

Resource Reviews

SOFTWARE QUALITY MANAGEMENT

Fit for Developing Software

Rick Mugridge and Ward Cunningham.

2005. Prentice Hall

Professional Technical Reference

(<http://www.phptr.com>).

361 pages.

ISBN: 0-321-26934-9

CSQE Body of Knowledge areas: Software Quality Management; Software Verification and Validation

*Reviewed by Scott Duncan
sduncan@computer.org*

As its subtitle says, this is a book about a "Framework for Integrated Tests." That is, it is about how to "clarify business rules, express them with concrete examples, and organize the examples into test tables that drive testing throughout the software life cycle." So the book is more than just about testing software. It is also about using testing as an approach to making requirements more concrete and clear as well as making testing a life-cycle-wide practice.

So what is "Fit"? It "is a tool for people to sit down and collaboratively create tests" and then, using language-specific drivers (called "fixtures"), run those tests and get feedback on their success or failure. Fit is used most widely in agile projects with client-server and Web-based architectures, using C, C++, Java, and so on. However, there is nothing fundamentally preventing its use in more legacy,

mainframe environments, using COBOL. It's just that those who have been supporting test automation through fixtures have been working in the former areas rather than legacy systems.

Fit accomplishes its goals of communicating what an application should do and then demonstrating that though successfully passed tests. The simplest form of a Fit table is a column devoted to one (input) data value and a second column devoted to an expected output value based on execution of the software for some calculation/function. Far more complex tabular arrangements, however, are possible as the book shows when it illustrates how an entire invoice can be defined in a Fit table.

Fixtures, written for different environments and languages, allow submitting the input values to the software being tested, capturing the outputs, and passing them back for Fit to compare to the expected value. Color-coding, fonts, and other HTML-controlled formatting are used to indicate whether a test passes or fails based on whether the expected value is returned or not as well as to show what value was returned if it was not the expected one.

Related to Fit is FitNesse "a convenient approach to creating, organizing, and running Fit tests," which "allow you to view, change, and create Web pages containing documentation and test tables that are to be run with Fit . . . through a Web browser and to share tables among people working on a project."

A limitation to Fit, for some, will be that it is Java-based in terms of its own execution environment. Though one does not

have to know Java to write and execute tests (unlike, for example, Junit, another popular client-server/Web testing tool), one does have to know about Java execution to install Fit. For more complex Fit table structures, knowing Java is a big help.

Another potential limitation is that Fit requires a fixture that can communicate with the system under test. Many fixtures have been written for languages and environments associated with client-server and Web-based development. This book offers guidance in how to write a fixture such that, if one has not already been developed, an organization (or individual) could create one for their specific needs.

This book, while focused on explaining the use of Fit, FitNesse, fixtures, and so on, also illustrates how one can use testing early to focus on what the software needs to do, helping with the clarification and specification of requirements in a very "testable" manner. The good news is that Fit, FitNesse, a variety of fixtures, and supporting tools are freely available on the Web at places like <http://fit.c2.com>, <http://sourceforge.net/projects/fitlibrary>, and <http://www.fitnessse.org>.

If the reader is looking for a new way to think about testing and its role in requirements clarification and validation, then Fit may be something to look into.

Scott Duncan has 30 years of experience in all facets of internal and external product software development with commercial and government organizations. For the last nine years he has been an internal/external consultant helping software organizations achieve international standard registration.

SOFTWARE ENGINEERING PROCESS

Strategic Software Engineering

Faid P. Deek, James A.M.
McHugh, and Osama M.
Eljabiri. 2005.

Auerbach Publications
(<http://www.crcpress.com>).
347 pages.
ISBN: 0-8493-3939-1

CSQE Body of Knowledge:
Software Engineering
Processes

Reviewed by Scott Duncan
sduncan@computer.org

The authors describe this book as “a view of software engineering as a strategic, business-oriented, interdisciplinary enterprise, rather than as a primarily technical and scientifically focused process.” It does not really say how to “do” software engineering as much as how processes, models, and standards fit within the business (managerial) context of developing software. Therefore, it is not a book to take into the CSQE exam.

This book does have decent summaries of various development life-cycle and process models, and it summarizes traditional factors related to such models. At a high level it mentions process improvement, assessing life-cycle models, problem-solving processes and technology, requirements and problem engineering, and interdisciplinary factors (that is, people, economics, and specific system types such as real-time, Web-based, and so on). But despite the 2005 copyright, the book refers to the Software Engineering Institute’s (SEI) older Capability Maturity Model (CMM), not the newer integrated version, CMMI, although it addresses agile methods and aspect-oriented devel-

opment, which are not significantly older than the CMMI.

Overall, this is a book for aspiring information technology managers rather than day-to-day quality engineering and development practitioners.

Agile Java Crafting Code with Test-Driven Development

Jeff Langr.
2005. Prentice-Hall/PTR
(<http://www.phptr.com>).
752 pages.
ISBN: 0-13-148239-4

CSQE Body of Knowledge
areas: Software Engineering
Processes-Analysis, design, and
development methods and tools

Reviewed by Ray Schneider
(rschneid@bridgewater.edu)

Readers may be thinking, “No—not another Java book!” At least that’s what they would be thinking if their bookshelves had as many Java books as mine does. Two things captured my interest: the words *agile* and *test-driven development*. “Agile” is the latest buzz in software development. Is it going to deliver on its promises or just be another silver bullet among the many that have promised breakthroughs and a new order of things in years past? Time will tell.

Test-driven development (TDD) is the fancy way of referring to extreme programming’s mantra of “test first.” In the introduction Langr proposes to teach Java in a new way, promising to replace the current code-run-and-observe approach with TDD, giving the programmer a high volume of low-level feedback in a step-by-step incremental test assertion driven methodology. The TDD paradigm goes something like: 1) write a test that asserts a particular result, 2)

execute the test showing that it fails, 3) write code that accomplishes what is asserted, 4) rerun the test to show it passes now, 5) refactor as required. One continues this fine-grained process until the work is finished. This might be summarized as test-fail-code-test-pass-refactor.

Readers might ask: “Why is this a good idea?” The book is an experiential answer. The advantages include a very short incremental development loop and incremental cumulative test suites created as one codes. This gives users confidence in the code. As Langr points out: “You learn how to incrementally develop code, getting feedback every few seconds or minutes that you are progressing in a valid direction. Tests are about confidence.”

At first I found the lack of any actual design context disconcerting, which is probably due to my process-oriented past. Here process is replaced with the creation of objects and coordinated suites of object tests. Langr cleared up my design difficulty with a brief discussion of the agile philosophy of “simple design,” which arises from conforming to four rules: 1) make sure tests are always complete and run 100 percent green (that is, pass), 2) eliminate duplication, 3) ensure code is clean and expressive, and 4) minimize the number of classes and methods.

To give the book a fair shake I worked methodically through the first few chapters devoted to beginning Java, TTD, and object-oriented programming. Agile Java is organized into 15 lessons, about 30 pages each. Three additional lessons close out the book: two on Swing-based user interface development, and a miscellaneous chapter that is a grab bag of topics including JARs, the finalize method, regular expressions Java Database Connectivity (JDBC), and others.

Agile Java doesn’t waste time on traditional methods of teaching

Reviews

languages. Instead it dives right into a simple test using the JUnit framework. The first test is simply the creation of an object of a class. Since the class does not exist the test fails. But readers learn how to set up a simple test. The next step is to write just enough code to make the test pass. Langr does an artful job of integrating not only the test development and the code development, but also the infrastructure. Brief discussions guide readers through the setup of JUnit and creating shell scripts (Unix and Windows) to compile the Java code and run the tests successfully.

Progress through the book is guided by the theme of incrementally developing a student information system using TDD. The code can be manually entered following the systematic discussion in the book. I strongly recommend that readers do that. Readers can also download the code as it stands at the end of any lesson from the book's Web site: <http://www.LangrSoft.com/agileJava/code>.

Agile Java is one of those books that come along only once in a long while. It presents a familiar topic in a new, challenging, and illuminating way. It is a terrific book for those who want to become intimately familiar with this important agile method. It's also a great book for programmers to learn Java by immersing themselves in the text and working steadily through it. For readers who are true beginners, they are unlikely to be able to use this book without a mentor to lean on and provide some helpful handholding. There is just too much material that requires at least a general knowledge of programming practices. For those who are up to the work, this is a book that will reward their effort with a lot of insight. Frankly, I loved it.

Ray Schneider holds a bachelor's degree in physics, a master's degree in engineering science, and a doctorate in information technology. He is a licensed professional engineer in the state of Virginia. He has more than 35 years of product development and applied research and development experience working for government, defense industry, and small business. He is currently an assistant professor in the Mathematics and Computer Science Department of Bridgewater College in Bridgewater, Va. He can be reached at rschneid@bridgewater.edu.

Competitive Engineering

Tom Gilb. 2005. Elsevier (<http://www.elsevier.com>). 468 pages.

CSQE Body of Knowledge area: Software Engineering Processes

Reviewed by Scott Duncan
sduncan@computer.org

The subtitle of this book is *A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*. Planguage is "a new industrial systems engineering language for communicating systems engineering and management specifications, and a set of methods providing advice on best practices. 'Planguage' is central to Competitive Engineering (CE) and permeates all themes of this book." So to buy in to what Gilb has to say, one would seem to need to accept Planguage as a way to say it. While that may be true at the most practical level, that is, of actually doing what Gilb describes and recommends, there is a lot of value in this book apart from having to adopt Planguage.

The four main themes of the book, which Gilb calls Planguage Methods, are:

- Requirements specification with an emphasis "on specifying competitive performance and resource attributes quantitatively."
- Impact estimation "to evaluate designs against the requirements" to "understand strategies" and "track progress toward meeting requirements."
- Specification quality control "to check the adherence of any plan, contract, bid or technical specification to best practice specification standards" or "how to know how well you specified."
- Evolutionary project management "to plan and monitor implementation of the selected designs" with early, frequent delivery of small increments useful to the stakeholder in priority order based on stakeholder benefit (in case you were wondering where agile methods may have started).

Besides chapters specifically on these themes, there are also chapters on:

- Functions (what a system does)
- Performance (how good the system does it)
- Scales of measure (how to quantify what the system does)
- Resources, budgets, and costs (what it takes to get the system to do it)
- Design ideas and design engineering (finding a way to make the system do it)

For those who are familiar with Gilb's *Principles of Software Engineering Management* (1988), he notes that this new book "is deliberately 'less chatty'" in its style than that work. Gilb says his aim is "to provide a fundamental systems engineering handbook" focused on the use of Planguage. (Indeed, in *Principles of Software Engineering Management*, Gilb notes how he was asked at one point "to show how the ideas could be used as a

planning language" for a project managers' handbook at a particular company. Planguage has been evolving ever since.)

I highly recommend this book as thought provoking and useful. A book like this isn't written without many years of experience and thought behind it.

PROJECT AND PROCESS MANAGEMENT

Two Books on Information Technology (IT) Manager Development

IT Staff Motivation and Development

Mike Sisco.

2001. MDE Enterprises (<http://www.mde.net>). 54 pages.

Building a Successful IT Organization

Mike Sisco.

2001. MDE Enterprises (<http://www.mde.net>). 84 pages.

CSQE Body of Knowledge area: Software Project Management

Reviewed by Scott Duncan
sduncan@computer.org

IT Staff Motivation and Development and *Building a Successful IT Organization* are two "monographs" by Mike Sisco in a series of 10 constituting what Sisco calls the IT Management Development Series. The former focuses on the specific topic of motivating and developing IT staff, while the latter addresses the broader subject of the entire IT organization, including motivation and development. Because they are closely related, reviewing them as a pair seems to

make sense and, indeed, there are places where referencing from one to the other is helpful.

At the outset these monographs—they're not really books and are available as PDF files—did not seem like they would be particularly enlightening, that is, they seemed to go over a lot of standard management concepts. However, they offer good, succinct coverage of material that might be presented in more long-winded fashion elsewhere. There's not a lot of theory here, just practical advice with examples drawn from Sisco's management experience at IBM and other companies. But it should be said that Sisco believes an IT organization's culture and vision are, as he says in *Building a Successful IT Organization*, "more important concept[s] to think about" before the obvious focus "on filling technical gaps within an organization." Indeed, he says, "You can always find technical skills." What you want to do is find people with "character and substance in nontechnical areas" who, preferably, "know the industry." (I don't believe Sisco is denigrating technical skills. Rather, he wants to make sure that managers understand that "success" comes from more than technical proficiency.)

Building a Successful IT Organization covers stepping into an existing organization and setting expectations for quality, service, performance, hiring, motivation and rewards, and measuring effectiveness. It offers advice on these topics as well as 13 appendices with sample organization charts and templates to carry out the advice. One main message is that "you should always strive to build an organization that can run on its own." This is repeated in various forms within the material. *Building a Successful IT Organization* steps through assessing what one has compared to what one needs, identifying potential sources for hiring

people with those skills and knowledge, appropriately recruiting and interviewing them, and, from the outset, showing them they matter by making sure what they need to do their job(s) is prepared the first day and spending time with them during their first week.

From a quality perspective, Sisco emphasizes that "every employee that talks to your client [internal or external] has professional communication and troubleshooting skills." I mention this because it seems, more and more, that people with whom customers make their first contact are often the lowest paid and least well-trained/empowered individuals in the company. Sisco does not discuss this, but I think it is telling that he devotes several pages to imbuing the organization with a firm sense of what service means.

If *Building a Successful IT Organization* has any gap, it is the matter of dealing with "staff that you don't need or staff that is not productive." Sisco notes that this is an issue, but doesn't say much about how to handle the situations. He does spend a good bit of time on setting clear goals and expectations for people through performance planning and reviews. He thinks these items help head off the more confrontational situations where people have to be let go. He says his experience shows that, after a couple sessions where people are told they are not meeting expectations, they will generally get the idea and leave or transfer on their own.

IT Staff Motivation and Development starts by noting how IT people "are a unique group of people that require more than most." Little is said, however, directly in the monograph about this point. Indeed, there are two places in *Building a Successful IT Organization* where this is addressed more specifically. Sisco does say that "some people will always be motivated by what they do and others simply are not." So the role of the IT manager, then,

is to: "facilitate motivation" by providing the "environment that reinforces their motivation and creates opportunities to succeed."

Finding out what motivates people, since everyone will likely be different, is an important aspect of what the IT manager needs to do, especially coming into a new organization. Some of Cisco's suggestions seem to depend on the larger organization's policies and procedures, but, from personal experience, managers who always suggest there is little they can do because of such policies and procedures are probably not spending the time to be as creative as Cisco suggests they should be. It's not usually about money or promotions, although, for some people, that might be it exactly. Cisco, however, suggests that a lot of careful attention and learning opportunities will motivate many people as much as the more typical corporate achievement symbols.

IT Staff Motivation and Development covers setting a vision for the organization and how individuals fit into that (as part of performance planning and review) as well as how project and non-project work fits into that vision. There's some generic planning guidance and advice at this point that isn't specifically related to motivation, though Cisco does suggest that IT people are "detail oriented" and "have an above average need for information," which effective and clear planning can satisfy. Part of the planning effort that touches on motivation has to do with creating training plans for each person, including showing them what path they might need to pursue to meet their longer-term goals. This is something IT people might miss in their shorter-term focus on near-term technical needs. The IT manager can help them appreciate the fact that multiple paths exist to go from programmer to CIO (or any point in between).

On the subject of getting the organization to "run on autopilot," Cisco has nine steps, several of which have to do with clear goals or responsibilities at the organization, team, and individual levels. Other steps address procedures, especially for escalating issues, success criteria, and having "competent people in the leadership positions of the organization." Two of the nine steps, however, address regular communication: of the "long-term vision" and "with the leaders to reinforce key success factors." Related to all of these is Cisco's emphasis on teamwork and making sure everyone understands that "Everyone wins when the team wins; no one wins unless the team wins." There are several pages devoted to this as sharing "victories and defeats."

So, as noted previously, these turn out to be effective summaries of things IT managers should know and practice to have motivated employees and a successful organization. Cisco offers a lot of lists, templates, and step-by-step advice for making this happen.

The ROI from Software Quality

Khaled El Emam.

2005. Auerbach Publications (<http://www.crcpress.com>).

296 pages.

ISBN: 0-8493-3298-2

CSQE Body of Knowledge areas: Project Management–Cost and Value Data; Project Management–Interpreting and Reporting Cost of Quality Data; Process and Product Measurement

Reviewer Pieter Botman
p.botman@ieee.org

Khaled El Emam opens this book by asserting that "there is a clear and

hard-to-dispute benefit to improved software quality." In a business environment, poor software quality costs the purchaser in many respects (he cites several pointed examples). Software producers and developers, not facing proportionate market or legal pressures, have been "getting away with" poor quality practices, poor economic and management practices, and issuing software products of unknown or uncontrolled quality.

This book is primarily aimed at managers wishing to perform ROI calculations, and to measure the economic factors associated with software quality in a meaningful way. Though the earlier sections discuss the plight of the software purchaser, the majority are written for the software producer, focusing on the management of development processes through measurement and economic analysis.

The opening chapters discuss software quality, its definition, and quality-related practices within the development life cycle. El Emam goes on to broaden the discussion to cover many ROI factors and practices beyond those related strictly to software quality assurance, such as project management, requirements, and process measurement. In subsequent chapters on cost of ownership and cost to developers, the discussion is raised to an economic level, explaining the underlying cost models associated with quality. Throughout these chapters, he cites underlying studies that have contributed hard evidence, and even brings out selected data values from these studies for use in example calculations.

The middle chapters contain explanations of economics concepts (such as "payback period" and "time value of money"), software and quality cost modeling, and the construction of the return on investment (ROI) calculation, and are the heart of this book. They are clear because they are kept separate

from lengthy discussions of technical benchmarks and parametric data. Indeed, the material in these chapters will remain valid, while some material presented in earlier chapters will be affected as benchmarking data and software practices change. Readers having any prior exposure to economics and the mathematics of finance will not find these chapters overly complex.

One chapter is devoted to "The Quality of Open Source Software (OSS)." It begins with background material on the notable forces associated with OSS adoption, the open source development process, and large-scale review test cycles. Some studies of complexity and defect levels for certain OSS products are presented, followed by a section on OSS (and non-OSS) security vulnerabilities. The author never misleads the reader, and presents evidence to both support and refute some assertions relating to certain quality aspects of OSS products. His cautious and well-reasoned conclusion is that the adoption of OSS because of vaguely defined OSS process advantages has not yet been adequately justified. This view makes a great deal of sense considering that the range of OSS products is broad (from operating systems to simple character counting utilities), and that the technical factors relating to quality vary significantly over that range. This is a fascinating topic, but the reader may be left wondering whether this large topic would be better (that is, more fully) addressed elsewhere.

The book contains several useful appendices, containing more detailed aspects of models relating defects to effort (cost), and relating the cost of additional inspections to schedule reduction. The reader is referred to an associated Web site for additional material related to software quality ROI calculation.

This book presents a readable, pragmatic yet detailed view of the application of ROI techniques and

economic cost models specifically to software development. It touches on many supporting topics related to quality theory, various software engineering practices, and benchmarking data from a wide range of sources, but cannot address them in great depth. All readers in the field of software engineering will appreciate and benefit from the disciplined economic analysis described here. Experienced managers who are already familiar with the fundamentals of software quality and measurement techniques should be able to apply these economic analyses within the context of their own process improvement programs. Readers new to software measurement will gain some insights into software quality factors, but should investigate the listed references in order to learn more about certain practices in detail (such as reliability modeling), and to understand the lack of uniformity across the many software projects contributing to the various benchmark data sets.

El Emam takes a pointed swipe at software engineering management practices within the software industry, saying that important decisions are frequently not evidence based. We have a long way to go in measuring products and processes. But by emphasizing the bottom line—the economic consequences of poor quality products and poor processes—he has given managers and senior engineers a push in the right direction.

Pieter Botman is an independent consultant, with more than 20 years of experience in software engineering and project/product management. He assists companies in the areas of software process assessment/improvement, project management, quality management, and product management. He is a Senior member of the IEEE, and holds CSQE and CSDP designations, among others.

METRICS AND MEASUREMENTS

Eight Books on Statistics

CSQE Body of Knowledge area: Metrics

Reviewed by Scott Duncan
sduncan@computer.org

This review has been done as eight separate reviews presented together since all of these books focus on statistics. From the most calculation focused coverage of the topic to the broadest, the order of the reviews will be: *Quality Engineering Statistics*; *Cliff's Quick Review: Statistics*; *Statistical Analysis with Excel™ for Dummies*; *Improving Performance Through Statistical Thinking*; *How to Lie with Statistics*; *Statistics for Dummies*, *The Complete Idiot's Guide to Statistics*; and *Introduction to the Practice of Statistics*. Which book to buy will depend on whether readers want an introduction to the topic, refresher material, or something more "philosophical."

Quality Engineering Statistics

Robert A. Dovich.

1992. ASQ Quality Press (<http://www.qualitypress.asq.org>). 111 pages. ISBN: 0-87389-141-4

If readers want to know how to "do the math," this book addresses that. It assumes readers have already collected their data and know how they will analyze the results. As it says in its brief preface, it is a book intended to provide "quality engineers and other professionals with a handy reference to many statistical formulas and techniques." Its chapters are entitled:

- Point Estimates (which includes standard deviation)
- Confidence Intervals
- Testing for Differences in Means

Reviews

- Testing for Differences in Variances
- Decision Errors and Risks in Hypothesis Testing
- Continuous Probability Distributions
- Goodness-of-Fit Tests
- Sample Size Determination of Tests of Hypotheses
- Analysis of Variance
- Tables (for example, distributions, chi-square, median rank values, constants, gamma function)

This book is not an overall introduction to statistics and data analysis; it is a handbook of statistical calculations and related tables.

Cliff's Quick Review: Statistics

David H. Voelker, Peter Z. Orton, and Scott V. Adams.
2001. Wiley Publishing Inc.
(<http://www.wiley.com>).
154 pages.
ISBN: 0-7645-6388-2

This is, as the cover states, "essentials fast from the experts at CliffsNotes™." Like the Dovich book, this one covers the calculations and offers tables useful for them, although it does not go into as much depth as Dovich's book does in some areas. It covers more than just calculations, however, giving advice on, for example, how to avoid some common mistakes made, as well as explaining some "whys" associated with the calculations. It has a chapter on basic statistical concepts (for example, distributions) and measurement, including a brief discussion of measurement scales. Another chapter covers types of graphical displays from bar charts to box plots while others cover probability, sampling, and testing hypotheses.

Overall, this is a nice summary of statistical concepts for the person wanting a bit more introduction to

the topic than just a "how to calculate" handbook, but needing less advanced analysis detail.

Statistical Analysis with Excel™ for Dummies

Joseph Schmuller.
2005. Wiley Publishing Inc.
(<http://www.wiley.com>).
392 pages.
ISBN: 0-7645-7594-5

It's hard to know exactly where to place this book in the spectrum of those being covered. It is a broad coverage of statistics and statistical analysis with a lot of detail on the hows and whys, including a chapter on traps and pitfalls. On the other hand, it is, as the title suggests, focused on using Excel™, explaining in a step-by-step fashion how to use Excel's built-in functions, including loading and using Excel's Analysis ToolPak.

This could easily be viewed as a broad introduction and coverage of statistics and statistical concepts. But some of the latter books, because they do not have all the Excel-specific content, do cover the subject in more breadth and depth and make for better generic "reference" documents. However, if readers want a book they can use to learn about basic statistics and get some experience doing statistical analysis and they have Excel available, this is a good choice.

Improving Performance Through Statistical Thinking

ASQ Statistics Division
(Britz, Emerling, Hare, Hoerl, Janis, and Shade).
2000. ASQ Quality Press
(<http://www.qualitypress.asq.org>). 173 pages.
ISBN: 0-87389-467-7

In contrast to most of the other books, this one has no "statistical tools and calculations" in it whatsoever. As the title suggests, its focus is on thinking statistically, which the authors state is a "critical precursor to proper data gathering, analysis, and interpretation." The book uses case studies to illustrate issues in applying the authors' statistical thinking model to understand variation, process improvement, and problem solving. It takes the results of using quality tools and calculations and describes an approach to applying those results.

