Testing IT

Testing IT

An Off-the-Shelf Software Testing Process

John Watkins

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
10 Stamford Road, Oakleigh, Melbourne 3166, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published in 2000

Printed in the United States of America

Typeface: Sabon 11/13 pt. *System*: QuarkXPress® [GH]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

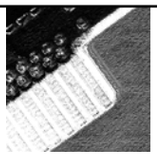
Watkins, John (John Edward)
Testing IT: an off-the-shelf software testing process handbook / John Watkins.
p. cm.
ISBN 0-521-79546-X (paperback)
1. Computer software – Testing. I. Title.

QA76.76.T48 W38 2001
005.1'4 – dc21 00-065143

ISBN 0 521 79546 X

To Francesca, Julie, and Valerie

Contents



Foreword

Why is astronomy considered a science while astrology is considered only a pseudo-science? In other words, how can we prove that a theory faithfully describes reality, and that this theory can then be used to predict unknown facts? Karl Popper, the well-known philosopher, studied these problems and summarized his conclusions in one phrase: "The criterion of the scientific status of a theory is its falsability, or refutability, or testability."* For Popper, "confirming evidence should not count except when it is the result of a genuine test of the theory."

The testing process of a scientific theory is quite similar to the process of providing confirmation either to risky predictions or to attempts to falsify that theory. Testing is a complex activity. It has to simultaneously bear in mind the theory and the external reality; it has to provide objective answers to complex questions related to our own perceptions of a rational reality.

When developing software, we follow the same thought process, since one builds an abstract model between the external world and the user. In our software, we define strict processes that will guide our actions, and we build the data we want to manipulate in complex databases and templates.

Can we test our software with Popper's principles in mind? The answer is definitively yes, because software testing should not only be a confirmation that the application is working correctly but also that it will react correctly when unexpected conditions occur. This constant and complex relationship between the software one tests and external reality should guide testers in their daily work.

Although testing is usually perceived as a necessity in software development, it is rarely applied as a rigorous activity. Within many projects, testing is simply omitted; in others, it is executed with the distinct intent to prove that the application performs correctly under test conditions.

After reading John Watkins's *Testing IT* you will be convinced that testing is not as complex as it seems and that it can be managed like any other development activity.

The first thing you notice in reading *Testing IT* is that John puts the accent on testing processes and real-world case studies, which are, in my opinion, the most important aspects of software testing, implicitly applying Popper's conclusions.

Testing IT is divided into three logical, distinct parts: Part 1 focuses on traditional testing processes. Although technology is evolving at lightening speed, processes remain. They become even more important because they are at the heart of any activity. You will find this part very useful since it presents the testing phases starting from unit testing up to regression testing in the order found on all projects. Each phase is presented in the same coherent plan, facilitating access to the information.

*Karl Popper, *Conjectures and Refutations*. London: Routledge and Kegan Paul, 1963.

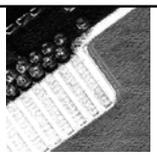
Part 2 gives practical case studies. Five projects are reported, each enhancing a different reality; we have again the confirmation that success is always related to a correct adaptation of processes to reality.

Part 3 presents ready-to-use templates and reference documents that you can adapt to your needs and that you will find very useful in your daily testing work.

Testing IT is not just another book on testing. It is, in addition, a guide for all testers who want to understand what testing is really about, and it proves once more that applying easy-to-understand processes is the key to success. In one word: indispensable.

I'm certain you will enjoy reading this book, as I did, and that you will keep it on your desk to profit in your daily work from John's rich experience.

Maurice Rozenberg, Paris
Author of *Test Logiciel* (1998, Eyrolles)



Acknowledgments

I would very much like to thank the following people for their advice, assistance, and encouragement with the writing of this book: Martin Adcock, Martin Allen, Steve Allott, Jo Armitage, Mike Ashton, Chris Ball, Bob Bartlett, Clive Bates, Judith Bays, Colin Bendall, Lesley Beasant, Duncan Brigginshaw, Kevin Buchta, Miriam Bromnick, Dorothy Buck, Dave Burgin, Yann Cluchey, Jon Collins, Trevor J. Collins, Richard Coop, Victor Copeland, Chris Cupit, Jim Davies, Steve Dawson, Barbara Eastman, Will Eisner, Isabel Evans, Mark Fewster, Virpi Flyktman, Ian Francis, Iain Gavin, Ian George, Sam Guckenheimer, Jon Hall, Marion Hampson, Steve Hancock, Richard Hands, Steve Harcombe, Kim Harding David Hayman, Mark Hirst, Dave Hodgkinson, Alan and Alison Jones, John Kent, Shelagh Kidd, Sami Korppi, Shaun Legeyt, Carol Li, Neil Love, Hatty and Nick Luft, Phillip March, Andrew Martin, Elaine Mathews, Peter McConnell, Alec MuCutcheon, Aidus McVeigh, Kevin Minier, Ian Murphy, Eric Nelson, Mike Newton, Tommi Oksanen, David "Saint" Palfreeman, Richard Pollard, Andy Redwood, Susan and Andy Roach, Angelina Samaroo, Ranj Sekhon, Darran Shaw, Graham Shore, David Shohet, Peter Terelak, Pete Thomas, Andrew Thompson, John Thompson, Richard Tinker, Paul Warne, Julie Valentine, Tony Wells, Martin Whitaker, and Helen White.

I would like to give particular thanks to Dorothy Graham for the benefit of her extensive experience in the testing field; Graham Titterington for his comprehensive and informed review comments; Geoff Quentin for his testing insight, encyclopedic knowledge of testing standards, and extensive collection of testing references; and James Bach, Paul Gerrard, and Steve Hancock for their input and informed discussion regarding risk-based testing issues.

I would also like to thank the following people for their assistance in writing the case studies in Part Two of this book, as well as for their general help and encouragement with the book and its contents: Morag Atkins, Ken Eves, Paul Goddard, Martin Kemble, David Marsh, Annette Philips, and James Toon.

I am also very grateful to Maurice Rozenberg for finding the time to write the Foreword, giving me the benefit of his extensive testing expertise in reviewing the chapters, case studies, and appendices, and giving me a signed copy of his testing book – *Test Logiciel*[1] (I will have to brush up on my French now so that I can actually read it!).

And last but certainly not least, I would like to express my appreciation for the insight and experience of my technical reviewer Kamesh Pemmaraju and for the constant "encouragement" and guidance from my editor Lothlorién Homet.

[1]Maurice tells me (winking) that this is the only French text on testing because the French do not make mistakes, so why would they need to test anything?

Testing IT