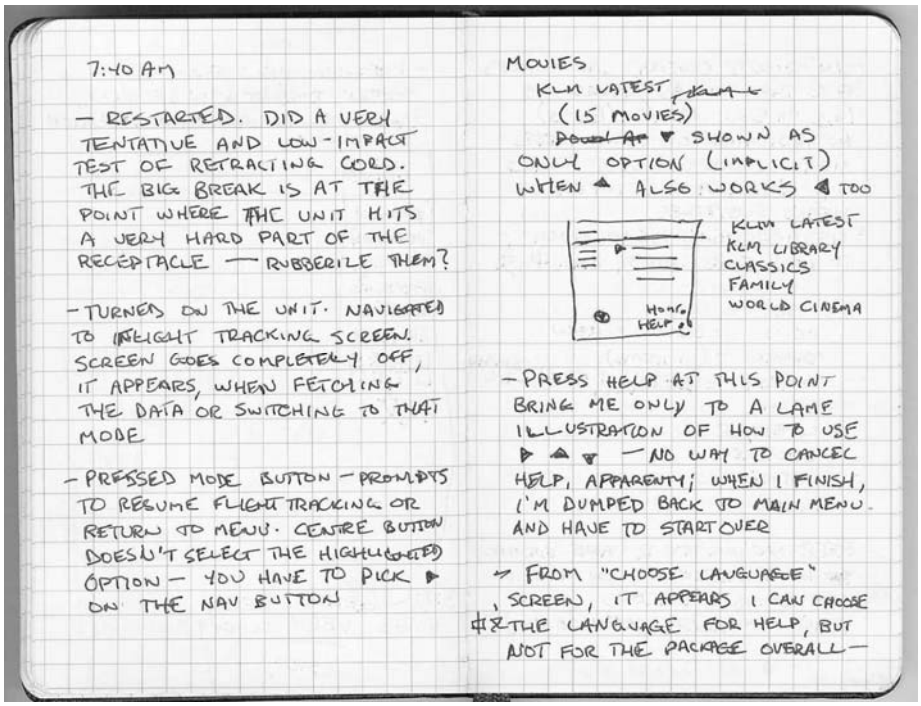
When I take notes like this, they're a tool, not a product. I don't expect to show them to anyone else; it's a possibility, but the principal purposes are to allow me to remember what I did and what I found, and to guide a discussion about it with someone who's interested.

I don't draw well, but I'm slowly getting better at sketching with some practice. I find that I can sketch better when I'm willing to tolerate mistakes.



In the description of the red block, at the top of the left page, I failed to mention that this red block appeared when I went right to the "What's On" section after starting the system. It didn't reproduce.

