Book review: "Structure and Interpretation of Computer Programs" by Harold Abelson, Gerald Jay Sussman

May 28, 2008 at 07:38 Tags Book reviews , SICP

I have the habit of writing reviews for the books I've read, and recently I noticed that something is missing. SICP. I've definitely read it, so I should probably write a review, eh ?

The real reason for my writing this review is, however, its Amazon entry. Although the first page shines with glimmering reviews by, no less, Peter Norvig and Paul Graham, the book has a 3.5 star average, which is really inexplicable. Many far inferior books have higher grades! So I want to make my small contribution by writing SICP the review it deserves, give it the grade it deserves (5!!!!), and post it to Amazon as well. Lets get started, then.

My first encounter with SICP was in the second year of university, where I took a course named "SICP 1". I already had working programming experience by that time, so the course wasn't hard, but I was struck by the beauty of the subjects it taught, by the Scheme language it used and by the long and interesting homework assignments it had. This course woke my curiosity about functional programming, and the Lisp family of languages in particular.

A few years later, with much more experience behind my back, I've decided to tackle SICP seriously and read the whole book, solving the vast majority of the exercises. This endeavor took about 10 months, and I enjoyed it immensely. I think SICP is a wonderful book, a seminal work in the field of programming. It is one of those rare books every programmer should read. In this review I'll try to explain why.

In contrast with most introductory books about programming that just teach you a language, SICP takes a far better approach. The main goal of the authors is not to teach Scheme, it is to teach programming. From the beginning, the book takes an integrative path, where the basic axioms of programming are presented, and later are fortified with examples and exercises. SICP teaches about computational processes - iterative and recursive. How to use them best in each situation, and how to implement them. It also explains abstraction by functional composition. These are topics rarely presented in programming books, but SICP puts them rightly in the first chapter, because they are the real stuff programming is based on.

The example programs developed in the book are real, large, and exciting. You'll get to develop a powerful picture language, a generic object-oriented arithmetic package including complex and polynomial arithmetic, a simulator for digital circuits, a symbolic differentiation package, an interpreter for Scheme written in Scheme, an interpreter for a logic programming language similar to Prolog, a virtual machine for a simplified pseudo-assembly DSL, an interpreter for Scheme written in this pseudo-assembly, and finally a compiler from Scheme to the assembly language. All these examples are real, well-thought out exercises for skill, taken from beginning to a very complete end. In no other book such a wealth of topics is addressed in an accessible manner.

A word about exercises in SICP. They are numerous, some of them are hard, but the exercises are the best way to really understand what the book tries to teach. In a manner, they're as integral part of the book as the text itself. The exercises are very well prepared and lead the reader through the examples coded by the authors into greater understanding of the topics taught. At times, it feels like the exercises are specifically designed to force you to think about the essence of the topics, and not just grasp them superficially.

SICP commonly suffers from the criticism that it's too hard for beginners. Maybe this criticism is rightful, and universities should give a simpler introductory course to programming before SICP. But this is an aspect of the educational systems, not pertaining to the book itself. I wouldn't know, I never read SICP as a beginner. However it is being taught, SICP is an amazing book. It is by far the best programming book I have ever laid by hands on, and I seriously doubt that it will be surpassed any time soon. Reading SICP will enlighten you as a programmer, and make you a better one. I can't imagine one programmer who won't gain something important by reading SICP.

For comments, please send me an email.

Back to top