THE ART OF SOFTWARE TESTING

Third Edition

GLENFORD J. MYERS
TOM BADGETT
COREY SANDLER

John Wiley & Sons, Inc.

Contents

Preface

In 1979, Glenford Myers published a book that turned out to be a classic. *The Art of Software Testing* has stood the test of time—25 years on the publisher's list of available books. This fact alone is a testament to the solid, essential, and valuable nature of his work.

During that same time, the authors of this edition (the third) of *The Art of Software Testing* published, collectively, more than 200 books, most of them on computer software topics. Some of these titles sold very well and, like this one, have gone through multiple versions. Corey Sandler's *Fix Your Own PC*, for example, is in its eighth edition as this book goes to press; and Tom Badgett's books on Microsoft PowerPoint and other Office titles have gone through four or more editions. However, unlike Myers's book, none of these remained current for more than a few years.

What is the difference? The newer books covered more transient topics—operating systems, applications software, security, communications technology, and hardware configurations. Rapid changes in computer hardware and software technology during the 1980s and 1990s necessitated frequent changes and updates to these topics.

Also during that period hundreds of books about software testing were published. They, too, took a more transient approach to the topic. *The Art of Software Testing* alone gave the industry a long-lasting, foundational guide to one of the most important computer topics: How do you ensure that all of the software you produce does what it was designed to do, and—just as important—doesn't do what it isn't supposed to do?

The edition you are reading today retains the foundational philosophy laid by Myers more than three decades ago. But we have updated the examples to include more current programming languages, and we have addressed topics that were not yet topics when Myers wrote the first edition: Web programming, e-commerce, Extreme (Agile) programming and testing, and testing applications for mobile devices.

Along the way, we never lost sight of the fact that a new classic must stay true to its roots, so our version also offers you a software testing philosophy, and a process that works across current and unforeseeable future hardware and software platforms. We hope that the third edition of *The Art of Software Testing*, too, will span a generation of software designers and developers.

Introduction

At the time this book was first published, in 1979, it was a well-known rule of thumb that in a typical programming project approximately 50 percent of the elapsed time and more than 50 percent of the total cost were expended in testing the program or system being developed.

Today, a third of a century and two book updates later, the same holds true. There are new development systems, languages with built-in tools, and programmers who are used to developing more on the fly. But testing continues to play an important part in any software development project.

Given these facts, you might expect that by this time program testing would have been refined into an exact science. This is far from the case. In fact, less seems to be known about software testing than about any other aspect of software development. Furthermore, testing has been an out-of-vogue subject; it was so when this book was first published and, unfortunately, this has not changed. Today there *are* more books and articles about software testing—meaning that, at least, the topic has greater visibility than it did when this book was first published—but testing remains among the "dark arts" of software development.

This would be more than enough reason to update this book on the art of software testing, but we have additional motivations. At various times, we have heard professors and teaching assistants say, "Our students graduate and move into industry without any substantial knowledge of how to go about testing a program. Moreover, we rarely have any advice to offer in our introductory courses on how a student should go about testing and debugging his or her exercises."

Thus, the purpose of this updated edition of *The Art of Software Testing* is the same as it was in 1979 and in 2004: to fill these knowledge gaps for the professional programmer and the student of computer science. As the title implies, the book is a practical, rather than theoretical, discussion of the subject, complete with updated language and process discussions.

Although it is possible to discuss program testing in a theoretical vein, this book is intended to be a practical, "both feet on the ground" handbook. Hence, many subjects related to program testing, such as the idea of mathematically proving the correctness of a program, were purposefully excluded.

Chapter 1 "assigns" a short self-assessment test that every reader should take before reading further. It turns out that the most important practical information you must understand about program testing is a set of philosophical and economic issues; these are discussed in Chapter 2. Chapter 3 introduces the important concept of noncomputer-based code walkthroughs, or inspections. Rather than focus attention on the procedural or managerial aspects of this concept, as most such discussions do, this chapter addresses it from a technical, how-to-find-errors point of view.

The alert reader will realize that the most important component in a program tester's bag of tricks is the knowledge of how to write effective test cases; this is the subject of Chapter 3. Chapter 4 discusses the testing of individual modules or subroutines, followed in Chapter 5 by the testing of larger entities. Chapter 6 takes on the concept of user or usability testing, a component of software testing that always has been important, but is even more relevant today due to the advent of more complex software targeted at an ever broadening audience. Chapter 7 offers some practical advice on program debugging, while Chapter 8 delves into the concepts of extreme programming testing with emphasis on what has come to be called the "agile environment." Chapter 9 shows how to use other features of program testing, which are detailed elsewhere in this book, with Web programming, including e-commerce systems, and the all new, highly interactive social networking sites. Chapter 10 describes how to test software developed for the mobile environment.

We direct this book at three major audiences. First, the professional programmer. Although we hope that not everything in this book will be new information to this audience, we believe it will add to the professional's knowledge of testing techniques. If the material allows this group to detect just one more bug in one program, the price of the book will have been recovered many times over.

The second audience is the project manager, who will benefit from the book's practical information on the management of the testing process. The third audience is the programming and computer science student, and our goal for them is twofold: to expose them to the problems of

program testing, and provide a set of effective techniques. For this third group, we suggest the book be used as a supplement in programming courses such that students are exposed to the subject of software testing early in their education.