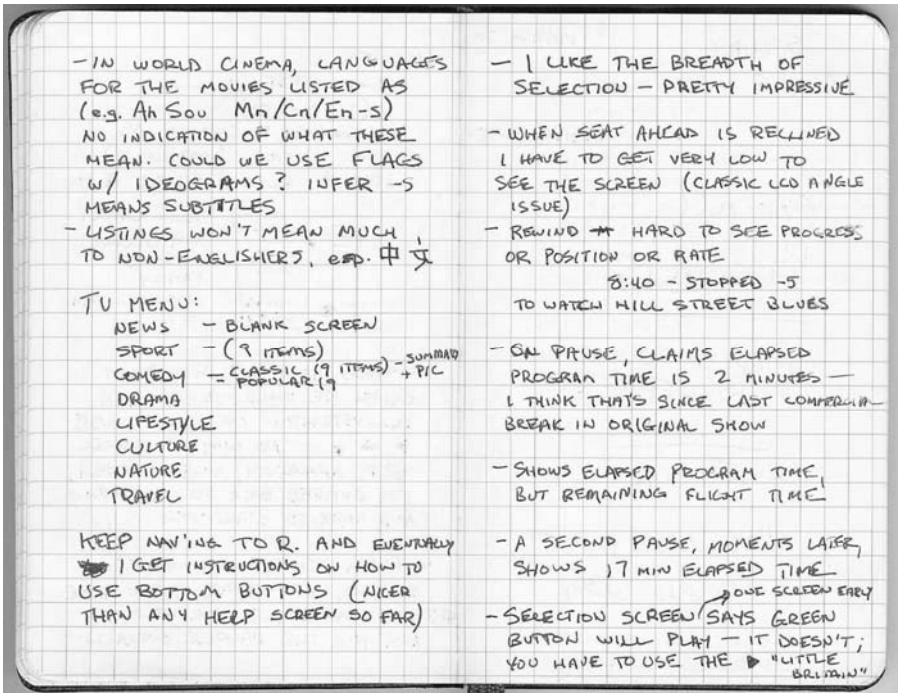
Whenever I look back on my notes, I recognize things that I missed. If they're important, I write them down as soon as I realize it. If they're not important, I don't bother. I don't feel bad about it either way; I try always to get better at it, but testers aren't omniscient. Note "getting sleepy"—if I keep notes on my own mental or emotional state, they might suggest areas that I should revisit later. One example here: on the first page of these notes, I mentioned that I couldn't find a way to contact the maker of the entertainment system. I should have recognized the "Feedback" and "Info" menu items, but I didn't; I noticed them afterwards.



After a few hours of rest, I woke up and started testing again.

Jon Bach recently pointed out to me that, in early exploration, it's often better to start not by looking for bugs, but rather by trying to build a model of the item under test. That suggests looking for the positives in the product, and following the happy path. I find that it's easy for me to fall into the trap of finding and reporting bugs. These notes reflect that I *did* fall into the trap, but I also tried to check in and return to modeling from time to time. At the end of this very informal and completely freestyle session, I had gone a long way towards developing my model and identifying various testing issues. In addition, I had found many irritating bugs.

Why perform and record testing like this? The session and these notes, combined with a discussion with the project owner, might be used as the first iteration in the process of determining an overall (and perhaps more formal) strategy for testing this product. The notes have also been a useful basis for my own introspection and critique



of my performance, and to show others some of my though process through an exploratory testing session.

A More Formal Structure for Exploratory Testing

Police forces all over the world use notebooks of some description, typically in a way that is considerably more formalized. This is important, since police notebooks will be used as evidence in court cases. For this reason, police are trained and required to keep their notebooks using elements of a more formal structure, including time of day; exact or nearest-to location; the offence or occurrence observed; the names and addresses of offenders, victims or witnesses; action taken by the officer involved (e.g. arrests), and details of conversations and other observations. (The object of the exercise here is not to turn testers into police, but to take useful insights from the process of more formal note-taking.)

How can we help to make testing similarly accountable? Session-Based Test Management (SBTM), invented by James and Jonathan Bach in 2000 is one possible answer. SBTM has as its hallmark four elements:

- Charter
- Time Box
- Reviewable Result
- Debriefing

The *charter* is a one- to three-sentence mission for a testing session. The charter is designed to be open-ended and inclusive, prompting the tester to explore the application and affording opportunities for variation. Charters are not meant to be comprehensive descriptions of what should be done, but the total set of charters for the entire project should include everything that is reasonably testable.

The *time box* is some period of time between 45 minutes and 2 ¼ hours, where a short session is one hour (+/- 15 minutes), a long session is two, and a normal session is 90 minutes. The intention here is to make the session short enough for accurate reporting, changes in plans (such as a session being impossible due to a broken build, or a session changing its charter because of a new priority), but long enough to perform appropriate setup, to get some good testing in, and to make debriefing efficient. Excessive precision in timing is discouraged; anything to the nearest five or ten minutes will do. If your managers, clients, or auditors are supervising you more closely than this,

The reviewable result takes the form of a *session sheet*, a page of text (typically ASCII) that follows a formal structure. This structure includes:

