CEM KANER
JAMES BACH
BRET PETTICHORD

Lessons Learned in SOFTWARE TESTING

A Context-Driven Approach

"Pick up this book, open it anywhere, read it for two minutes, and take one lesson as a suggestion. Your testing, test planning, test management, or thinking about testing will improve dramatically."

Johanna Rothman, Rothman Consulting Group, Inc.

"If you test software, or depend on people who do, then read this book. Each page bubbles with hard-won advice for handling the practical problems you encounter every day."

Sam Guckenheimer
Senior Director of Automated Testing Technology
Rational Software Corporation

"Definitely a book worth reading and keeping around. Smart, practical, insightful and thought-provoking."

Ross Collard, Collard & Company

"These three distinguished test professionals have written a precisely-stated and thought-provoking book that offers a distinctive and important perspective on testing and test project management."

Rex Black, Author of Managing the Testing Process
and Critical Testing Processes

"This is the book the testing community has been looking for and didn't realize it. A must read for any test engineer or manager."

George Hamblen Jr., Director of Software Quality Assurance
for a large financial services company

"This isn't textbook stuff. It's better. It's real life under discussion and observation. I'm excited to see so many aspects of testing being brought together into one book. I expect great discussions to be had because of this book."

Steve Tolman, Manager of Software Quality, PowerQuest

"These lessons contain wonderful insights about software testing in the real world, from the leading practical experts on software testing. Whether you test software, or work with people who do, this book is great stuff."

Alan Myrvold

"Clear and succinct. It has brought clarity to many of my own learning experiences and provoked a lot of new thoughts."

Fran McKain
Software Test Manager
Hewlett-Packard Company

"Reviewing this book was one of my greatest learning experiences. I warmly recommend this as a must-have for any testing professional."

Hans Buwalda, Author of Integrated Test Design and Automation

"The book is packed with nuggets of gold derived from years of practical experience. The chapter on test automation alone is more useful than any of the books I've seen on test automation. The chapter on techniques has powerful ideas, simply stated!"

Doug Hoffman, Consultant, Software Quality Methods, LLC

"Lessons Learned in Software Testing is a must read for the beginner who needs tried and true tips and for the mature test manager who is looking for more refinements for his or her organization."

Chris DeNardis, Supervisor of Software Engineering, Rockwell Automation

" . . . offers an invaluable collection of real world practices based on years of experience shared by the authors collectively and many of their colleagues . . . an absolute must for anyone who has a serious interest in software testing."

Hung Q. Nguyen, President and CEO, LogiGear Corporation
Author of Testing Applications on the Web

"The lessons format is simple and succinct, just the thing for us to use in late night test planning sessions. Where other books have been long on theory and are great for study, this is long on reality, practicality, and immediate usefulness."

