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/`

The problems that are addressed in the book range from how to build a parser with the tools that Scala provides to explaining the combinator pattern when building a portable system's path manipulation library. The book ends with a chapter describing how to build a basic computer algebra system plus a discussion about the *expression problem*, i.e. how to extend a given program's data types and operations without recompiling old code, and the strengths and weaknesses of the object and functional centric approaches and how Scala can benefit from both.

The main flaw of the book is the chapter on how to write graphical user interfaces in Scala. This chapter is 80 pages length, roughly twenty per cent of the whole book, and covers the usual content of GUIs tutorials, how to create frames, tabs, attaching buttons, listening to events and the like. Unless your interest in Scala is about writing GUIs the chapter's material can be found elsewhere in the Internet. The graphical elements are the common ones you can find in any Swing introductory text and adds nothing to the computer professional. The topic is uninteresting and it is something for which you will surely refer to the documentation of the language instead of this book when you need to. The chapter ends with a brief mention of functional graphics, a subject that could have deserved the attention of this chapter but as in the case of monads, it is only briefly sketched. All the pages in this chapter could have been used for more detailed explanations of other topics in the book.

A minor criticism for the book is the lack of industrial or "real" topics. Since functional programming and Scala in particular are going mainstream these days, some treatment of the basic tools could have benefited the computer professional. For instance, talking about the *Simple Build Tool*[2] and its dependency management system, the *Scalatest*[3] framework with its different Test Driven Development facets, or the jewel of the crown, the *Akka project*[4] for concurrent programming, just to name a few.

Steps in Scala: An introduction to Object-Functional Programming is a book that at times seems to be directed towards a highbrow academic audience but then it trims the more advanced material leaving the reader with a sense of improvisation or late additions. When exposing simple language concepts the book ranges from extreme conciseness (the main language traits are exposed in the first two chapters) to the unhelpful long exposition of graphical user interface construction mentioned earlier. The practical chapters about functional programming abuse the file system example and more diverse use cases should have been looked for. Although there are interesting chapters, like the one with the discussion of expression problem, newcomers to Scala will find it hard to learn the language using this book.

Toni Cebrián
(e-mail: `cebrian@tid.es`)

[2] `https://github.com/harrah/xsbt/wiki`
[3] `http://www.scalatest.org/`
[4] `http://akka.io/`