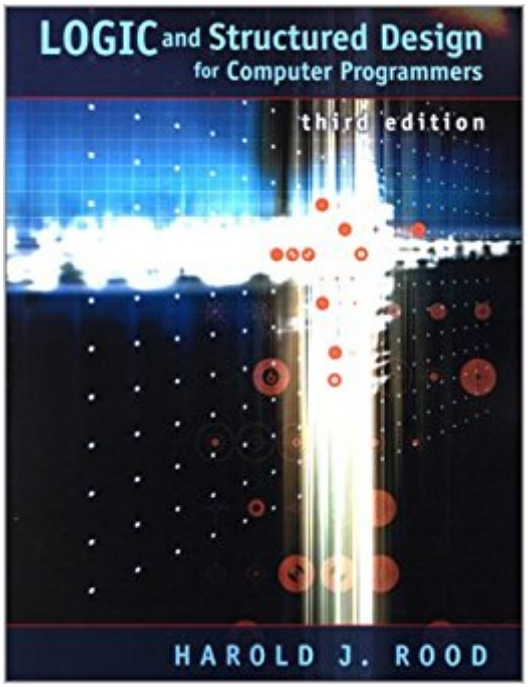
Logic And Structured Design For Computer Programmers



Synopsis

LOGIC AND STRUCTURED DESIGN is an introduction to the logic of data processing. It is intended for those who plan, but have not yet begun, to study programming, particularly those with little background in mathematics or logic. The author avoids reference to specific programming languages, isolating questions of logic from questions of syntax. This approach enables readers to concentrate on the logic of problems. The book walks readers through logical problems common to a variety of programming languages and provides the background in logic that many programming texts and courses assume.

Book Information

Paperback: 464 pages

Publisher: Course Technology; 3 edition (December 26, 2000)

Language: English

ISBN-10: 0534373860

ISBN-13: 978-0534373863

Product Dimensions: 10.9 x 8.5 x 0.8 inches

Shipping Weight: 2.5 pounds

Average Customer Review: 4.6 out of 5 stars 5 customer reviews

Best Sellers Rank: #456,588 in Books (See Top 100 in Books) #26 inÃ Â Books > Engineering & Transportation > Engineering > Electrical & Electronics > Circuits > Logic #77 inÃ Â Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Structured Design #98 inÃ Â Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Logic

Customer Reviews

Ph.D. Michigan State University --This text refers to an out of print or unavailable edition of this title.

Simple Question: How and why can this book be relevant at this writing (in 2016) given UML? Simple answer: the purpose of UML is to achieve a level of systems and program design abstraction that manages complexity, yet maintains enough precision to be machine readable. The old fashioned diagrams were thus like pseudocode vs. actual code, giving a representative, simplified way to understand what lies beneath the code, and why, while maintaining enough formality to avoid "custom" interpretations of the diagrams. This book is at the programming level, not the meta level of UML 2, for example, for systems rather than code design, or for portable code generation and

transportability, so you WILL need to graduate to UML at some point.So, think of this wonderful volume as a "prelude to UML" if you will-- exercising your brain with page after page that relates pseudocode to many logic diagrams, generally independent of language. In fact, UML IS independent of language, although most of us who teach programming think of it in terms of our favorite code and languages. But both this fine volume and UML can both be used with Julia, Java, R, or Haskell-- do it all the time. Of course UML was designed by the OOP community so does favor its modularity (as in Java/C#), but we work in functional languages all the time, and use both UML and this book without problems or misunderstandings.I teach both undergrad and grad programming, including assembly and circuit level sims for robotics and HPC, as well as the design of software proof assistants based on Topology. This won't show you how the tabs work in UML (for one of the best books I use for that, for .75c, try "the gorilla book" (Learning UML 2.0), but this book will help you understand the much more basic logic of good design and programming before tackling UML.You can decide whether the book is dated or timeless. I voted timeless because, even though the days of drawing little diamonds, circles and boxes with a pencil and template are thankfully gone, the logic of how the modules, classes, types, switches, variables, attributes, functions, properties, parameters, values, etc. work together are still a LOT easier to understand, especially for a beginner to programming, with diagrams and pseudocode, before delving into the tabs and workings of yet another language, and UML IS a language! If you're already an advanced UML user, you might find this dated or too simple, but, ironically, if you ever try to teach or explain what's under the UML, you'll see this as a gem more than a buggy whip (ironic because UML was created to standardize communication as well as manage complexity via "just right" abstraction). Highly recommended for the right reader.

I mean it's a textbook...for 2$... What cons are there

Professor Rood's book is about the best textbook in this subject that I have encountered in 20 years of teaching the material. It is intended for a CIS/MIS/IT or similar audience. It will provide an excellent introduction of a multitude of problem analysis methods and algorithm design languages (flowcharts, Nassi-Schneiderman, Warnier-Orr). It fosters structured programming and top down design as natural processes.Since the inside of an Object is procedure oriented code, this work is also an excellent reference for those who think they can jump into so-called "object oriented programming" without learning "old fashioned" procedure oriented first.They may be better texts for CS majors, but not for the rest.

I am just completing Dr. Rood's course at Washburn Univ. and from the student's perspective, I think his book is terrific. I found it to be both challenging and interesting, and I would say it's been an excellent tool for me in beginning to learn what programming is all about and how programs are structured. I would recommend it to anyone who is thinking about using it to teach a course, because I think most students find it a good and helpful book with good practice problems. I would also recommend it to anyone who's just interested in picking up a book about programming and trying to learn how things work.

First, I would say that overall this is a good book. Just that - not excellent, not inspiring, not... current. The techniques and principles mentioned in this book are applicable only if you really plan to throw out some of the cruft (with the predominance of OO languages such as Java, C++, Perl, Python and others, do we really need flowcharts?). Good logic and good structure are always required and this book gives you that - just remember there are other (better?) ways of doing design (UML anyone?). Also keep in mind that the way Warnier Orr is presented in this book deviates from the "standard" - so don't be surprised if you have some relearning to do.

Object-Oriented Design (7th Edition) 101 Design Methods: A Structured Approach for Driving Innovation in Your Organization Priests and Programmers: Technologies of Power in the Engineered Landscape of Bali Grokking Algorithms: An illustrated guide for programmers and other curious people Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages (Pragmatic Programmers)