Editor: Warren Keuffel ■ wkeuffel@computer.org

Practical Testing Advice

Diomidis Spinellis

Testing Computer Software, Second Edition, *by Cem Kaner, Jack Falk, and Hung Quoc Nguyen, John Wiley & Sons, New York, 1999, ISBN 0-471-35846-0, 480 pp., US$34.99.*

Testing is sometimes regarded as the ugly duckling of software development. Programmers want to write perfect code, development managers prefer to schedule releases without the additional uncertainties testing results bring, and financial officers surely want to do away with additional testing costs. Fortunately, three authors who love this ugly duckling wrote *Testing Computer Software*, and they successfully transfer their knowledge and—more importantly—their views to the reader.

The authors primarily target testers and test managers involved in producing non-safety-critical software. They readily admit that computer science graduates rarely have useful testing knowledge—software testing can be better performed by suitably trained college graduates. The authors thus painstakingly introduce all relevant software engineering and programming concepts needed to perform testing, and they also point to relevant material for further reading. The software life cycle, corresponding testing cycles, glass-box versus black-box testing, regression testing, maintenance, and boundary conditions are some of the topics covered.

Bug tracking

An important aspect of testing is the clear and effective communication between testers, developers, and managers—a key factor being comprehensive and well-organized problem reports. Problem reports should clearly identify the program and the problem, the problem type, its severity, and steps to reproduce it. In addition, the development team should further expand such reports by grouping them into functional categories, assigning problems to specific developers, and tracking their evolution and final resolution.

Having designed and implemented a bug-tracking system for managing a medium-scale development project, I was impressed by the pragmatic attitude the authors advanced; all suggestions in this area are easy to put into practice and are clearly the result of extensive real-life experience. The implementation of a

0740-7459/01/$10.00 © 2001 IEEE

bug-tracking system, which is outlined in a separate chapter, is likely to bring forward political issues. Although many useful metrics can be extracted from such a system, the authors wisely point out that it would be counterproductive to use it for tracking individual performance of programmers or testers. Reports can still help make estimates based on past performance figures and judging project progress.

Another particularly thorny issue concerns the problem report life cycle. A state diagram would help readers better understand the temporal and organizational relationship between problem resolution and status. However, even the authors appear split on the issue of how to treat deferred bugs.

Test cases

Designing effective test cases is a challenge for black-box testers. Although experience is clearly an important asset, the numerous methods and heuristics described provide a starting point for a junior tester. Equivalence classes, boundary values, and regression testing can minimize the number and increase the effectiveness of test cases. Understanding state transitions, race conditions, time dependencies, and the judicious use of random input will help uncover more subtle problems. User manuals and help systems are also part of the tester's work domain; the authors devote a separate chapter to this area.

Although the book's second edition was published in 1999, the material concerning the testing of printers, testing tools, localization, and legal aspects needs to be brought up-to-date. Most printers nowadays are not directly driven by end-user software and are seldom operated using simple escape sequences. Testing tools suitable for character input/output are not applicable to graphical user interfaces, character code pages are disappearing in favor of Unicode, and language libraries and operating systems provide internationalization support with radically different testing demands from earlier ad hoc localization approaches. References also cover only material published up to 1991.

Test management

Test management, planning, and documentation are important aspects of any large testing effort. The relevant IEEE standards offer important guidance; the authors describe in simple terms the contents and usefulness of test plan elements and related documents. A concrete example for each document would have been useful but would have significantly increased the book's 480 pages. Test plans inevitably must organize the numerous elements to be tested into coherent and understandable groups. We can use lists, tables, outlines, and matrices to organize this information; the book's examples for each type of chart help the reader identify which is most appropriate.

The authors devote the last part of the book to testing management, explaining why we should integrate it into the software development life cycle. Important milestones include alpha and beta testing and the user interface freeze. Testing can be performed in-house by a dedicated testing group, or it can be outsourced to an external agency; readers who have read the book's previous chapters will probably favor the former approach. The authors' empathy with software testers and managers is commendable. Recognizing that testing is often an entry-level job that can lead to a software development position can help testers work toward professional advancement and help managers constructively deal with the inevitably high staff turnover rate.