- Charter
- Coverage areas (not code coverage; typically product areas, product elements, quality criteria, or test techniques)
- Start Time
- Tester Name(s)
- Time Breakdown
    - session duration (long, normal, or short)

    - test design and execution (as a percentage of the total on-charter time)
    - bug investigation and reporting (as a percentage of the total on-charter time)
    - session setup (as a percentage of the total on-charter time)
    - charter/opportunity (expressed as a percentage of the total session, where opportunity time does not fit under the current charter, but is nonetheless useful testing work)
- Data Files
- Test Notes
- Bugs (where a "bug" is a problem that the tester and the test manager reasonably believe represents a threat to the value of the product)
- Issues (where an "issue" is a problem that threatens the value of the testing process—missing information, tools that are unavailable, expertise that might be required, questions that the tester might develop through the course of the session)

There are two reasons for this structure. The first is simply to provide a sense of order and completeness for the report and the debrief. The second is to allow a scripting tool to parse tagged information from the session sheets, such that the information can be sent to other applications for bug reporting, coverage information, and inquiry-oriented metrics gathering. The SBTM package, available at http://www.satisfice.com/sbtm, features a prototype set of batch files and Perl scripts to perform these tasks, with output going to tables and charts in an Excel spreadsheet.

The *debrief* is a conversation between the tester[15] who performed the session and someone else—ideally a test lead or a test manager, but perhaps simply another tester. In the debrief, the session sheet is checked to make sure that it's readable and understandable; the manager and the tester discuss the bugs and issues that were found; the manager makes sure that the protocol is being followed; and coaching, mentoring, and collaboration happen. A typical debrief will last between five to ten minutes, but several things may add to the length. Incomplete or poorly-written session sheets produced by testers new to the approach will prompt more questions until the tester learns the protocol. A highly complex or risky product area, a large number of bugs or issues, or an unfamiliar product may also lead to longer conversations.

Several organizations have reported that scheduling time for debriefings is difficult when there are more than three or four testers reporting to the test manager or test lead, or when the test manager has other responsibilities. In such cases, it may be possible to have the testers debrief each other.

At one organization where I did some consulting work, the test manager was also responsible for requirements development and business analysis, and so was frequently unavailable for debriefings. The team chose to use a round-robin testing and debriefing system. For a given charter, Tester A performed the session, Tester B debriefed Tester A, and at regression testing time, Tester C took a handful of sheets and used them as a point of departure for designing and executing tests. For the next charter, Tester B performed the testing, Tester C the debrief, and Tester A the regression; and so forth. Using this system, each tester learned about the product and shared information with others by a variety of means—interaction with the product, conversation in the debrief, and written session sheets. The entire team reported summaries of