Mary Romero Sweeney, Author of Visual Basic for Testers

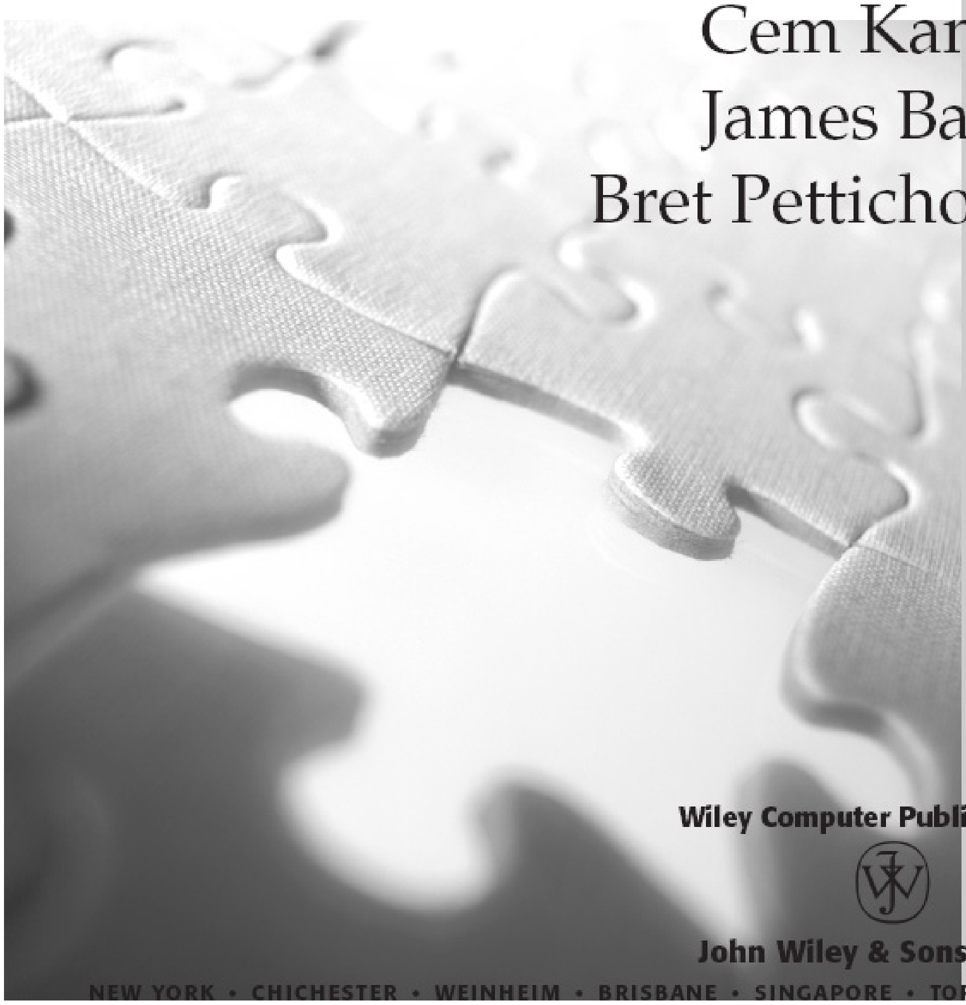
"This is an excellent book. I have had similar experiences as documented in this book, without being able to learn the lessons whilst in the middle of the problem."

Stale Amland, Amland Consulting, Norway

Lessons Learned in Software Testing

A Context-Driven Approach

Cem Kaner
James Bach
Bret Pettichord



Wiley Computer Publishing

John Wiley & Sons, Inc.

NEW YORK • CHICHESTER • WEINHEIM • BRISBANE • SINGAPORE • TORONTO

Publisher: Robert Ipsen
Editor: Margaret Eldridge
Assistant Editor: Adaobi Obi
Managing Editor: Micheline Frederick

Text Design & Composition: D&G Limited, LLC

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Copyright © 2002 by Cem Kaner, James Bach, Bret Pettichord. All rights reserved. Published by John Wiley & Sons, Inc., New York.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

This title is also available in print as ISBN 0-471-08112-4. Some content that appears in the print version of this book may not be available in this electronic edition.

For more information about Wiley products, visit our web site at www.Wiley.com

DEDICATION

To Brian Marick and Sam Guckenheimer, who set the spark for this book.

To Dave Gelperin, who believed in us and built a community.

To Jerry Weinberg, whose life and work embodies the highest ideals of an expert tester.

In memoriam, Anna Allison, colleague and friend, September 30, 1952-September 11, 2001.

CONTENTS

Lessons

Foreword

Preface

Acknowledgments

CHAPTER 1 The Role of the Tester

CHAPTER 2 Thinking Like a Tester

CHAPTER 3 Testing Techniques

CHAPTER 4 Bug Advocacy

CHAPTER 5 Automating Testing

CHAPTER 6 Documenting Testing

CHAPTER 7 Interacting with Programmers

CHAPTER 8 Managing the Testing Project

CHAPTER 8 Managing the Testing Project

CHAPTER 10 Your Career in Software Testing

CHAPTER 11 Planning the Testing Strategy

APPENDIX The Context-Driven Approach to Software Testing

Bibliography

Index

Title Page

Copyright

Dedication

Foreword

Preface

Acknowledgments

CHAPTER 1 The Role of the Tester

- lesson 1 You are the headlights of the project
- lesson 2 Your mission drives everything you do
- lesson 3 You serve many clients
- lesson 4 You discover things that will "bug" someone whose opinion matters
- lesson 5 Find important bugs fast
- lesson 6 Run with the programmers
- lesson 7 Question everything, but not necessarily out loud
- lesson 8 You focus on failure, so your clients can focus on success
- lesson 9 You will not find all the bugs
- lesson 10 Beware of testing "completely."

