Book reviews

Functional Programming for Java Developers – Tools for Better Concurrency, Abstraction, and Agility, By Dean Wampler, O'Reilly Media, July 2011, ISBN-13: 978-1449311032, 90pp.
doi: 10.1017/S0956796812000299

Functional Programming for Java Developers is a timely idea for a book. The concept behind it is that functional programmers have some really neat ideas over in the land of Haskell, ML & co, but it's too hidden behind obtuse mathematical-like concepts – so write a book that takes the best of functional programming and presents it in an approachable, pragmatic form for a Java developer. It's a good concept, but "Functional Programming for Java Developers" does not pull it off.

The major flaw in the book is that it tries to convert the ideas of functional programming too directly into Java – in programming terms, the book is too often a "low-level port" of exact features. The problem is that these features don't directly translate well into Java. The Maybe type is transformed into two pages of a class hierarchy of Option/Some/None, while lists are transformed into a lengthy Java equivalent of the cons head/tail representation. The overall impression is that functional programming makes your Java code long, ugly and unwieldy.

A better approach would surely have been to leave the list type abstract, and spend more time focusing on how to program with immutable lists, such as adding to the one example of using map and filter in the book. This single example is an instance of a bad functional programming habit: using horribly artifical examples to demonstrate the concepts. The example shows how to filter to keep even values, multiply them by 2 and sum them. It is doubtful that the intended audience – pragmatic Java programmers – are likely to see the benefits.

The book is well written, and the author clearly understands Java and functional programming. However, the author falls into the trap of many a functional programming advocate: having learnt all these wonderful new concepts in Haskell (or similar), the learner realises that they can program the exact same things (the Maybe type! the cons-style list! laziness! fold!) in an imperative language, which solidifies their own understanding of the concepts. However, explaining these connections at length (as this book does) is not really the best way to "sell" functional programming, especially to an intended audience who want to remain in Java but borrow a few ideas from Haskell.

Although the idea behind the book is timely, it was written too early to showcase the functional extensions to Java (and its standard library) that will accompany the forthcoming "Java 8". The book can only allude to the proposed lambda syntax, which would have made the code more readable. The book includes a chapter on concurrency, explaining actors and STM; but Java 8 will include classes and functions allowing for parallelism, with parallel map and so on. So the forthcoming practical advantages of functional style in Java (which all those Java developers may actually use) are nowhere to be found in the current edition of the book.

The book is short, with just 60 pages of content, and it shows: most of the content is covered only at a very superficial level. Concepts such as referential transparency and laziness are explained so briefly that someone without a functional programming background is very unlikely to understand them – and having been introduced, the concepts are barely mentioned

again. More time is spent covering immutability, and there the concept of the book begins to show through, with good, well-structured arguments as to why immutability is a good idea in Java: controlling the scope of mutation and making object-oriented programming easier and more reliable.

In summary: the concept for the book is good, but the realisation of the concept does not live up to the billing. The book is not terrible, but it feels worse for having missed a golden opportunity to explain how to use functional programming concepts in practical settings in a very popular object-oriented programming language. A greatly revamped second edition would be very welcome, if it could find the right level of abstraction at which to transfer the functional concepts into Java – backed by longer and more practical examples.

NEIL BROWN
School of Computing, University of Kent, Canterbury, UK

Steps in Scala: An introduction to Object-Functional Programming By Christos K. K. Loverdos, Apostolos Syropoulos, Cambridge University Press, 2010, 504 pp, ISBN 0521747589.
doi:10.1017/S0956796812000330

The Scala programming language[1] has recently gained popularity because of its use in some of the coolest companies of the moment. Twitter migrated its back-end infrastructure from Ruby on Rails to Scala, it seems that Foursquare is using Lift, a web development framework written in Scala, to serve their web pages and Linkedin is using the language for their large graph computations. So Scala can be considered among the most successful functional languages to date, at least in getting to the masses.

It is in this context that the book *Steps in Scala: An Introduction to Object-Functional Programming*, (SiS) is presented here. The book is intended as an introduction to the language and the functional style of programming. As such, the book is divided into two main parts. The first one, presents the language and the basic building blocks of Scala: basic types, flow constructions and operators, and it gradually moves into more advanced features, traits, pattern matching and the different kinds of polymorphism present in the language.

The audience is thus computer science students and professionals. The focus of this book is the functional paradigm, but if the reader comes from the imperative world, the learning curve for the more advanced functional concepts can be a little bit daunting. For instance, monads are presented in a eight page section, they are described using the full mathematical apparatus (category theory is employed when exposing the monad concepts), but they will not appear again in the rest of the book. The newcomer to functional programming will go through those pages with no idea about what is going on here.

When reviewing an introductory book on the Scala language one must compare it with *Programming in Scala: A Comprehensive Step-by-Step Guide* by the creator of the language, Martin Odersky, and others. While *Programming in Scala* aims to be the language reference and the whole book is devoted to presenting the language and all its intricacies, *Steps in Scala* tries to go beyond merely presenting the language: it introduces the language in the first three quarters of the book and then presents more advanced functional design patterns in the rest of the book by means of constructing non trivial applications. This is perhaps the strong point of the book: to present a novel programming language by doing hands on interesting problems on their own rather than plaguing the writing with small code snippets.

[1] `http://www.scala-lang.org/`