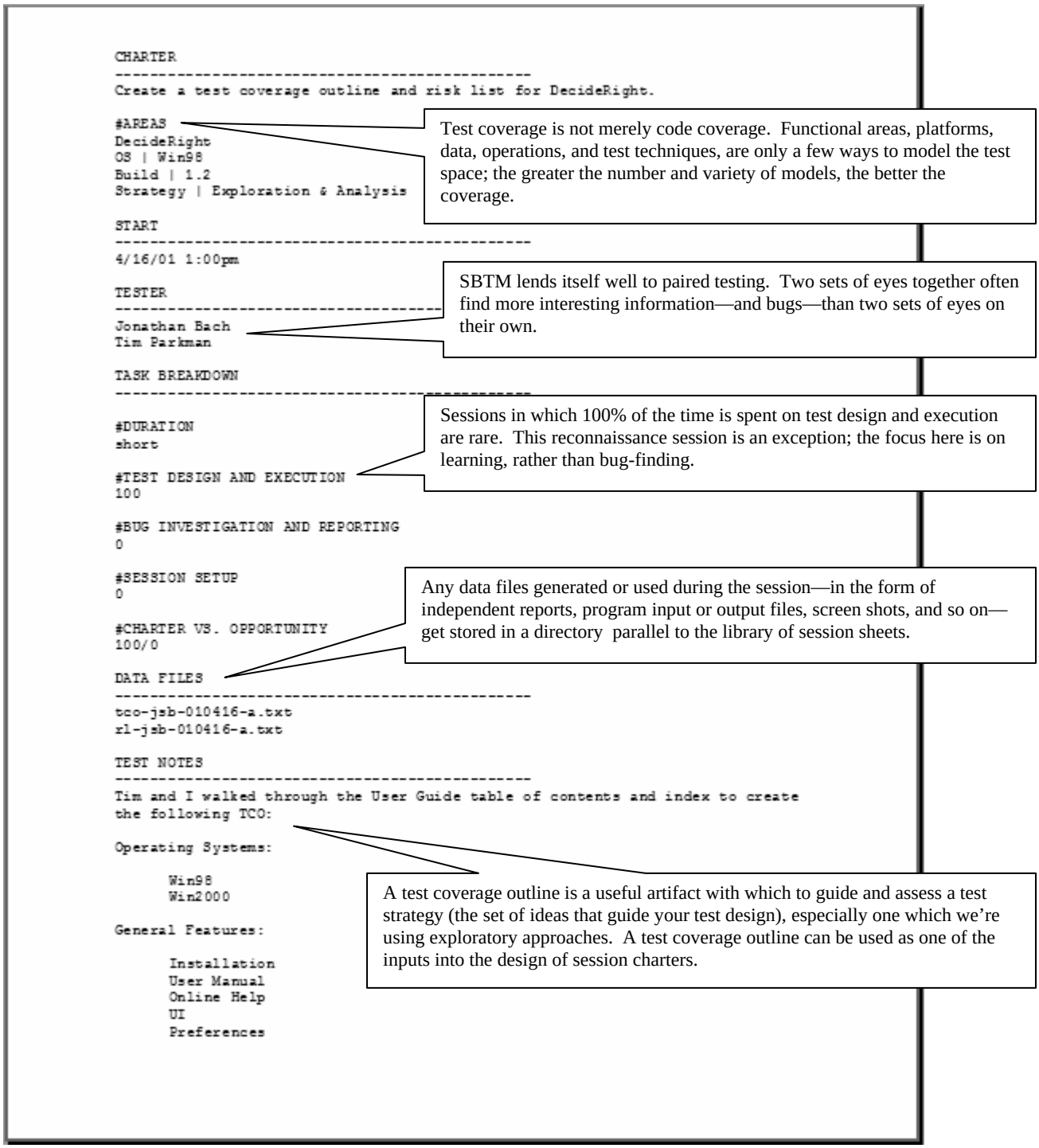
[15] Or "testers"; SBTM can be used with paired testers.

the debriefings to the test manager when he was not available, and simply debriefed directly with him when he was.

Two example session sheets follow. The first is an account of an early phase of exploratory testing, in which the testers have been given the charter to create a test coverage outline and a risk list. These artifacts themselves can be very useful, lightweight documents that help to guide and assess test strategy. Here the emphasis is on learning about the product, rather than searching for bugs.

The second is an account of a later stage of testing, in which the tester has sufficient knowledge about the product to perform a more targeted investigation. In this session, he finds and reports several bugs and issues. He identifies moments at which he had new test ideas and the motivations for following the lines of investigation.

