# DevelopSense

## Rapid Software Testing Reading, Resources and Tools

Compiled by Michael Bolton and James Bach
Last revised January 31, 2006

To learn about **finding bugs and risks**, read

- Lessons Learned in Software Testing by Cem Kaner, James Bach, and Bret Pettichord.  293 bite-sized lessons from three of the leaders of the Context-Driven School of Software Testing.
- Testing Computer Software by Cem Kaner, Jack Falk, and Hung Quoc Nguyen.  The book that, for many testers, started it all.  The best-selling testing book in history.  Somewhat out of date these days, since it predates the rise of Windows and the rise of the Internet, but a very important text in terms of the *thinking* part of software testing.  Also contains an excellent (if overwhelming) example of a bug and testing taxonomy.
- How to Break Software by Whittaker, and How to Break Software Security by Whittaker and Thompson.  Two wonderful testing books that actually (gasp!) show specific bugs and the classes of tests that expose them.  This book presents a useful perspective for finding problems—identifying customers of the application as the end-user, the file system, the operating system, and application programming interfaces.
- Hacking Exposed by Stuart McClure, Joel Scambray, and George Kurtz, and Hacking Web Applications Exposed by Joel Scambray and Mike Shema.  Hackers and testers have a lot in common in terms of the approaches that they can use to find out how software really works and how to expose its weaknesses.  Testers owe hackers a favour, in a way, since the work of the former is risk that underscores the value of the latter.

To learn about **testing philosophy**, read

- The Pleasure of Finding Things Out by Richard Feynman.  In particular, read his Appendix to the Rogers Commission's report on the Challenger.
- Surely You're Joking, Dr. Feynman!  Adventures of a Curious Character by Richard Feynman.  Feynman's curiosity drove his apparently insatiable desire to find out about the world, often in the same manner that a tester or hacker might.  This book contains (among other things) accounts of Feynman's safecracking exploits at Los Alamos.
- What Do You Care What Other People Think? by Richard Feynman.  The first page of this book alone—in which Feynman notes that learning about things only adds to a deeper appreciation of them—would make it a worthwhile recommendation.
- Introduction to General Systems Thinking by Jerry Weinberg
- Are Your Lights On by Don Gause and Jerry Weinberg.  A more lightweight approach to some of the concepts in the latter book.
- Quality Software Management Vols. 1 – 4 by Jerry Weinberg.  Lots of different angles on software quality from one of the patron saints of software testers.
- *Anything* by Jerry Weinberg

# *DevelopSense*

**61Ashburnham Road   Toronto, ON  M6H 2K4   (416) 656-5160    info@developsense.com**

To find good stuff **on the Web** about testing and other topics, see

- Black Box Software Testing Course (http://www.testingeducation.org/BBST/index.html) This course was co-authored by Cem Kaner and James Bach, and contains much in common with Rapid Software Testing.  The course features video lectures, course notes, recommended readings, self-study and self-testing resources.  Fabulous—and free.
- Cem Kaner  (http://www.kaner.com)  An overwhelming collection of articles, papers, and presentations on software testing, test management, elements of software law.
- James Bach  (http://www.satisfice.com)  A less overwhelming but still comprehensive collection of essays, papers, and tools from the author of the Rapid Software Testing course.
- Michael Bolton  (http://www.developsense.com)  Articles and resources on software testing topics, including test matrices, all-pairs testing, installation programs, and beta tests.  Also refer to the archived newsletters.
- The Florida Institute of Technology  (http://www.testingeducation.org)  The host for the Black Box Software Testing course above, this site also contains a large number of interesting links and articles, many written and produced by Cem Kaner and his students at Florida Tech.
- Risks Digest  (http://catless.ncl.ac.uk/risks)  A fine collection of horror stories and risk ideas that makes for excellent occasional browsing.
- StickyMinds  (http://www.StickyMinds.com)  The online presence for Better Software magazine (formerly Software Testing and Quality Engineering; STQE; "sticky"—get it?). There's a big collection of articles here of varying value.  Articles from the magazine and "StickyMinds Originals" have been edited and tend to be of higher quality than the contributed articles.
- For tutorials on various markup languages, browser scripting, server scripting, and technologies related to Web development, try www.w3schools.com.