lesson 11 You don’t assure quality by testing
lesson 12 Never be the gatekeeper!
lesson 13 Beware of the not-my-job theory of testing
lesson 14 Beware of becoming a process improvement group
lesson 15 Don’t expect anyone to understand testing, or what you need to do it well

CHAPTER 2 Thinking Like a Tester

lesson 16 Testing is applied epistemology
lesson 17 Studying epistemology helps you test better
lesson 18 Testing is grounded in cognitive psychology
lesson 19 Testing is in your head
lesson 20 Testing requires inference, not just comparison of output to expected results
lesson 21 Good testers think technically, creatively, critically, and practically
lesson 22 Black box testing is not ignorance-based testing
lesson 23 A tester is more than a tourist
lesson 24 All tests are an attempt to answer some question
lesson 25 All testing is based on models

lesson 26 Intuition is a fine beginning, but a lousy conclusion
lesson 27 To test, you must explore
lesson 28 Exploring involves a lot of thinking
lesson 29 Use the logic of abductive inference to discover conjectures
lesson 30 Use the logic of conjecture and refutation to evaluate a product
lesson 31 A requirement is a quality or condition that matters to someone who matters
lesson 32 You discover requirements by conference, inference, and reference
lesson 33 Use implicit as well as explicit specifications
lesson 34 “It works” really means it appears to meet some requirement to some degree
lesson 35 In the end, all you have is an impression of the product
lesson 36 Don’t confuse the test with the testing
lesson 37 When testing a complex product: plunge in and quit
lesson 38 Use heuristics to quickly generate ideas for tests
lesson 39 You can’t avoid bias, but you can manage it

lesson 40 You're harder to fool if you know you're a fool
lesson 41 When you miss a bug, check whether the miss is surprising or just the natural outcome of your strategy
lesson 42 Confusion is a test tool
lesson 43 Fresh eyes find failure
lesson 44 Avoid following procedures unless they followed you first
lesson 45 When you do create test procedures, avoid "1287."
lesson 46 One important outcome of a test process is a better, smarter tester
lesson 47 You can't master testing unless you reinvent it

CHAPTER 3 Testing Techniques

lesson 48 Testing combines techniques that focus on testers, coverage, potential problems, activities, and evaluation
lesson 49 People-based techniques focus on who does the testing
lesson 50 Coverage-based techniques focus on what gets tested
lesson 51 Problems-based techniques focus on why you're testing (the risks you're testing for)
lesson 52 Activity-based techniques focus on how you test

lesson 53 Evaluation-based techniques focus on how to tell whether the test passed or failed
lesson 54 The classification of a technique depends on how you think about it

CHAPTER 4 Bug Advocacy

lesson 55 You are what you write
lesson 56 Your advocacy drives the repair of the bugs you report
lesson 57 Make your bug report an effective sales tool
lesson 58 Your bug report is your representative
lesson 59 Take the time to make your bug reports valuable
lesson 60 Any stakeholder should be able to report a bug
lesson 61 Be careful about rewording other people’s bug reports
lesson 62 Report perceived quality gaps as bugs
lesson 63 Some stakeholders cannot report bugs-you’re their proxy
lesson 64 Draw the affected stakeholder’s attention to controversial bugs
lesson 65 Never use the bug-tracking system to monitor programmers’ performance