Example: Session Sheet for a Reconnaissance Session



```
CHARTER
-----------------------------------------------
Create a test coverage outline and risk list for DecideRight.

#AREAS
DecideRight
OS | Win98
Build | 1.2
Strategy | Exploration & Analysis

START
-----------------------------------------------
4/16/01 1:00pm

TESTER
-----------------------------------------------
Jonathan Bach
Tim Parkman

TASK BREAKDOWN
-----------------------------------------------

#DURATION
short

#TEST DESIGN AND EXECUTION
100

#BUG INVESTIGATION AND REPORTING
0

#SESSION SETUP
0

#CHARTER VS. OPPORTUNITY
100/0

DATA FILES
-----------------------------------------------
tco-jsb-010416-a.txt
rl-jsb-010416-a.txt

TEST NOTES
-----------------------------------------------
Tim and I walked through the User Guide table of contents and index to create
the following TCO:

Operating Systems:

      Win98
      Win2000

General Features:

      Installation
      User Manual
      Online Help
      UI
      Preferences
```

Test coverage is not merely code coverage. Functional areas, platforms, data, operations, and test techniques, are only a few ways to model the test space; the greater the number and variety of models, the better the coverage.

SBTM lends itself well to paired testing. Two sets of eyes together often find more interesting information—and bugs—than two sets of eyes on their own.

Sessions in which 100% of the time is spent on test design and execution are rare. This reconnaissance session is an exception; the focus here is on learning, rather than bug-finding.

Any data files generated or used during the session—in the form of independent reports, program input or output files, screen shots, and so on—get stored in a directory parallel to the library of session sheets.

A test coverage outline is a useful artifact with which to guide and assess a test strategy (the set of ideas that guide your test design), especially one which we're using exploratory approaches. A test coverage outline can be used as one of the inputs into the design of session charters.

```
Prominent Windows:

      Main Table window
      Criteria Weights window
      Option Ratings window
      Documents window
      Start-up window

Managers and Wizards:

      DecideRight Advisor
      Category Label Editor
      Numeric Editor
      Scenario Manager
      Report Generator
      QuickBuild

Decision Elements:

      Language Elements
      Preferences
      Sensitivity Indicators
      Weighting
      Input Options
      Decision Table
      Options Ratings
      Baseline

Interoperability:

      OLE
      Import / Export
      Graphs
      Printing

Risk list for DecideRight:

* It will suggest the wrong decisions.
* People will use the product incorrectly.
* It will incorrectly compare scenarios.
* Scenarios may become corrupted.
* It will not be able to handle complex decisions.

BUGS
-------------------------------------------
#N/A

ISSUES
-------------------------------------------
#ISSUE
Manual mentions different platforms (Win 3.1, WFW, and WinNT 3.51) and does
not mention Win2000. We think Win 2000 is important to test on and that the
older OSes are no longer meaningful.

#ISSUE
We did this analysis on Win98.  I have no data to suggest that features may
be different on other operating systems, but I'm not sure about that.
```

A risk list is another useful tool to help guide a test strategy. The risk list can be as long or as short as you like; it can also be broken down by product or coverage areas.

"Issues" are problems that threaten the value of the testing process. Issues may include concerns, requests for missing information, a call for tools or extra resources, pleas for testability. In addition, if a tester is highly uncertain whether something is a bug, that can be reported here.

Example: Session Sheet for a Bug-Finding Session

```
CHARTER
-----------------------------------------------
Explore a decision created with QuickBuild -- the wizard that guides the user
through the options, criteria, and weights needed to calculate the best
decision.

#AREAS
OS | Win98
Build | 1.2
DecideRight | QuickBuild
DecideRight | Report Generator
Strategy | Exploration & Analysis

START
-----------------------------------------------
4/17/01 1:30pm

TESTER
-----------------------------------------------
Jonathan Bach

TASK BREAKDOWN
-----------------------------------------------

#DURATION
short

#TEST DESIGN AND EXECUTION
70

#BUG INVESTIGATION AND REPORTING
20

#SESSION SETUP
10

#CHARTER VS. OPPORTUNITY
90/10

DATA FILES
-----------------------------------------------
food.drd
food.rtf
food2.rtf
food3.rtf

TEST NOTES
-----------------------------------------------

Created a new "decision" that I already knew the answer to: What kind of food
to have for dinner?  I wanted to see if DecideRight could reach the same
conclusion.

Options:
  * American
  * Chinese
  * Mexican
  * Italian
  * Pizza
  * Nothing
```

