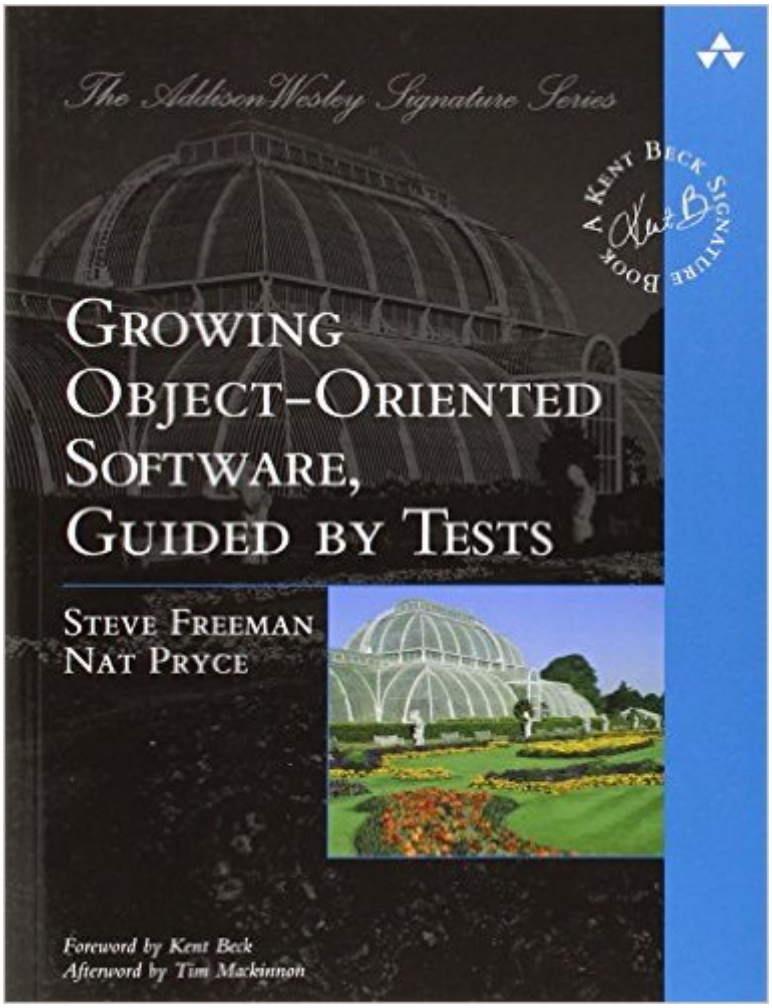
Growing Object-Oriented Software, Guided By Tests



Synopsis

Foreword by Kent Beck Â "The authors of this book have led a revolution in the craft of programming by controlling the environment in which software grows.â • --Ward Cunningham Â â œAt last, a book suffused with code that exposes the deep symbiosis between TDD and OOD. This one's a keeper.â • --Robert C. Martin Â â œIf you want to be an expert in the state of the art in TDD, you need to understand the ideas in this book.â •--Michael Feathers Â Test-Driven Development (TDD) is now an established technique for delivering better software faster. TDD is based on a simple idea: Write tests for your code before you write the code itself. However, this "simple" idea takes skill and judgment to do well. Now there's a practical guide to TDD that takes you beyond the basic concepts. Drawing on a decade of experience building real-world systems, two TDD pioneers show how to let tests guide your development and â œgrowâ • software that is coherent, reliable, and maintainable. Â Steve Freeman and Nat Pryce describe the processes they use, the design principles they strive to achieve, and some of the tools that help them get the job done. Through an extended worked example, youâ ™ll learn how TDD works at multiple levels, using tests to drive the features and the object-oriented structure of the code, and using Mock Objects to discover and then describe relationships between objects. Along the way, the book systematically addresses challenges that development teams encounter with TDD--from integrating TDD into your processes to testing your most difficult features. Coverage includes Â â ¢Â Â Implementing TDD effectively: getting started, and maintaining your momentum Â Â Â throughout the project â ¢Â Â Creating cleaner, more expressive, more sustainable code â ¢Â Â Using tests to stay relentlessly focused on sustaining quality â ¢Â Â Understanding how TDD, Mock Objects, and Object-Oriented Design come together Â Â Â in the context of a real software development project â ¢Â Â Using Mock Objects to guide object-oriented designs â ¢Â Â Succeeding where TDD is difficult: managing complex test data, and testing persistence Â Â Â and concurrency Â

Book Information

Paperback: 384 pages

Publisher: Addison-Wesley Professional; 1 edition (October 22, 2009)

Language: English

ISBN-10: 0321503627

ISBN-13: 978-0321503626

Product Dimensions: 7 x 0.9 x 9.1 inches

Shipping Weight: 1.4 pounds (View shipping rates and policies)
Average Customer Review: 4.5 out of 5 starsÂ Â See all reviewsÂ (44 customer reviews)
Best Sellers Rank: #65,455 in Books (See Top 100 in Books) #29 inÂ Books > Textbooks > Computer Science > Object-Oriented Software Design #30 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Testing #80 inÂ Books > Textbooks > Computer Science > Software Design & Engineering

Customer Reviews

There are many books about Test-Driven Development on the market, but this book is unique. It presents a style of TDD which originated in the London software development community. It’s a style which pushes several key ideas to the extreme: "tell, don’t ask" object design, fully end-to-end incremental development, and the deep synergy between testability and good design. Steve and Nat have done a stellar job refining and presenting these ideas. The text is lucid and precise. When you read this book you’ll be exposed to far more than just another style of TDD, you’ll be exposed to a depth of insight about emergent object oriented design which is both rare and extremely valuable.