lesson 66 Never use the bug-tracking system to monitor testers' performance
lesson 67 Report defects promptly
lesson 68 Never assume that an obvious bug has already been filed
lesson 69 Report design errors
lesson 70 Extreme-looking bugs are potential security flaws
lesson 71 Uncorner your corner cases
lesson 72 Minor bugs are worth reporting and fixing
lesson 73 Keep clear the difference between severity and priority
lesson 74 A failure is a symptom of an error, not the error itself
lesson 75 Do follow-up testing on seemingly minor coding errors
lesson 76 Always report nonreproducible errors; they may be time bombs
lesson 77 Nonreproducible bugs are reproducible
lesson 78 Be conscious of the processing cost of your bug reports
lesson 79 Give special handling to bugs related to the tools or environment
lesson 80 Ask before reporting bugs against prototypes or early private versions
lesson 81 Duplicate bug reports are a self-correcting problem

lesson 82 Every bug deserves its own report
lesson 83 The summary line is the most important line in the bug report
lesson 84 Never exaggerate your bugs
lesson 85 Report the problem clearly, but don’t try to solve it
lesson 86 Be careful of your tone. Every person you criticize will see the report
lesson 87 Make your reports readable, even to people who are exhausted and cranky
lesson 88 Improve your reporting skills
lesson 89 Use market or support data when appropriate
lesson 90 Review each other’s bug reports
lesson 91 Meet the programmers who will read your reports
lesson 92 The best approach may be to demonstrate your bugs to the programmers
lesson 93 When the programmer says it’s fixed, make sure it isn’t still broken
lesson 94 Verify bug fixes promptly
lesson 95 When fixes fail, talk with the programmer
lesson 96 Bug reports should be closed by testers
lesson 97 Don’t insist that every bug be fixed. Pick your battles
lesson 98 Don’t let deferred bugs disappear

lesson 99 Testing inertia should never be the cause of bug deferral
lesson 100 Appeal bug deferrals immediately
lesson 101 When you decide to fight, decide to win!

CHAPTER 5 Automating Testing

lesson 102 Speed the development process instead of trying to save a few dollars on testing
lesson 103 Expand your reach instead of trying to repeat the same tests over and over
lesson 104 Select your automation strategy based on your context
lesson 105 Don’t mandate 100 percent automation
lesson 106 A test tool is not a strategy
lesson 107 Don’t automate a mess
lesson 108 Don’t equate manual testing to automated testing
lesson 109 Don’t estimate the value of a test in terms of how often you run it
lesson 110 Automated regression tests find a minority of the bugs
lesson 111 Consider what bugs you aren’t finding while you automate tests
lesson 112 The problem with bad automation is that no one may notice

lesson 113 Capture replay fails
lesson 114 Test tools are buggy
lesson 115 User interfaces change
lesson 116 Select GUI test tools based on compatibility, familiarity, and service
lesson 117 Automated regression tests die
lesson 118 Test automation is a software development process
lesson 119 Test automation is a significant investment
lesson 120 Test automation projects require skills in programming, testing, and project management
lesson 121 Use pilot projects to prove feasibility
lesson 122 Have testers and programmers charter automation projects
lesson 123 Design automated tests to facilitate review
lesson 124 Don’t skimp on automated test design
lesson 125 Avoid complex logic in your test scripts
lesson 126 Don’t build test libraries simply to avoid repeating code
lesson 127 Data-driven test automation makes it easy to run lots of test variants
lesson 128 Keyword-driven test automation makes it easy for nonprogrammers to create tests

lesson 129 Use automated techniques to generate test inputs
lesson 130 Separate test generation from test execution
lesson 131 Use standard scripting languages
lesson 132 Automate tests using programming interfaces
lesson 133 Encourage the development of unit test suites
lesson 134 Beware of using automators who don’t understand testing
lesson 135 Avoid automators who don’t respect testing
lesson 136 Testability is often a better investment than automation
lesson 137 Testability is visibility and control
lesson 138 Start test automation early
lesson 139 Give centralized automation teams clear charters
lesson 140 Automate for immediate impact
lesson 141 You may have more test tools than you realize

CHAPTER 6 Documenting Testing

lesson 142 To apply a solution effectively, you need to understand the problem clearly.

lesson 143 Don’t use test documentation templates: A template won’t help unless you don’t need it.
lesson 144 Use test documentation templates: They foster consistent communication.
lesson 145 Use the IEEE Standard 829 for test documentation.
lesson 146 Don’t use the IEEE Standard 829.
lesson 147 Analyze your requirements before deciding what products to build; this applies as much to your documentation as to your software.
lesson 148 To analyze your test documentation requirements, ask questions like the ones in this list.
lesson 149 Summarize your core documentation requirements in one sentence with no more than three components.