A single session can cover more than one functional area of the product. Here the testers obtain coverage on both the QuickBuild wizard and the report generator

The goal of any testing session is to obtain coverage—test design and execution, in which we learn good and bad things about the product. Bug investigation (learning things about a particular bug) and setup (preparing to test), while valuable, are interruptions to this primary goal. The session sheet tracks these three categories as inquiry metrics—metrics that are designed to prompt questions, rather than to drive decisions. If we're doing multiple things at once, we report the highest-priority activity first; if it happens that we're testing as we're investigating a bug or setting up, we account for that as testing.

Test notes tend to be more valuable when they include the motivation for a given test, or other clues as to the tester's mindset. The test notes—the core of the session sheet—help us to tell the testing story: what we tested, why we tested it, and why we believe that our testing were good enough.

Information generated from the session sheets can be fed back into the estimation process.

- First, we'll cast a set of charters representing the coverage that we'd like to obtain in a given test cycle. (Let's say, for this example, 80 charters).
- Second, we'll look at the number of testers that we have available. (Let's say 4.)
- Typically we will project that a tester can accomplish three sessions per day, considering that a session is about 90 minutes long, and that time will be spent during the day on email, meetings, breaks, and the like.
- We must also take into account the productivity of the testing effort. Productivity is defined here *the percentage of the tester's time spent*, in a given session, *on coverage*—that is, on test

design and execution. Bug investigation is very important, but it reduces the amount of coverage that we can obtain about the product during the session. It doesn't tell us more about the product, even though it may tell us something useful about a particular bug. Similarly, setup is important, but it's *preparing to test*, rather than *testing*; time spent on setup is time that we can't spend obtaining coverage. (If we're setting up and testing at the same time, we account for this time as testing. At the very beginning of the project, we might estimate 66% productivity, with the other third of the time spent on setup and bug investigation. This gives us our estimate for the cycle:

80 charters x .66 productivity x 4 testers x 3 sessions per day = **10 days**

```
Criteria:
  * price
  * taste
  * convenience
  * last had
  * health

Report notes:

FOOD.RTF

  "Nothing" appears to be the best choice even though my answer was "Pizza."

  ??? How did it reach this calculation?  I would like to devote a session to
this.
  ??? what's the difference between N/A and ??? values
  (see BUG 1 below)

FOOD2.RTF

  * DecideRight showed my 6 choices (options) in order of importance but does
not describe why it ranked them (see BUG below)
  * DecideRight did show my criteria ranked in order, however

FOOD3.RTF

  Created this file because I had a test idea: add some new criteria options
to an existing decision table and re-run the report.

      Result: PASS -- changes get reflected and recalc'ed

      Found a problem in the formatting, though (see BUG 3)

  Test Idea: does eliminating unknown values remove the disclaimer at the top
of the report: ("Warning!  Some elements in the decision table which
generated this report are labeled "To Be Rated" or "Unknown," and it may
therefore be premature to draw conclusions from the data.")

      Result: PASS -- the disclaimer was removed.


OPPORTUNITY: Noticed that pushpin icon on toolbar for decision table does
nothing when no option is highlighted. (see BUG 4 below)

Session interrupted by phone call.  Will pick this up in other session
tomorrow.

Conclusions: I'd like another session or two to learn the algorithm
DecideRight uses to make decisions.  Then I can verify that the report is
accurate.


BUGS
----------
#BUG 1
Not dragging the weight slider for a criteria item leads to an ??? instead of
max "Poor"

Repro:
1 -- launch QuickBuild to create a new decision
2 -- put in some options | Next
```

Exploratory testing, by intention, reduces emphasis on specific predicted results, in order to reduce the risk of inattentional blindness. By giving a more open mandate to the tester, the approach affords better opportunities to spot unanticipated problems.