This book has been in my pre-ordered list for quite a while and I was looking forward to this. I found the title alone already excellent. Steven and Nat (authors of jMock) are well known expert TDD practitioners, so I was looking forward to what they had to say. The book was better than I had expected.The book consists of 5 parts. The first part of a very quick introduction to TDD and jMock. The second part discusses the tdd cycle in more detail. The third part (150 pages) is a very large example of growing a piece of software. The fourth part discusses topics on how to sustain TDD and the last part covers some advanced topics.In this review, I’ll skip part 1 as it was short and nothing special. Part two covers the TDD cycle and the link to evolutionary design. Steve and Nat have a design style that focuses almost purely on the interactions between classes which are most frequently tested using expectations on mock objects (which, as authors of jMock, they have lots of experience with). Most notable from part 2, for me, were the classifications of objects that they used, the strong focus on interaction and mocking (more than I usually have when test-driving) and their lack of focus on classes but focus on roles and responsibilities. Nat and Steve clarify their thinking exceptionally well which makes it all easy to understand.Part 3 takes the largest part of the book, which is where they test-drive an AuctionSniper application. It is a small application, but large for a book example. The authors show how they gradually build up the application by adding one test at the time and how they gained insights during this process which made them adjust their design. I

had mixed feelings about this part as a book didn’t seem like the best medium for doing this, but still I appreciate the insights they had and also their attempt to make it as close to "real world" as possible.Writing tests is one thing, maintaining them in another. Part 4 discusses how to make the tests maintainable and the tdd cycle sustainable. Personally, I found this part very insightful and the authors discipline exemplar. The authors start of with different test smells and what to do about it. They then discuss readability of the tests and of the error messages and spend some time of test object creation. Most notable from that part (for me) was their focus on using builders for creating test data, rather than object mothers.The final part covers three (or actually two!) advanced topics. First is testing persistence where most interesting was how the authors seemed to prefer to "go all the way" whereas the common advise (for test speed) is to rollback and mock more. (this was actually a common theme in their book). The last two chapters deal with multi-threading and async code. I was unclear why these were separated in two chapters and they they were in this particular order. The content was excellent though, except that I missed some typical design guidelines related to multi-threading design. It almost felt they were in a hurry to write the last two chapters...Anyways, in conclusion, this will definitively be one of my favorite (if not the favorite) TDD books and general design books. Steven and Nat did a wonderful job on this one. Though the book is not perfect, I enjoyed it thoroughly. A definite recommendation for anyone interested in modern design and TDD.

This is a GREAT book... one of those you don’t wanna stop reading. But the kindle version sux so bad that I gave up when I saw the first code samples...A programming book with code samples that are almost impossible to read is a huge drawback. 70% of the value is lost in the kindle version.. I am sure that sooner or later this will be fixed but until then... stick to the printed version.

In a way this book presents the essence of a decade of test-driven development practice. The authors bring together the various tools of uptodate TDD like mock objects, bdd-style naming and acceptance tests. Open the book on any chapter and you will most likely find useful and deep advice, even if you consider yourself already a TDD expert.There’s one catch, though, that made me lower the rating to 4 stars: The authors go for an extended example which covers the full TDD cycle; from a walking skeleton, to the first acceptance test, into many obvious and some non-obvious refactorings. As noble as this endeavour is, it didn’t work for me as a reader. Coming back to the text - and the code - after a day or two I often got lost trying to grasp the subtle nuances; I just couldn’t remember all the necessary details of previous chapters.Nonetheless, it’s an excellent

book and I enjoyed it. Get a copy, read it and become a better TDD practitioner.

The person who handed me this book said it was "better than Lasse's book" (Test Driven.) I disagree. One can't compare the two books - Test Driven is meant for beginners and this book is meant for an advanced audience. If you have never written unit tests before, this book is very hard to follow. So put it down, get an intro book and come back.I really liked the emphasis on making the software responsive to change along with separating acceptance and unit tests. The book uses Junit 4.6 and therefore covers Hamcrest matchers for both JUnit and JMock. I like the authors cover best practices, good design and clearly indicate what is a personal preference. I really liked part 4's emphasis on things that are hard to test at a higher level than "extract method."The only thing that prevents me from giving full marks, is the case study. While I did read this part in one sitting, it was still hard to follow. There was a lot of information to keep in mind while trying to focus on the lessons of the example. I also think it was admirable for the authors to use a Swing example since Swing is harder to test. However, Swing is also less common for Java developers to use actively adding another block to understanding the software growing/testing aspects. And it is even harder for non-Java developers who are in the target audience for the book.Except for the case study, I thought the book was amazing. And I'm sure the case study is a matter of taste.---Disclosure: I received a copy of this book from the publisher in exchange for writing this review on behalf of CodeRanch.

Mathematics Tests (McGraw-Hill's 500 Questions) Object-Oriented Frameworks Using C++ and CORBA Gold Book: The Must-have Guide to CORBA for Developers and Programmers Distributed Object-Oriented Architectures: Sockets, Java RMI and CORBA Building Distributed, Object-Oriented Business Systems Using CORBA SNMP++: An Object-Oriented Approach to Developing Network Management Applications (Bk/CD-ROM) Object Oriented Programming with Swift 2 Smalltalk V 32-Bit Object-Oriented Programming System - Tutorial (1994 Win32 Series Version 3.0) Digitalk