CHAPTER 7 Interacting with Programmers

lesson 150 Understand how programmers think
lesson 151 Develop programmers’ trust
lesson 152 Provide service
lesson 153 Your integrity and competence will demand respect

lesson 154 Focus on the work, not the person
lesson 155 Programmers like to talk about their work. Ask them questions
lesson 156 Programmers like to help with testability

CHAPTER 8 Managing the Testing Project

lesson 157 Create a service culture
lesson 158 Don’t try to create a control culture
lesson 159 Develop the power of the king’s ear
lesson 160 You manage the subproject that provides testing services, not the development project
lesson 161 All projects evolve. Well-run projects evolve well
lesson 162 There are always late changes
lesson 163 Projects involve a tradeoff among features, reliability, time, and money
lesson 164 Let the project manager choose the project lifecycle
lesson 165 Waterfall lifecycles pit reliability against time
lesson 166 Evolutionary lifecycles pit features against time

lesson 167 Be willing to allocate resources to the project early in development
lesson 168 Contract-driven development is different from market-seeking development
lesson 169 Ask for testability features
lesson 170 Negotiate the schedules for builds
lesson 171 Understand what programmers do (and don’t do) before delivering builds
lesson 172 Be prepared for the build
lesson 173 Sometimes you should refuse to test a build
lesson 174 Use smoke tests to qualify a build
lesson 175 Sometimes, the right decision is to stop the test and fix cycle and redesign the software
lesson 176 Adapt your processes to the development practices that are actually in use
lesson 177 “Project documents are interesting fictions: Useful, but never sufficient.”
lesson 178 Don’t ask for items unless you will use them
lesson 179 Take advantage of other sources of information
lesson 180 Flag configuration management problems to the project manager
lesson 181 Programmers are like tornadoes

lesson 182 Great test planning makes late changes easy
lesson 183 Test opportunities arise whenever one person hands off an artifact to another
lesson 184 There is no universal formula for knowing how much testing is enough
lesson 185 “Enough testing” means “enough information for my clients to make good decisions.”
lesson 186 Never budget for just two testing cycles
lesson 187 To create a schedule for a set of tasks, estimate the amount of time needed for each task
lesson 188 The person who will do the work should tell you how long a task will take
lesson 189 There is no right ratio of testers to other developers
lesson 190 Trade tasks or transfer people from tasks that they are failing at
lesson 191 Rotate testers across features
lesson 192 Try testing in pairs
lesson 193 Assign a bug hunter to the project
lesson 194 Charter testing sessions, especially exploratory testing sessions
lesson 195 Test in sessions
lesson 196 Use activity logs to reveal the interruptions that plague testers’ work

lesson 197 Regular status reports are powerful tools
lesson 198 There’s nothing more dangerous than a vice president with statistics
lesson 199 Be cautious about measuring the project’s progress in terms of bug counts
lesson 200 The more independent coverage measures you use, the more you know
lesson 201 Use a balanced scorecard to report status on multiple dimensions
lesson 202 Here’s a suggested structure for a weekly status report
lesson 203 A project dashboard is another useful way for showing status
lesson 204 Milestone reports are useful when milestones are well defined
lesson 205 Don’t sign-off to approve the release of a product
lesson 206 Do sign-off that you have tested a product to your satisfaction
lesson 207 If you write a release report, describe your testing work and results, not your opinion of the product
lesson 208 List unfixed bugs in the final release report
lesson 208 A useful release report lists the 10 worst things critics might say

