软件工程实践丛书
软件学院教材

PEARSON
Addison Wesley

Automated Software Testing

自动化软件测试

——入门、管理与实现

Elfriede Dustin
[美] Jeff Rashka 著
John Paul

Introduction, Management, and Performance

配 套 光 盘
包含ATLI小图解

Pearson Education

清华大学出版社

软件工程实践丛书

自动化软件测试

——入门、管理与实现

Elfriede Dustin
[美] Jeff Rashka 著
John Paul

清华大学出版社

·北京·

内 容 简 介

本书是一本由由浅入深地学习自动化测试涉及的高效工具、技术和方法的技术图书。通过对成功的实现的案例学习和研究，本书提供了在软件开发过程中进行成功的自动化测试所需的所有要素。

本书可作为软件学院及大学计算机等专业相关课程的教材，也可以作为软件公司各级管理和开发人员参考。

EISBN: 0-201-43287-0

北京市版权局著作权合同登记号：图字 01-2003-0543 号

书　　名：自动化软件测试—入门、管理与实现
出 版 者：清华大学出版社（北京清华大学学研大厦，邮编 100084）
http://www.tup.com.cn
http://www.tup.tsinghua.edu.cn
责任编辑：尤晓东
印 刷 者：北京市人民文学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：880×1230 1/32　印张：19.125
版　　次：2003 年 3 月第 1 版　2003 年 3 月第 1 次印刷
书　　号：ISBN 7-89494-044-5
印　　数：0001～3000
定　　价：39.00 元

Automated Software Testing

Automated Software Testing

Introduction, Management, and Performance

Elfriede Dustin

Jeff Rashka

John Paul

ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Pearson Education Corporate Sales Division
One Lake Street
Upper Saddle River, NJ 07458
(800) 382-3419

Visit AW on the Web: www.awl.com/cseng/

Library of Congress Cataloging-in-Publication Data

Dustin, Elfriede.
Automated software testing : introduction, management, and performance / Elfriede Dustin, Jeff Rashka, John Paul.
p. cm.
Includes bibliographical references.
ISBN 0-201-43287-0
1. Computer software—Testing—Automation. I. Rashka, Jeff. II. Paul, John, 1968– . III. Title.
QA76.76.T48D87 1999
005.1'4—dc21 99-25423
CIP

ISBN 0-201-43287-0
Text printed on recycled paper
3 4 5 6 7 8 9 10—MA—03020100
Third printing, June 2000

Preface

Automated Software Testing addresses the challenge for today's software professionals who are faced with real schedule deadlines and need to introduce, manage, and perform automated testing on a project. The book addresses automated testing within a client-server or Web environment.

The focus of this book is the pragmatic concerns and information needed by the **software test engineer/manager** who faces the necessity of performing testing more thoroughly and quickly. By the same token, these same concerns may apply to the **software developer** who has the responsibility for development testing (that is, unit and integration testing) and, on some projects, system testing. The book also represents an informative guide that bolsters the ability of the **quality assurance engineer** to perform quality reviews on test design, test procedures, and the results of test activities.

The software **project manager**, who is responsible for the overall development effort, may also find this book useful. The text provides the project manager with guidelines concerning the goals and objectives of the testing effort and the decision about whether to automate testing. It also offers guidance on introducing automated testing on a project and outlines the processes for performing test planning, design, development, execution, and evaluation.

The authors have worked intimately with a number of automated testing professionals around the world, who were generous enough to share their problems and concerns. One primary concern voiced by these test engineers related to the fact that the test industry does not have the kind of structured methodologies that developers have traditionally enjoyed. Similarly, project managers, test managers, and test engineers may not be familiar with the kinds of approaches that are required to perform automated testing as opposed to the traditional test approach.

Clearly, the emphasis on automated testing represents a paradigm change for the software industry. This change does not simply involve the application of tools

and the performance of test automation. Rather, it is pervasive across the entire test life cycle and the system development life cycle. The approach taken by project managers, test managers, software engineers, and test engineers is altered as a result. For software professionals to successfully make the leap to automated testing, structured approaches to testing must be embraced.

Automated Software Testing is revolutionary in that it promulgates a new structured, building-block approach to the entire test life cycle, while also providing relevant test automation and associated test management guidance needed by industry test professionals.

Automated Testing

Software project managers and software developers building today's applications face the challenge of doing so within an ever-shrinking schedule and with minimal resources. As part of their attempt to do more with less, organizations want to test software adequately, but as quickly and thoroughly as possible. To accomplish this goal, organizations are turning to automated testing.

Faced with this reality and realizing that many tests cannot be executed manually, such as simulating 1,000 virtual users for volume testing, software professionals are introducing automated testing to their projects. While needing to introduce automated testing, software professionals may not know what's involved in introducing an automated test tool to a software project, and they may be unfamiliar with the breadth of application that automated test tools have today. *Automated Software Testing* provides guidance in these areas.