To learn **other wonderful stuff that I believe is worth thinking about**, look at

- Please Understand Me by David Kiersey ◆ The Myers-Briggs Type Inventory, which provides insight into your own preferences and why *other people* seem to think so strangely.
- The Visual Display of Quantitative Information,  Edward Tufte ◆ How to present information in persuasive, compelling, and beautiful ways.  Other books by Tufte are terrific, too—and if you ever have an opportunity to attend his one-day course on presentation, do it!
- A Pattern Language, Christopher Alexander ◆ A book about architecture, even more interesting as a book about thinking and creating similar but unique things—like computer programs and tests for them.
- Domain Driven Design by Eric Evans, in which he introduces the concepts of "ubiquitous language"—essentially making sure that everyone in the project community is using the same terms to describe the project domain, even to the extent that that language is used in the code itself; and "knowledge crunching"—essentially why all those meetings and documents and diagrams and discussions are valuable, and how they can become more effective.
- Better Software**,** a most unfortunate name of an otherwise wonderful magazine.  Excellent information for professional testers.  Michael writes a monthly column for this magazine.

- The Amplifying Your Effectiveness Conference, held every November in Phoenix, hosted by Jerry Weinberg and his colleagues.  AZ.  See http://www.ayeconference.com for details.
- Blink, by Malcolm Gladwell.  A pop-science book about rapid cognition.  There are four central points that he tries to express in the book:  1) Snap judgments are a central part of how we make sense of the world.  2) Snap judgments are vulnerable to corruption by forces that are outside of our awareness.  3) It's possible that we may improve our snap judgments by removing information.  4) Instead of solving the problem by fixing the decision-maker, change the context in which the decision is made.  Note that the book doesn't always make it clear that these are the points; I got these from attending a lecture by Mr. Gladwell during the book tour, in which he addressed some of the criticisms of the book with these four points.  Another point that came up during the lecture:  experts simplify the field in front of them, because of their expertise behind them.  In the moment, they whittle down to the essentials.

  Mr. Gladwell's other work—his book The Tipping Point, and his New Yorker articles, archived at http://www.gladwell.com —is informed by the idea that little things make a big difference.  Not all of it can be directly related to testing, but it's all fun reading.

- About Face: The Essentials of User Interface Design  and The Inmates are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity, by Alan Cooper.  In both of these books, Mr. Cooper provides some interesting and thoughtful (and sometimes provocative) tips for people involved with the design of computer software. Most software shows us a map of its own internals, and the user interface (or, as Mr. Cooper calls it, the user interaction) must be adapted to fit that functional map. Yet customers purchase software to get work done; Mr. Cooper consistently and expertly advocates keeping the user's task in mind, and designing software that helps users instead of frustrating them. While both books are primarily oriented towards developers and software designers, managers and marketers should seriously consider reading both books, and particularly Inmates.

- Code : The Hidden Language of Computer Hardware and Software, by Charles Petzold. Code is about encoding systems--the kinds of systems by which we represent numbers and letters using computers and other kinds of machines. Hmm.... encoding systems. Sounds fascinating, huh?  As a matter of fact, this is a highly useful book. I wish it had been around when I was learning about computing machines; the book would have made a lot of things clear right away. Effective testers need to know something about boundary conditions; so do effective programmers. Numbers like 255, -32768, 65335, 4294967295, and -2147483648 are interesting; so are symbols like @, [, `, and {. Don't know why? Code will tell you.

  The book helps the reader to understand some of the otherwise obscure boundary conditions that exist because of the ways that computers work, and because of the choices that we've made in constructing those machines. You also get to understand what those dots of Braille mean, and how machines (under our instructions, of course) make decisions and evaluate information. This book probably isn't for everyone, but anyone on the engineering

side of the computer community should be familiar with the principles that Mr. Petzold explains so clearly.

- Tools of Critical Thinking, by David A. Levy, 1997.  This is a key book for Rapid Testers, in that it provides terrific, digestible descriptions of "metathoughts"—ways of thinking about thinking, and in particular, thinking errors and biases to which people are prone.  This book purports to be about clinical psychology, but we think it's about about the thinking side of testing in disguise.

- Exploring Requirements: Quality Before Design, by Don Gause and Gerald M. Weinberg

- How to Solve It, by George Polya.

- Cognition in the Wild, by Edwin Hutchins

- Thinking and Deciding, by Jonathan Baron

- Lateral Thinking: Creativity Step by Step, Ed De Bono

- The Social Life of Information, John Seely Brown, Paul Duguid

- Things That Make Us Smart: Defending Human Attributes in the Age of the Machine, by Donald Norman

- The Sciences of the Artificial, 3rd Ed., by Herbert A. Simon

- Conjectures and Refutations: The Growth of Scientific Knowledge, by Karl Popper

- Theory and Evidence: The Development of Scientific Reasoning, by Barbara Koslowski

- Abductive Inference: Computation, Philosophy, Technology, by John R. Josephson and Susan G. Josephson

