1. Record Nr. UNINA9910799500103321

Autore	Forgacs Istvan
Titolo	Modern Software Testing Techniques : A Practical Guide for Developers and Testers / / by István Forgács, Attila Kovács
Pubbl/distr/stampa	Berkeley, CA : , : Apress : , : Imprint : Apress, , 2024
ISBN	1-4842-9893-4
Edizione	[1st ed. 2024.]
Descrizione fisica	1 online resource (XVIII, 266 p. 53 illus.)
Disciplina	005.14
Soggetti	Computer programs - Testing Software engineering - Management Software Testing Software Management
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Intro -- Table of Contents -- About the Authors -- About the Technical Reviewer -- Acknowledgments -- Introduction -- Abbreviations -- Chapter 1: Software Testing Basics -- Bugs and Other Software Quality Destroyers -- Lifetime of Bugs: From Cradle to Coffin -- Pesticides Against Bugs -- Classification of Bugs -- Software Testing -- Testing Life Cycle -- Test Planning -- Test Monitoring and Control -- Test Analysis -- Test Design -- Test Implementation and Execution -- Test Closure -- Fault-Based Testing -- Requirements and Testing -- Testing Principles -- 1. Testing is Possible -- 2. Early and Balanced Testing -- 3. Testing is Independent and Context Dependent -- 4. Continuity of Testing -- 5. Defect Clustering -- Two Misconceptions -- Comparison of the Existing and Our Principles -- Summary -- Chapter 2: Test Design Automation by Model-Based Testing -- Higher-Order Bugs -- Model-Based Testing -- One-Phase (Traditional) Model-Based Testing -- Two-Phase Model-Based Testing -- Stateless Modeling -- Use Case Testing -- Stateful Modeling -- FSM and EFSM-Based Modeling -- How to Select States? -- Model Maintenance -- How to Create a Stateful Model - Example -- Efficiency, Advantages, and Disadvantages -- Stateless and Stateful Together - Action-State Testing -- The Action-State Model -- Test Selection Criteria for Action-State Testing -- Creating Action-State Model -- Comparison

with Stateful Modeling -- How a Real Bug Can Be Detected? -- Summary -- Chapter 3: Domain Testing -- Equivalence Partitioning -- Obtaining Partitions Without Partitioning -- Example: Price Calculation -- Equivalence Partitioning and Combinatorial Testing -- Domain Analysis -- Test Selection for Atomic Predicates -- Selecting Tests for Predicates Comprising Two Atomic Components -- Closed Borders -- One Open and One Closed Border -- Two Open Borders -- Other Cases -- Summary.
Test Selection for General Compound Predicates -- Test Selection for Multidimensional Ranges -- Optimized Domain Testing (ODT) -- Boundary-Based Approach -- Example: Online Bookstore -- Rule-Based Approach -- Example: Online Bookstore Revisited -- Example: Paid Vacation Days -- Safety-Critical Aspects of ODT -- How ODT Can Help Developers -- ODT at Different Abstraction Levels -- Black-Box Solution -- Gray-Box Solution -- White-Box Solution -- Comparing ODT with Traditional Techniques -- Applying ODT with Other Techniques -- Summary -- Chapter 4: Developers and Testers Should Constitute a Successful Team -- How Developers Can Help Testers -- How Testers Can Help Developers -- How to Find Tricky a Tricky Bug -- Flaky Test -- Developer - Tester Synergies -- Summary -- Chapter 5: Conclusion -- Appendixes -- Appendix I: Java Code for Quicksort -- Appendix II: Test Cases for the Stateless Model of Car Rental Example -- Appendix III: Test Cases for Stateful Model of Car Rental Example -- Appendix IV: Test Cases for Action-State Model of Car Rental Example -- Appendix V: ODT Tool Description -- Glossary -- References -- Index.

Sommario/riassunto

Many books have been written about software testing, but most of them discuss the general framework of testing from a traditional perspective. Unfortunately, traditional test design techniques are often ineffective and unreliable for revealing the various kinds of faults that may occur. This book introduces three new software testing techniques: Two-Phase Model-Based Testing, the Action-State Testing, and the General Predicate Testing, all of which work best when applied with efficient fault revealing capabilities. You'll start with a short recap of software testing, focusing on why risk analysis is obligatory, how to classify bugs practically, and how fault-based testing can be used for improving test design. You'll then see how action-state testing merges the benefits of state transition testing and use case testing into a unified approach. Moving on you'll look at general predicate testing and how it serves as an extension of boundary valueanalysis, encompassing more complex predicates. Two-phase model-based testing represents an advanced approach where the model does not necessarily need to be machine-readable; human readability suffices. The first phase involves a high-level model from which abstract tests are generated. Upon manual execution of these tests, the test code is generated. Rather than calculating output values, they are merely checked for conformity. The last part of this book contains a chapter on how developers and testers can help each other and work as a collaborative team. You will: Apply efficient test design techniques for detecting domain faults Work with modeling techniques that combine all the advantages of state transition testing and use case testing Grasp the two-phase model-based testing technique Use test design efficiently to find almost all the bugs in an application.