CHAPTER 8 Managing the Testing Project

lesson 210 Mediocrity is a self-fulfilling prophecy
lesson 211 Treat your staff as executives
lesson 212 Read your staffs bug reports
lesson 213 Evaluate your staff as executives
lesson 214 If you really want to know what’s going on, test with your staff
lesson 215 Don’t expect people to handle multiple projects efficiently
lesson 216 Build your testing staffs domain expertise
lesson 217 Build your testing staffs expertise in the relevant technology
lesson 218 Work actively on skills improvement
lesson 219 Review technical support logs
lesson 220 Help new testers succeed
lesson 221 Have new testers check the documentation against the software
lesson 222 Familiarize new testers with the product through positive testing
lesson 223 Have novice testers edit old bug reports before writing new ones
lesson 224 Have new testers retest old bugs before testing for new bugs
lesson 225 Don’t put novice testers on nearly finished projects

lesson 226 The morale of your staff is an important asset
lesson 227 Don’t let yourself be abused
lesson 228 Don’t abuse your staff with overtime
lesson 229 Don’t let your staff be abused
lesson 230 Create training opportunities
lesson 231 Your hiring decisions are your most important decisions
lesson 232 Hire contractors to give you breathing room during recruiting
lesson 233 Rarely accept rejects from other groups into testing
lesson 234 Plan in terms of the tasks you need to do in your group and the skills needed to do them
lesson 235 Staff the testing team with diverse backgrounds
lesson 236 Hire opportunity candidates
lesson 237 Hire by consensus
lesson 238 Hire people who love their work
lesson 239 Hire integrity
lesson 240 During the interview, have the tester demonstrate the skills you’re hiring him for
lesson 241 During the interview, have the tester demonstrate skills he’ll actually use on the job over informal aptitude tests
lesson 242 When recruiting, ask for work samples

lesson 243 Hire quickly after you make up your mind
lesson 243 Put your hiring promises in writing and keep them

CHAPTER 10 Your Career in Software Testing

lesson 245 Choose a career track and pursue it
lesson 246 Testers' incomes can be higher than programmers' incomes
lesson 247 Feel free to change your track and pursue something else
lesson 248 Whatever path you take, pursue it actively
lesson 249 Extend your career beyond software testing
lesson 250 Extend your career beyond your company
lesson 251 Conferences are for conferring
lesson 252 Lots of other companies are as screwed up as yours
lesson 253 If you don't like your company, look for a different job
lesson 254 Be prepared in case you have to bet your job (and lose)
lesson 255 Build and maintain a list of companies where you'd like to work
lesson 256 Build a portfolio
lesson 257 Use your resume as a sales tool

lesson 258 Get an inside referral
lesson 259 Research salary data
lesson 260 If you’re answering an advertisement, tailor your answer to the advertisement
lesson 261 Take advantage of opportunities to interview
lesson 262 Learn about companies when you apply for jobs with them
lesson 263 Ask questions during job interviews
lesson 264 Negotiate your position
lesson 265 Be cautious about Human Resources
lesson 266 Learn Perl
lesson 267 Learn Java or C++
lesson 268 Download demo copies of testing tools and try them out
lesson 269 Improve your writing skills
lesson 270 Improve your public speaking skills
lesson 271 Think about getting certified
lesson 272 If you can get a black belt in only two weeks, avoid fights
lesson 273 A warning about the efforts to license software engineers

CHAPTER 11 Planning the Testing Strategy

lesson 274 Three basic questions to ask about test strategy are “why bother?”, “who cares?' and “how much?”
lesson 275 There are many possible test strategies
lesson 276 The real test plan is the set of ideas that guides your test process
lesson 277 Design your test plan to fit your context
lesson 278 Use the test plan to express choices about strategy, logistics, and work products
lesson 279 Don’t let logistics and work products blind you to strategy
lesson 280 How to lie with test cases
lesson 281 Your test strategy is more than your tests
lesson 282 Your test strategy explains your testing
lesson 283 Apply diverse half-measures
lesson 284 Cultivate the raw materials of powerful test strategies
lesson 285 Your first strategy on a project is always wrong
lesson 286 At every phase of the project, ask yourself “what can I test now and how can I test it?”
lesson 287 Test to the maturity of the product