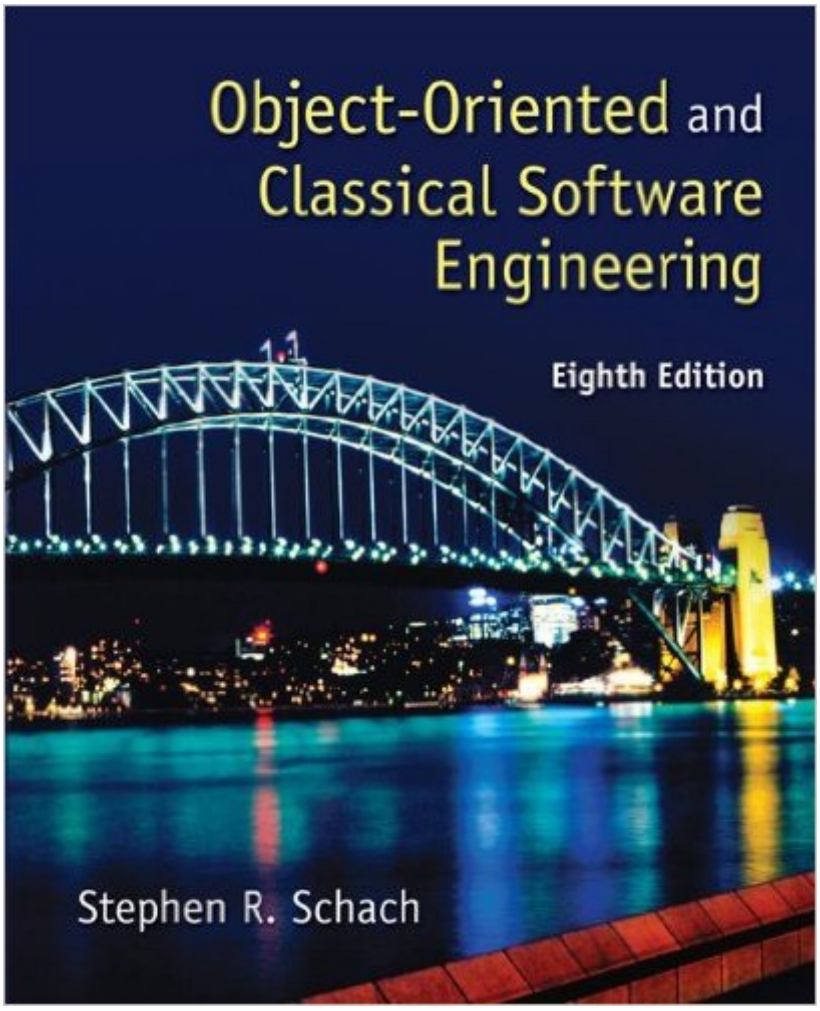
Object-Oriented And Classical Software Engineering



Synopsis

Building on seven strong editions, the eighth edition maintains the organization and approach for which Object-Oriented and Classical Software Engineering is known while making significant improvements and additions to content as well as problems and projects. The revisions for the eighth edition make the text easier to use in a one-semester course. Integrating case studies to show the object oriented approach to software engineering, Object-Oriented and Classical Software Engineering, 8/e presents an excellent introduction to software engineering fundamentals, covering both traditional and object-oriented techniques. While maintaining a unique organization with Part I covering underlying software engineering theory, and Part II presenting the more practical life cycle, the eighth edition includes significant revision to problems, new content, as well as a new chapter to enable instructors to better-utilize the book in a one-semester course. Complementing this well-balanced approach is the straightforward, student-friendly writing style, through which difficult concepts are presented in a clear, understandable manner.

Book Information

Hardcover: 688 pages
Publisher: McGraw-Hill Education; 8 edition (July 19, 2010)
Language: English
ISBN-10: 0073376183
ISBN-13: 978-0073376189
Product Dimensions: 7.5 x 1.3 x 9.4 inches
Shipping Weight: 2.6 pounds (View shipping rates and policies)
Average Customer Review: 3.7 out of 5 starsÂ Â See all reviewsÂ (26 customer reviews)
Best Sellers Rank: #393,211 in Books (See Top 100 in Books) #31 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > UML #156 inÂ Books > Textbooks > Computer Science > Object-Oriented Software Design #548 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Object-Oriented Design

Customer Reviews

Schach's Object-Oriented and Classical Software Engineering is a textbook in the traditional sense of the word. The book is divided into two parts. The first part, Introduction to Software Engineering, deals with software life-cycle models, teams, software engineering tools, and a few other general topics. The second part, The Phases of the Software Life Cycle, then takes a more detailed look at requirements, design, implementation, and so forth.The book has undergone a number of revisions

during its lifetime, and it shows. Schach discusses both structured and object-oriented methods, but the coverage isn't unified enough; the book feels like a quickly-made patch. I also wish that iterative development and agile methods had received more attention.To make matters worse, Schach's writing style doesn't impress me much. Ineffective passive constructions abound and modifiers always are put before the verb, even if the sentence contains a modal verb or the verb is be. This actually gets annoying after a while! What's more, Scach's approach is very, very prescriptive, and at least I frequently found myself objecting vehemently to the advice presented.It's sad that to my knowledge there are no really good general books on software engineering. Sommerville's book suffers from the exact same defects as Schach's. Can a software engineering book not be made accurate, thought-provoking, and fun to read?