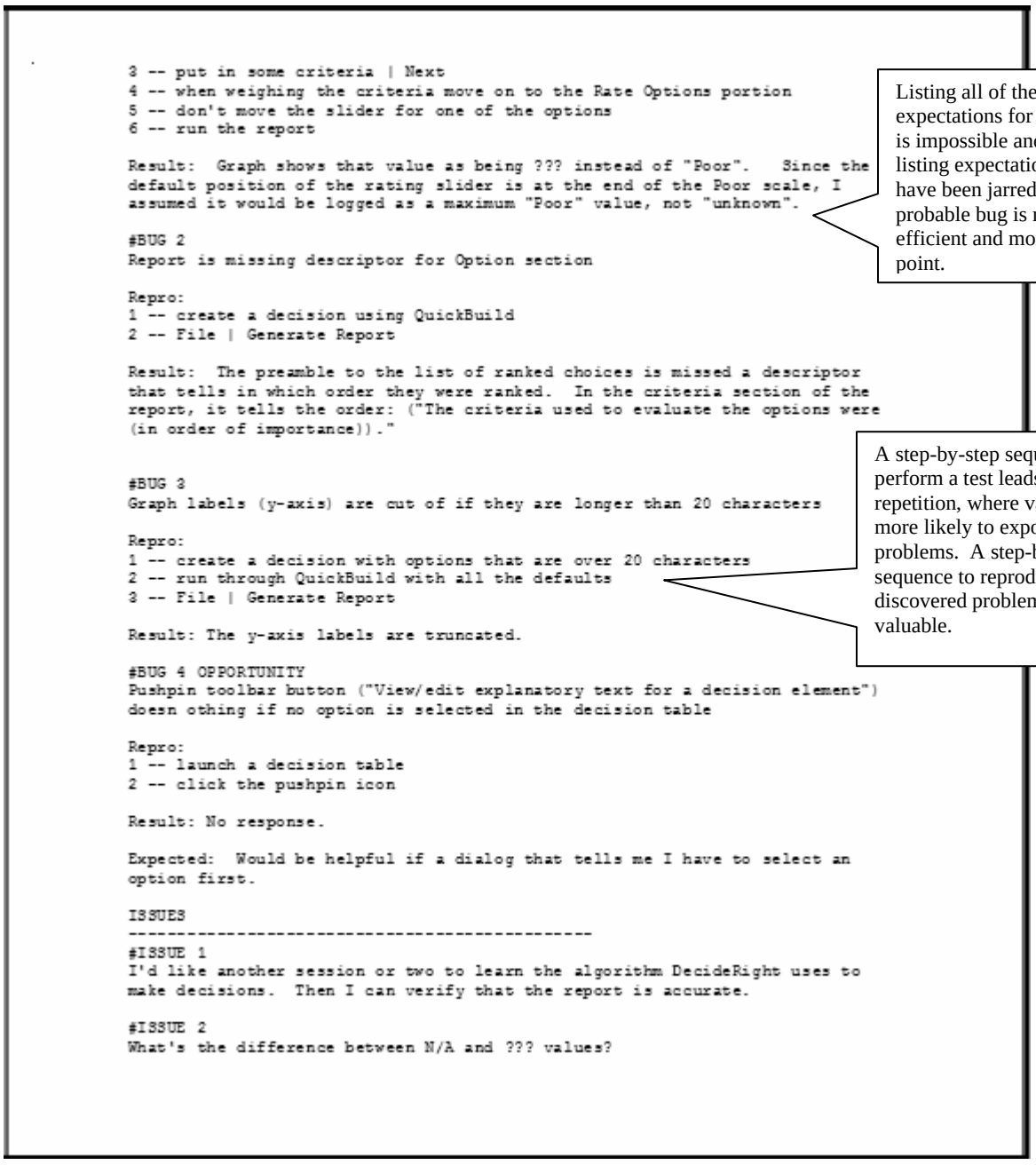
New test ideas come up all the time in an exploratory testing session. The tester is empowered to act on them right away.