- Science as a Questioning Process, by Nigel Sanitt

- Administrative Behavior, 4th ed., by Herbert Simon

- Software Testing: A Craftman's Approach, by Paul C. Jorgensen

- Bad Software: What to Do When Software Fails, by Cem Kaner and David Pels

- Introducing Semiotics, by Paul Cobley and Litza Jansz

- <u>Proofs and Refutations</u>, by Imre Lakatos

- <u>Play as Exploratory Learning</u>, by Mary Reilly

- <u>Radical Constructivism: A Way of Learning (Studies in Mathematics Education)</u>, by Ernst von E. Glasersfeld

- <u>Why Art Cannot Be Taught</u>, by James Elkins

- <u>Exploratory Research in the Social Sciences</u>, by Robert A. Stebbins

- <u>Applications of Case Study Research</u>, by Robert K. Yin

- <u>What If....Collected Thought Experiments in Philosophy</u>, by Peg Tittle

- <u>Abductive Inference : Computation, Philosophy, Technology</u>, by John R. Josephson

- <u>Time Pressure and Stress in Human Judgment and Decision Making</u>, edited by A.J. Maule and O. Svenson

- <u>Outlines of Scepticism</u>, by Sextus Empiricus

- <u>System of Logic Rationcinative and Inductive</u>, by John Stuart Mill

## *Tools*

The simplest way to find these tools, at the moment, is to Google for them.  Everything listed here is either free or a free trial; we encourage readers to register the commercial products if you find them useful.

In addition to the tools listed here, check out the tools listed in the course notes and in the article "Boosting Your Testing Superpowers" in the Appendix.  Danny Faught also provides reviews and listings of testing and configuration management tools at <u>http://www.tejasconsulting.com/open-testware/</u>.

**Netcat** (a.k.a. NC.EXE) This is a fantastic little tool that, from the command line, allows you to make TCP/IP connections and observe traffic on specific ports.  For lots of examples on how to use it, see the above-referenced Hacking Web Applications Exposed.

**SysInternals Tools** at <u>http://www.sysinternals.com</u>.  These wonderful, free tools for Windows are probes that reveal things that we would not ordinarily see.  **FileMon** watches the file system for opening, closing, reading, and writing, and identifies which process was responsible for each action.  **RegMon** does the same thing for the Windows Registry.  **Process Explorer** identifies which files

are open and which Dynamic Link Libraries (DLLs) are in use by applications and processes on the system.  **Strings** is a bog-simple little utility that dumps the textual contents of any kind of file, most useful for executables.  I've found lots of silly little spelling errors with this tool; I've also found hints about the relationships between library files.

**Perl**. Grab Perl from the ActiveState distribution, http://www.activestate.com.  They also have development tools that allow you to do things like create and distribute .EXE files from Perl scripts—which means that people can run programs written in Perl without having to install the whole gorilla.  Also see CPAN, the Comprehensive Perl Archive Network at http://www.cpan.org. This is a library of contributions to the Perl community.  Many, many problems that you'll encounter will already have a solution posted in CPAN.

**Ruby**.  Get Ruby from www.rubycentral.com and/or the sites that link from it.  After you've done that, look into the beginner's tutorial at http://pine.fm/LearnToProgram/?Chapter=00; some of Brian Marick's scripting for testers work at http://www.visibleworkings.com/little-ruby/.  Then read the Pickaxe book whose real name is Programming Ruby (look up Pickaxe on Google); you might also like to look at the *very* eccentric "Why's Poignant Guide to Ruby" at http://poignantguide.net/ruby/.

**WATIR** (Web Application Testing In Ruby) and **SYSTIR** (System Testing In Ruby) are emerging and interesting tools based on Ruby, with the goal of permitting business or domain experts to comprehend examples or tests.

**SAMIE** was the Perl-based tool that at least partially inspired Ruby-based WATIR. SAMIE, with the Slingshot utility package, allows you to identify objects on a Web page so that you can more easily build a Perl-based script to fill out forms and drive the page.

**SnagIt**, a wonderful screen capture and logging utility from TechSmith.  Available in trialware at http://www.techsmith.com

**TextPad**, a terrific text editor with excellent regular expression support and the ability to mark and copy text by columns as well as by lines.  Shareware, available at http://www.textpad.com.

**PerlClip**, a utility for blasting lots of patterned data onto the Windows Clipboard for input constraint attacks.   So-called because it's written in Perl, and it uses Perl-like syntax to create the patterns.  Counterstrings—strings that report on their own length—are perhaps the coolest of several cool features.  Written by James Bach and Danny Faught, and available free from http://www.satisfice.com and in the course materials for the Rapid Software Testing course.

**AllPairs**, to generate minimally-size tables of data that include each pair of possible combinations at least once.  Written by James Bach and available free from http://www.satisfice.com and in the course materials for the Rapid Software Testing course.