The growth of automated test capability has stemmed in large part from the growing popularity of rapid application development (RAD), a software development methodology that focuses on minimizing the development schedule while providing frequent, incremental software builds. The objective of RAD is to engage the user early and throughout design and development of each build so as to refine the software, thereby ensuring that it more closely reflects the needs and preferences of the user. In this environment of continual changes and additions to the software through each software build, where requirements are encouraged to evolve, software testing takes on an iterative nature itself. Each new build is accompanied by a considerable number of new tests as well as rework to existing test scripts, just as there is rework on previously released software modules. Given the continual changes and additions to software applications, automated software testing becomes an important control mechanism to ensure accuracy and stability of the software through each build.

As noted above, a primary goal of RAD is to shorten the overall development schedule, by addressing the riskiest aspects of development in early builds. As a

result, test activities are undertaken at the start of the initial RAD cycle and through each subsequent RAD cycle as well. As noted in Part III of this book, test design and development represent a complex undertaking. A test effort, in fact, may be as time-consuming as the effort required to develop the software application. When the project involves the integration of commercial off-the-shelf (COTS) products, for example, the test effort may even require more resources than software development. Instances where the test team does not participate in software specification or when test is not initiated soon enough pose a risk to the project. In these situations, potential outcomes include an incomplete software test effort, an insufficient test schedule, and an unplanned extension to the development schedule to accommodate testing.

Much of the test effort required on a project now needs to be supported by automated test tools. Manual testing is labor-intensive and error-prone, and it does not support the same kind of quality checks that are possible with an automated test tool. The introduction of an automated test tool can replace mundane manual test activities with a more efficient and repeatable automated test environment, which itself improves test engineer morale and retention.

Although some automated test tools began as capture and playback tools, the functionality and capabilities of automated test tool suites have been expanding. Automated test capabilities for software products include testing of the graphical user interface, requirements compliance, load performance, code coverage, Web interface, network communications, memory leakage, and more. New capabilities continue to be added to keep pace with the growing demand for test support.

ATLM—Automated Test Life-Cycle Methodology

This book concentrates on the concerns of the software test professional within the framework of an Automated Test Life-cycle Methodology (ATLM). ATLM is a structured methodology geared toward ensuring successful implementation of automated testing. The ATLM approach mirrors the benefits of modern, rapid application development efforts, where such efforts engage the user early in the development cycle. The end user of the software product is actively involved throughout analysis, design, development, and test of each software build, which is augmented in an incremental fashion.

The ATLM incorporates a multistage process consisting of six components. It supports the detailed and interrelated activities that are required to decide whether to acquire an automated testing tool. The methodology takes into account the process of introducing and optimizing an automated test tool and addresses test planning, analysis, design, development, execution, and management. The scope of the test program is outlined within the test plan, as a top-level description of test

approach and implementation. The scope is further refined through the definition of test goals, objectives and strategies, and test requirements. Similar to software application development, test requirements are specified before test design is constructed. Likewise, the test program must be mapped out and consciously designed to ensure that the most efficient and effective tests for the target application are performed. Test design is developed through graphical portrayals of the test effort, so as to give project and test personnel a mental framework on the boundary and scope of the test program.

Test Training

The evolution of automated testing has given birth to new career opportunities for software engineers. In fact, while the demand for automated software test professionals has exploded, community colleges and universities have not produced a reciprocal response to help support industry demand.

Universities, corporations, and government operations already have been proactive in responding to changes in the software industry and have implemented software test and quality assurance courses. For example, George Mason University (GMU) provides software test and quality assurance courses (see the GMU Web site at http://www.isse.gmu.edu/ for more information). Kansas State University (KSU) offers students several courses on the subject of software test and quality assurance and other courses that touch on the subject (see the KSU Web site at http://www.ksu.edu/).

Purdue University offers two undergraduate courses in software engineering that cover software testing and reliability. Purdue also has established a software engineering research center (http://serc.uoregon.edu/serc/), in which faculty from eight universities and representatives from eleven companies participate. Among other areas this center supports software quality research. For more information, see the Purdue University Web site at http://www.purdue.edu/.

The North Seattle Community College has established one of the most progressive testing curricula in the country. The college offers three levels of software testing courses (introduction, automation, and leadership) and one- and two-year software testing programs. For more information about these courses and programs, visit its Web site at http://nsccux.sccd.ctc.edu/. Information concerning additional university and industry training resources is available on the authors' Web site at http://www.autotestco.com/. For corporate and other test training organizations, see Table C.3 in Appendix C.

Automated Software Testing is intended to help in classroom instruction on software testing that uses modern automated test tool capabilities. The book provides

students with an introduction to the application and importance of software test, and it describes the different kinds of automated tests that can be performed. Instruction continues with the definition of the test approach, the roles and responsibilities of the test team, test planning, test design, test script development, test execution, defect tracking, and test progress reporting.

About the Authors

Automated Software Testing was developed by three software industry professionals.

Elfriede Dustin has performed as a computer systems analyst/programmer developing software applications and utilities, process and data modeling using CASE tools, and system design simulation models. She has experience supporting a variety of system application development projects, including health, financial, logistic, and enterprise information management systems. In addition, Elfriede has been responsible for implementing the entire development life cycle, from requirements analysis, to design, to development, to automated software testing. She has been a test manager and a lead consultant guiding the implementation of automated testing on many projects. Because of her automated test expertise, she has been sought out to help modify the capabilities of commercial test tool products, where her use of and feedback on test products have proved invaluable.