Ready for the third edition

I enjoyed reading *Testing Computer Software*. The text contains numerous highlights offering practical advice, authoritative figures you can cite to customers and higher management, and entertaining anecdotes to share with coworkers. Although some sections need updating, I still think it is a valuable training and reference source for software testers, managers, and developers.

Diomidis Spinellis is an assistant professor in the Department of Technology and Management at the Athens University of Economics and Business. Contact him at dds@aueb.gr.

The Semi-Fundamental Corba Book

Hernan A. Wilkinson

CORBA 3 Fundamentals and Programming, Second Edition, *by Jon Siegel, John Wiley & Sons, New York, 2000, ISBN 0471295183, 900 pp., $54.99.*

CORBA 3 Fundamentals and Programming offers a compelling description of the new Object Management Group standards, called Corba 3. It explains the OMG's point of view about objects and how to distribute them, concentrating on the Common Object Request Broker Architecture (Corba) and the Object Management Architecture. It also refers to other specifications—such as the Unified Modeling Language, Meta-Object facility, and XML Metadata Interchange Format—that concentrate more on analysis, design, and information-passing issues.

Jon Siegel organized his book into three conceptual sections. Section 1 introduces Corba and is useful for people who don't know the basics of the standard and for architects who need to compare it to similar solutions (such as COM+). Section 2 concentrates more on technical details, such as the Object Request Broker (ORB) implementation, CorbaServices, CorbaFacilities (business-oriented services), and CCM (Corba Component Model). Section 3 provides a programmatic example that implements a simple point-of-sale system using three different languages (C++, Java, and Cobol) on many Corba implementations (such as IBM, Orbix, Inprise, and so forth).

Siegel covers all the OMG's specifications, but he doesn't cover them well enough to call this book a user's guide. He explains the objectives of the OMG's standards and issues that they solve instead of giving a deep description of Corba's API and tips

on how to program a Corba system.

Siegel directs the content toward different kinds of readers, from managers and architects to enterprise programmers. He also uses an informal writing style to help readers understand difficult issues and keep their attention. Unfortunately, his attempt to make the book suitable for all readers ends up being the book's biggest drawback. From a programmer's point of view, the content isn't deep enough, the example is too simple (it does not use transactions, which is a must for enterprise applications), and Siegel concentrates more on showing how to accomplish interoperability (using different languages and Corba implementations) than on real Corba tips, idioms, and patterns. He does not provide a complete description of the frameworks Corba implements or the object's IDLs that compose those frameworks.

If you are a programmer, you will find that the most interesting chapters are the ones that discuss the ORB. Those chapters explain some inside details that are important to consider if you want to implement a Corba system.

On the other hand, if you need to know what Corba is and what it covers, you'll want this book. It also shows how the OMG is focusing on giving good and complete solutions for enterprise systems problems. A good example of such vision is the new type of object called *valuetype*, which lets the systems pass objects by value, not just by reference (a solution needed after the Java ability of sending objects by the wire). Helping this vision are the new services' specifications like the CCM, Quality Of Service, and the new Persistence State Service that replaces the previous persistence specification because of its lack of acceptance and use.

Another example of how the OMG is working to minimize the implementation problems of enterprise systems is the CorbaFacilities. These specifications are oriented to solve domain-specific problems. Right now, the OMG's Domain Technology Committee includes task forces to manage issues for financing, electronic commerce, manufacturing, telecommunications, healthcare, and more. As Siegel expresses in the book, this is the fastest-growing OMG committee, and it will help the interoperability (at system level) of different domain and nondomain systems.

In summary, the book explains what all the specifications do but in a more readable way, suitable to people who are less technical. It lacks the details needed to implement a Corba system, but it is good for people who need to know about the big picture of Corba.

Hernan Wilkinson is an architect at the Banco Galicia, Argentina. Contact him at hernan.wilkinson@bancogalicia.com.ar.