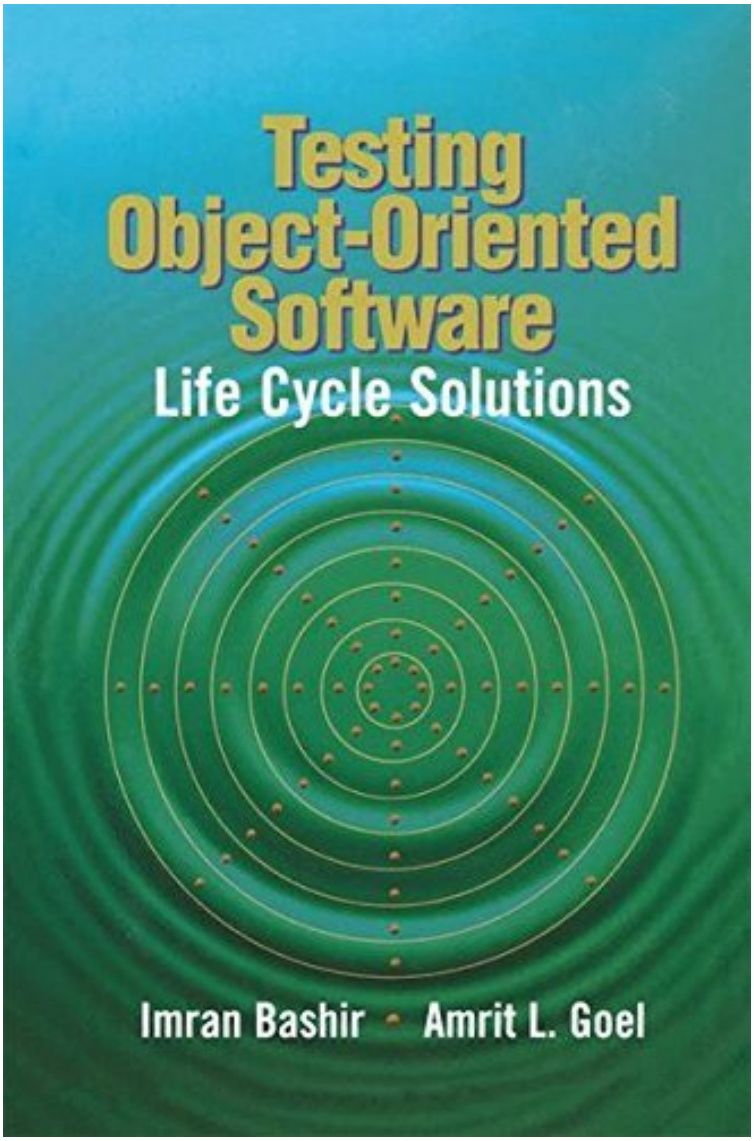
Testing Object-Oriented Software: Life Cycle Solutions

Von Imran Bashir, Amrit L. Goel
*ePub | *DOC | audiobook | ebooks | Download PDF*



Produktinformation Verffentlicht am: 2012-12-06Erscheinungsdatum: 2012-12-06File Name: B001GNBM7A | File size: 32.Mb

Von Imran Bashir, Amrit L. Goel : Testing Object-Oriented Software: Life Cycle Solutions before purchasing it in order to gage whether or not it would be worth my time, and all praised Testing Object-Oriented Software: Life Cycle Solutions:

KundenrezensionenHilfreichste Kundenrezensionen0 von 0 Kunden fanden die folgende Rezension hilfreich. OO Testing is differentVon George F. CorlissThis book's author scrutinized the software testing in a totally different way from most software testing books. The author focused on modern OO software testing, mainly glass box testing. It is

very helpful, because it covers some important changes for OO system testing, which are not included in most software testing books.Here are some points that impressed me most:1) Information Hiding and Abstraction: Although information hiding brings many benefits to programmers, it may be a strain on the life of a software tester. To test a method, if a tester wants to check the state of an object before and after the invocation of the method, he needs to access the internal state or data of that object. However, it is hidden to testers. Abstraction separates the essential behavior of an object form its implementation. An object can be tested as a black box using the abstraction of the object.2) Testability of Object-Oriented Systems: The author's definition of testability is a prediction of the probability of software failure occurring due to the existence of a fault. The definition implies the software testability is related to the ability of software to hide faults for a selected input distribution. Obviously, OO system's testability is lower than procedural counterpart.3) More opportunities for testers: Object orientation is employed to improve productivity and efficiency. Higher-complexity software is being produced in less time. This increased complexity is conducive to more error opportunities in novel ways. - Hai Huang

KurzbeschreibungAddressing various aspects of object-oriented software techniques with respect to their impact on testing, this text argues that the testing of object-oriented software is not restricted to a single phase of software development. The book concentrates heavily on the testing of classes and of components or sub-systems, and a major part is devoted to this subject. C++ is used throughout this book that is intended for software practitioners, managers, researchers, students, or anyone interested in object-oriented technology and its impacts throughout the software engineering life-cycle.KurzbeschreibungAddressing various aspects of object-oriented software techniques with respect to their impact on testing, this text argues that the testing of object-oriented software is not restricted to a single phase of software development. The book concentrates heavily on the testing of classes and of components or sub-systems, and a major part is devoted to this subject. C++ is used throughout this book that is intended for software practitioners, managers, researchers, students, or anyone interested in object-oriented technology and its impacts throughout the software engineering life-cycle.Synopsis This book is about testing object-oriented software. It addresses various aspects of object-oriented software techniques with respect to their impact on testing. It argues that the testing of object-oriented software is not restricted to a single phase of software development. Instead, it runs concurrently with other development activities. This book concentrates heavily on the testing of classes and of components or sub-systems. A major part of the book is devoted towards testing of classes, the basic unit of composition for object-oriented software programs. C++ is the language used in this book. The book is intended for software practitioners, managers, researchers, students, or anyone interested in object-oriented technology and its impacts throughout the software engineering life-cycle.