To Francesca, Julie, and Valerie
## Contents

Foreword xiii
Acknowledgments xv

### 1 Introduction

1.1 Purpose of the Book 1
1.2 Readership 2
1.3 How to Read This Book 2
1.4 Structure and Content of This Book 3

### Part 1

The Traditional Testing Process 5

### 2 An Overview of Testing

2.1 Introduction 7
2.2 The Challenge of Testing 7
2.3 What Is Testing? 8
2.4 Verification and Validation 10
2.5 What Is the Cost of Not Testing? 11
2.6 Testing – The Bottom Line 11
2.7 Additional Information 12

### 3 Testing Techniques

3.1 Introduction 15
3.2 General Testing Techniques 16
3.3 Functional Testing Techniques 19
3.4 Nonfunctional Testing Techniques 22
3.5 Further Reading on Testing Techniques 26

### 4 The Management and Planning of Testing

4.1 Introduction 27
4.2 The Organization of Testing 28
4.3 Roles and Responsibilities 29
4.4 The Testing Phases 34
4.5 Role of the V Model in Planning 39
4.6 The Management of Test Requirements 41
4.7 The Role and Use of Configuration Management 42
4.8 The Role and Use of Defect Tracking 42
4.9 The Role of Risk in Test Planning and Management 43

5 Unit Testing 45
5.1 Overview 45
5.2 Unit Test Approach 47
5.3 Unit Test Data Requirements 47
5.4 Roles and Responsibilities 48
5.5 Planning and Resources 49
5.6 Inputs 50
5.7 Testing Techniques for Unit Testing 50
5.8 Outputs 50

6 Integration Testing 53
6.1 Overview 53
6.2 Integration Test Approach 54
6.3 Integration Test Data Requirements 55
6.4 Roles and Responsibilities 55
6.5 Planning and Resources 56
6.6 Inputs 57
6.7 Testing Techniques for Integration Testing 57
6.8 Outputs 57

7 System Testing 59
7.1 Overview 59
7.2 System Test Approach 60
7.3 System Test Data Requirements 60
7.4 Roles and Responsibilities 61
7.5 Planning and Resources 62
7.6 Inputs 63
7.7 Testing Techniques for System Testing 63
7.8 Outputs 63

8 Systems Integration Testing 65
8.1 Overview 65
8.2 Systems Integration Test Approach 66
8.3 Systems Integration Test Data Requirements 67
8.4 Roles and Responsibilities 67
8.5 Planning and Resources 68
8.6 Inputs 69
8.7 Testing Techniques for Systems Integration Testing 69
8.8 Outputs 70

9 User Acceptance Testing 73
9.1 Overview 73
9.2 User Acceptance Test Approach 74
9.3 User Acceptance Test Data Requirements 75
9.4 Roles and Responsibilities 75
9.5 Planning and Resources 76
9.6 Inputs 77
9.7 Testing Techniques for User Acceptance Testing 78
9.8 Outputs 78

10 Operations Acceptance Testing 81
10.1 Overview 81
10.2 Operations Acceptance Test Approach 82
10.3 Operations Acceptance Test Data Requirements 83
10.4 Roles and Responsibilities 84
10.5 Planning and Resources 85
10.6 Inputs 86
10.7 Testing Techniques for Operations Acceptance Testing 86
10.8 Outputs 86

11 Regression Testing 89
11.1 Overview 89
11.2 Regression Test Approach 90
11.3 Regression Test Data Requirements 91
11.4 Roles and Responsibilities 92
11.5 Planning and Resources 92
11.6 Inputs 94
11.7 Testing Techniques for Regression Testing 94
11.8 Outputs 94

12 Improving the Testing Process 97
12.1 Introduction 97
12.2 Overview of the Role and Use of Metrics 98
12.3 Metrics Typically Used within the Testing Process 99
12.4 Setting Up and Administering a Metrics Program 102
12.5 A Proposal for a Simple and Effective Metrics Set 104
12.6 Further Reading 106
13 Introduction, Adoption, and Maintenance of the Testing Process 107

13.1 Introduction 107
13.2 Introduction and Adoption of a Testing Process 107
13.3 Maintenance of the Testing Process 111

Part 2
The Testing Process in the Real World: Illustrative Case Studies 113

14 Case Study 1: The British Library 121
15 Case Study 2: Reuters Product Acceptance Group 129
16 Case Study 3: Crown Quality Assurance Group 143
17 Case Study 4: The Wine Society 155
18 Case Study 5: Automatic Data Processing Limited 167

Part 3
The Appendices 181

A Terms of Reference for Testing Staff 183
B Testing Guides 197
C Test Plan Document Template 211
D Test Specification Document Template 221
E Test Script Template 231
F Test Result Record Form Template 237
G Test Log Template 241
H Test Certificate Template 245
I Re-use Pack Checklist 247
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>J Test Summary Report Template</td>
<td>249</td>
</tr>
<tr>
<td>K Equivalence Partition Example</td>
<td>255</td>
</tr>
<tr>
<td>L Boundary Analysis Example</td>
<td>257</td>
</tr>
<tr>
<td>M State Transition Example</td>
<td>259</td>
</tr>
<tr>
<td>N Automated Testing Tool Selection Criteria</td>
<td>261</td>
</tr>
<tr>
<td>O Usability Testing Overview</td>
<td>275</td>
</tr>
<tr>
<td>P Testing Process Health Check</td>
<td>279</td>
</tr>
<tr>
<td>Q The Testing of Object-Oriented Software</td>
<td>287</td>
</tr>
</tbody>
</table>

References 295
Glossary 297
Index 311
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 falsifiability, 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.

---

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)
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 Beaumont, Duncan Briggs, 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, VirpiFlyktman, 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, Shelaugh Kidd, Sami Korppi, Shaun Legeyt, Caro 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 (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 Lothloriën Homet.

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