Jeff Rashka has managed a multitude of significant information system and systems integration projects. System applications on which he has served as manager include worldwide transportation asset management, enterprise information management, financial management, bar-coded inventory management, and shipboard information systems. Jeff also has process improvement management experience in implementing the guidelines contained within the Software Engineering Institute's Capability Maturity Model (CMM).

John Paul has worked as a senior programmer/analyst on financial and budgeting systems as well as a host of other information systems. His software development leadership responsibilities have included system analysis and design, application prototyping, and application development using a number of different methodologies and programming techniques. His software development responsibilities have included application testing using automated test tools. John has also assumed a lead role in the performance of year 2000 compliance testing.

The authors have applied their collective knowledge of software engineering, automated testing, and management to develop a book that addresses the pragmatic concerns and information needed by the software test engineer and manager. *Automated Software Testing* is designed to be a useful—and practical—guide for software engineers and software project managers who are responsible for software test activities.

Book Style and Organization

This book's organization correlates with the phases, tasks, and steps of the ATLM. The sequence of the book is fashioned in a purposeful way. It addresses the reader as if he or she had just been handed a note giving that individual the responsibility for automated software testing on a project. A first question might be, "What exactly is automated testing, and why do I need it?" Part I of the book answers this question and provides other fundamental instruction allowing the would-be test engineer to approach the new responsibility with confidence. The reader is then guided through the decision to automate testing as well as automated test tool selection and evaluation.

After receiving this fundamental instruction, the test engineer would have several more questions: "What is involved in setting up the tool?" "How do I get the test team in place?" "What early test planning is required?" Part II answers these questions. Specifically, the process for introducing an automated test tool is outlined and guidelines for structuring the test team are provided.

Part III focuses on automated test planning, analysis, design, and test automation (programming). This section of the book addresses test design techniques, which are comparable to the structured software design techniques introduced over the past 20 years. Specifically, it highlights the discipline required in the design of test automation. The goal is to give useful information on test design and test case development, so that the test engineer doesn't have to discover (by trial and error) how to put together a good test design and set of test procedures.

Part IV helps the reader address several additional questions: "What's involved in the performance of test?" "How do I manage my test schedule?" "How do I document and track defects?" This section provides guidelines pertaining to test execution, defect tracking, and test program status tracking. A set of best practices for the development and execution of automated test procedures is provided to assist test professionals in executing test activities in an efficient manner.

Overall, the book seeks to allay the concerns of the software test professional within the framework of the ATLM. ATLM is a structured methodology, and one that is geared toward ensuring successful implementation of automated testing. Readers with questions and comments may contact the authors via their home page at http://www.autotestco.com/. This Web site also provides more information on automated software testing and resources that are available to support automated software testing programs.

Acknowledgments

Special thanks to Oliver Jones and Chris Dryer for their enthusiasm and support of the book. Their encouragement, positive feedback, and valuable comments helped enhance the material presented here.

Thanks to all the testing professionals, such as Boris Beizer, for their test industry leadership and their persistent pursuit of software quality. We are especially grateful to all the individuals below whose contributions made this book possible.

- Brad Appleton
- Stacey Cornelius
- Matthias Daigl
- Chris Dryer
- Joyce Enos
- Robert L. Glass
- Sam Guckenheimer
- Dave Gustafson
- Richard J. Hedger
- Oliver Jones
- Anuradha Kare
- Bruce Katz
- Kirk Knoernschild
- Matthew Leahy
- Tilo Linz
- Ian Long
- Brett Schuchert
- Robert Schultz
- Andy Tinkham
- Will Tracz
- Chris Welch

ELFRIEDE DUSTIN
JEFF RASHKA
JOHN PAUL

CD-ROM Warranty

Addison Wesley Longman, Inc., warrants the enclosed disc to be free of defects in materials and faulty workmanship under normal use for a period of ninety days after purchase. If a defect is discovered in the disc during this warranty period, a replacement disc can be obtained at no charge by sending the defective disc, postage prepaid, with proof of purchase to:

Editorial Department
Computer and Engineering Publishing Group
Addison-Wesley
One Jacob Way
Reading, Massachusetts 01867-3999

After the ninety-day period, a replacement disc will be sent upon receipt of the defective disc and a check or money order for $10.00, payable to Addison Wesley Longman, Inc.

Addison Wesley Longman, Inc., makes no warranty or representation, either expressed or implied, with respect to this software, its quality, performance, merchantability, or fitness for a particular purpose. In no event will Addison Wesley Longman, Inc., its distributors, or dealers be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the software. The exclusion of implied warranties is not permitted in some states. Therefore, the above exclusion may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you may have that vary from state to state. The contents of this CD-ROM are intended for personal use only.

More information and updates are available at:
http://www.awl.com/cseng/titles/0-201-43287-0

Contents