I gave up on this book when I reached the extended example of object-oriented analysis, design and implementation. The analysis was ok; the design dropped a few elements without explanation, but was largely coherent.The implementation was a nightmare. It looked like procedural C++, with practically no relationship to the analysis and design.I think the book does a good job of conveying the time-tested key concepts behind software engineering. It should not be taken seriously as a discussion of object-oriented methodology.

The title of the book is very misleading, in that the book has very little uml and c++. Worse however is the lack of good problems and examples throughout the book. It seems like a watered-down version of Ghezzi's Software-Engineering book, the latter of which I would much more recommend. Some instructors may prefer Schach because it has *something* on uml and OOA&D, but it is probably better to supplement Ghezzi with a good book from this area (one I have yet locate).

My criticism of the book is not with its content. The content is fine for a text book. My criticism is on the price: $$?! The information is basic software engineering material found in numerous sources. The fundamentals that students need can be found in other texts just as well written and significantly more economical. If you are an instructor you may want to look at Pfleeger or even the Systems Analysis and Design book in the Cashman series.Again, this is a respectable reference and text book - the price is too much to ask of students though!

If you'd like, I can just plug my ears. Seriously, this book is so AWFUL, I can barely read it. Nothing is well explained. There are diagrams right in the middle of a flow of text (not in the good way). Let's

start with the explanations, they throw ’terms’ at you, like ’flow diagram’ then barely explain how to draw one, or what the symbols mean. Then have problems asking for answers which are based on their vague explanations. Later they put those symbols in the middle of the text. Right where you can’t read the text as easily. Not as a help, but as additional material. I am seriously considering contacting the chair of the department, asking this book not be used again. The internet is likely better resource.

Stephen R. Schach’s "Object-Oriented & Classical Software Engineering" (7ed) is an OK book: it’s not bad, but it could certainly be better.First, some minor quibbles: even though the typography and editing is good, I’m not all that enamored with the color scheme: the orange and black theme is too much like a pumpkin. I know it’s trivial, but I thought I’d just pass it along. A little more meaningful is that Schach seems to place too much emphasis on definitions. I don’t need multiple reminders of the differences between things like corrective, perfective and adaptive maintenance. It would be better if he just focused on the function and not on the definition. For university use, I suppose this is OK. But, I found it a bit irritating.The medium-level problem with the book is that there’s a lot of temporal shift in the presentation: he would talk about some model or methodology in terms that implied it was the latest and greatest thing. Yet, it had been around for decades. This is probably a function of the overall age of the book: this is the 7th edition.Most importantly, Schach needs to pick a methodology and stick with it: either talk about the classical methodology or the object-oriented one. Not both. Nowadays, most people probably work with, and are interested in, an object-oriented methodology. Having 1/3 of a book filled with the classical methodology is useless to them. Ditto for those people still working in a classical environment: they won’t care about 2/3 of the book. And, for those people who are in a classical environment and want to move to an object-oriented one, there’s really nothing in the book that will help them with the transition. If he removed the classical material from the book and published a "how to transition" book instead, that would be great.Again, it’s not a bad book. But, it’s not that great. I rate it at an OK 3 stars out of 5.

I got this book for my Software Engineering class, and have to say it is an easy read, and provides an in depth study on each topic introduced. It is hardcover (which is always a plus for me, I am not a big fan of paperback) and is not too large, where it is a pain to lug it around. The book is in color, which is nice for the many illustrations and diagrams presented in the text. In the end it is an easy read and has an in depth analysis on all the topics. While I got this for a class, I highly enjoyed it and plan to keep it for future reference.

Object Success : A Manager's Guide to Object-Oriented Technology And Its Impact On the Corporation (Object-Oriented Series) Reusable Software : The Base Object-Oriented Component Libraries (Prentice Hall Object-Oriented Series) Object-Oriented and Classical Software Engineering Object-Oriented Software Engineering: Practical Software Development Using UML and Java Object-oriented software development: Engineering software for reuse Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) [Economy Edition] Object-Oriented Software Engineering: Using UML, Patterns and Java (2nd Edition) Visual Object-Oriented Programming Using Delphi With CD-ROM (SIGS: Advances in Object Technology) Object-Oriented Technology and Computing Systems Re-Engineering Re-Engineering Business Solutions With Object-Oriented Development Design Patterns CD: Elements of Reusable Object-Oriented Software (Professional Computing) Growing Object-Oriented Software, Guided by Tests Design Patterns: Elements of Reusable Object-Oriented Software Design Patterns: Elements of Reusable Object-Oriented Software (Adobe Reader) Non-Functional Requirements in Software Engineering (International Series in Software Engineering) Software Engineering Classics: Software Project Survival Guide/ Debugging the Development Process/ Dynamics of Software Development (Programming/General) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition) Object-Oriented Frameworks Using C++ and CORBA Gold Book: The Must-have Guide to CORBA for Developers and Programmers Object-Oriented Analysis and Design for Information Systems: Modeling with UML, OCL, and IFML