"Opportunity" work is testing done outside the scope of the current charter. Again, testers are both empowered and encouraged to notice and investigate problems as they find them, and to account for the time in the session sheet.

The #BUG tag allows a text-processing tool to transfer this information to a master bug list in a bug tracking system, an ASCII file, or an Excel spreadsheet.

"Some options" might be vague here; more likely, based on our knowledge of the tester, the specific options are be unimportant, and thus it might be wasteful and even misleading to provide them. The tester, the test manager, and product team develop consensus through experience and mentoring on how to note just what's important, no less and no more.

When new information comes in—often in the form of new productivity data—we change one or more factors in the estimate, typically by increasing or decreasing the number of testers, increasing or reducing the scope of the charters, or shortening or lengthening the cycle.



```
3 -- put in some criteria | Next
4 -- when weighing the criteria move on to the Rate Options portion
5 -- don't move the slider for one of the options
6 -- run the report

Result:  Graph shows that value as being ??? instead of "Poor".   Since the
default position of the rating slider is at the end of the Poor scale, I
assumed it would be logged as a maximum "Poor" value, not "unknown".

#BUG 2
Report is missing descriptor for Option section

Repro:
1 -- create a decision using QuickBuild
2 -- File | Generate Report

Result:  The preamble to the list of ranked choices is missed a descriptor
that tells in which order they were ranked.  In the criteria section of the
report, it tells the order: ("The criteria used to evaluate the options were
(in order of importance))."


#BUG 3
Graph labels (y-axis) are cut of if they are longer than 20 characters

Repro:
1 -- create a decision with options that are over 20 characters
2 -- run through QuickBuild with all the defaults
3 -- File | Generate Report

Result: The y-axis labels are truncated.

#BUG 4 OPPORTUNITY
Pushpin toolbar button ("View/edit explanatory text for a decision element")
doesn othing if no option is selected in the decision table

Repro:
1 -- launch a decision table
2 -- click the pushpin icon

Result: No response.

Expected:  Would be helpful if a dialog that tells me I have to select an
option first.

ISSUES
-----------------------------------------------
#ISSUE 1
I'd like another session or two to learn the algorithm DecideRight uses to
make decisions.  Then I can verify that the report is accurate.

#ISSUE 2
What's the difference between N/A and ??? values?
```

Listing all of the possible expectations for a given test is impossible and pointless; listing expectations that have been jarred by a probable bug is more efficient and more to the point.

A step-by-step sequence to perform a test leads to repetition, where variation is more likely to expose problems. A step-by-step sequence to reproduce a discovered problem is more valuable.

Some questions have been raised as to whether exploratory approaches like SBTM are acceptable for high-risk or regulated industries. We have seen SBTM used in a wide range of contexts, including financial institutions, medical imaging systems, telecommunications, and hardware devices.

Some also question whether session sheets meet the standards for the accountability of bank auditors. One auditor's liaison with whom I have spoken indicates that his auditors would not be interested in the entire session sheet; instead, he maintained, "What the auditors really want to

see is the charter, and they want to be sure that there's been a second set of eyes on the process. They don't have the time or the inclination to look at each line in the Test Notes section."

Conclusion

Notebooks have been used by people in the arts, sciences, and skilled professions for centuries. Many exploratory testers may benefit from the practice of taking notes, sketching, diagramming, and the like, and then using the gathered information for retrospection and reflection.

One of the principal concerns of test managers and project managers with respect to exploratory testing is that it is fundamentally unaccountable or unmanageable. Yet police, doctors, pilots, lawyers and all kinds of skilled professions have learned to deal with problem of reporting unpredictable information in various forms by developing note-taking skills. Seven years of positive experience with session-based test management suggests that it is a useful approach, in many contexts, to the process of recording and reporting exploratory testing.

Thanks to Launi Mead and Doug Whitney for their review of this paper.