In that regard, this book, too, is not an overall introduction to statistics and data analysis. It attempts to show how one would approach using statistical information to improve their work.

How To Lie With Statistics

Darrell Huff.
1954. W.W. Norton
(<http://www.wwnorton.com>).
140 pages.
ISBN: 0-393-31072-8

Though a bit odd, compared to the other books covered, this is a classic book on how to avoid misrepresentations in statistical analysis. Actually, the author says it is "a sort of primer in ways to use statistics to deceive" and may seem "too much like a manual for swindlers." But, as he then goes on to say, "The crooks already know these tricks; honest men must learn them in self-defense."

Perhaps a listing and brief description of each chapter will explain the book best:

- The Sample with the Built-In Bias—Unfortunately, "samples, biased or too small or both, lie behind much of what we read or think we know."

- The Well-Chosen Average – An “average” doesn’t tell you much “unless you can find out which of the common kinds of average it is”
- The Little Figures That Are Not There – “Sooner or later . . . a test group is going to show a big improvement worthy of a headline and perhaps a whole advertising campaign.”
- Much Ado about Practically Nothing – “Excluded were all figures and any hint that the difference was negligible.”
- The Gee-Whiz Graph – “Nothing has been falsified – except the impression that it gives.”
- The One-Dimensional Picture – The “danger in varying the size of objects in a chart.”
- The Semiattached Figure – “If you can’t prove what you want to prove, demonstrate something else and claim that they are the same thing.”
- Post Hoc Rides Again – The fallacy that “if B follows A, then A has caused B.”
- How to Statisticulate – “Misinforming people by the use of statistical material” or to manipulate statistics.
- How to Talk Back to a Statistic – “Explaining how to look at a phony statistic in the eye and stare it down; and not less important, how to recognize sound and usable data.”

Though not a book to take into a certification exam, and one that shows its age in the examples it chooses, this is an entertaining and informative book about using, not abusing, statistics.

Statistics for Dummies

Deborah Rumsey.
2003. Wiley Publishing, Inc.
(<http://www.wiley.com>).
355 pages.
ISBN: 0-7645-5423-9

This book and the *The Complete Idiot’s Guide to Statistics* book cover much of the same material and introduce the subject of statistics well. This book, however, focuses less on calculations and explains a bit more about the “why.” It also has an appendix devoted to listing chapter-by-chapter sources for some of its material.

The Complete Idiot’s Guide to Statistics

A. Donnelly, Jr.
2004. Alpha (member of Penguin Group (USA), Inc.).
(<http://www.us.penguin.com>). 383 pages.
ISBN: 1-59257-199-9

As with *Statistics for Dummies*, this book broadly covers statistics and introduces the topic effectively.

Interestingly, this book’s first part mentions the use of Excel, while *Statistics for Dummies* does not. (Of course, the *Dummies* series does have a whole book on the subject of using Excel.) This book also has a nice glossary and exercises (with solutions) for many of its chapters, which *Statistics for Dummies* does not, emphasizing its slightly more “how,” than “why” approach.

Introduction to the Practice of Statistics

David S. Moore and
George P. McCabe.
2002. W. H. Freeman & Co.
(<http://www.whfreeman.com>).
848 pages.
ISBN: 0-7167-9657-0

In the preface to the first edition, the authors note that, had the publisher allowed it, they would have called the book “What You Should Know Before You Talk to a Statistician.” They also state that the book is “an elementary but serious introduction to modern statistics for general college audiences.” As such, the book has received a broad spectrum of reviews based on whether it was being viewed as a textbook for statisticians. The authors don’t claim it to be the former.

This book examines data and its collection (distributions, change and growth, relationships), probability and inference (for distributions, count data, and regression), and analysis of variance. In the process, it covers key data collection, display and analysis techniques, and calculations. Given that it is a textbook, however, there is some redundancy for deliberate reinforcement purposes, including many exercises for each chapter.

Among all the books reviewed, if one is looking to learn the use of statistics for application to actual R&D or production data analysis, this one probably serves the purpose best, though it is the largest and most expensive given that it is a college-level textbook.

CMM® and CMMI® are registered trademarks of the Software Engineering Institute, Carnegie Mellon University.