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 “grow” software that is coherent, reliable, and maintainable.

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, you’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:

- Implementing TDD effectively: getting started, and maintaining your momentum throughout the project
- Creating cleaner, more expressive, more sustainable code
- Using tests to stay relentlessly focused on sustaining quality
- Understanding how TDD, Mock Objects, and Object-Oriented Design come together in the context of a real software development project
- Using Mock Objects to guide object-oriented designs
- Succeeding where TDD is difficult: managing complex test data, and testing persistence and concurrency

My Personal Review:
This book has been in my Amazon 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.

For More 5 Star Customer Reviews and Lowest Price: