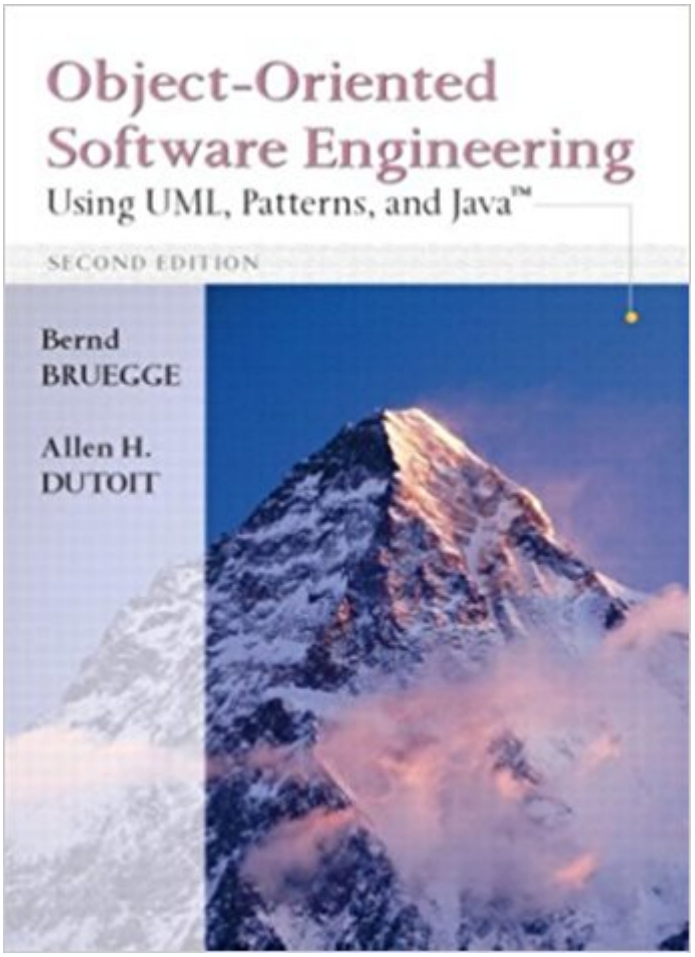
Object-Oriented Software Engineering: Using UML, Patterns And Java (2nd Edition)



Synopsis

For courses in Software Engineering, Software Development, or Object-Oriented Design and Analysis at the Junior/Senior or Graduate level. This text can also be utilized in short technical courses or short, intensive management courses. This textbook shows how to use both the principles of software engineering as well as the practices of various object-oriented tools, processes, and products. Using a step by step case study to illustrate the concepts and topics in each chapter, this book emphasizes practical experience: participants can apply the techniques learned in class by implementing a real-world software project.

Book Information

Hardcover: 800 pages

Publisher: Prentice Hall; 2 edition (October 5, 2003)

Language: English

ISBN-10: 0130471100

ISBN-13: 978-0130471109

Product Dimensions: 7.2 x 1.4 x 9.2 inches

Shipping Weight: 2.9 pounds

Average Customer Review: 3.8 out of 5 starsÂ Â See all reviewsÂ (25 customer reviews)

Best Sellers Rank: #1,358,977 in Books (See Top 100 in Books) #100 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > UML #462 inÂ Books > Textbooks > Computer Science > Object-Oriented Software Design #646 inÂ Books > Computers & Technology > Computer Science > Systems Analysis & Design

Customer Reviews

It is a highly readable book. The authors are good at explaining concepts with clarity.But the book is sloppy in any area that requires precision. They make no distinction of the four kinds of message sending in sequence diagrams. It is important for a UML user to differentiate synchronous, asynchronous, return and flat arrows. Otherwise a diagram will have different meaning. The authors use indiscriminately the notation of synchronous message when most of messages in their diagrams should be asynchronous.The coverage on OCL is even worse. More than half of the OCL constraints are wrong!!!You cannot rely on the corrections found on the authors' website because it only contains minor typos but misses the serious mistakes.Though it is more prescriptive than the standard software engineering books such as the ones by Pressman and Sommerville, I would NOT recommend its use as a textbook due to the many errors. I found "Object-oriented Systems Analysis

and Design" by Bennett, McRobb and Farmer a better how-to book in software engineering.

Many SE books tell you about SE (eg., Sommerville). Those kinds of books equip you to win in a software engineering version of the trivia game Jeopardy! but will hardly impart any skill and will not make you a better software engineer, only more informed.In contrast, this book tells you how to do software engineering. They tell you what, Bruegge shows you how. Rather than cover all the concepts in SE, Bruegge picks the most essential ones, gives you a brief but thorough explication of those and then proceeds to teach how they are used.Professor Bruegge’s approach to teaching his SE students is by having his entire class work *together* as one team on *one* real-life project during the term (that’s one project for the whole class).Typically, this project is an upgrade of the previous class’s project. Stop and imagine how realistic this approach is -- modifying a system created by engineers who are no longer available for interview, working with as many as 50 different people, working with designs that do not match the code anymore, working with code of varying quality, etc.Bruegge distills the lessons learned from these practical projects and illustrates practical (not idealistic) approaches to solutions.Expect German thoroughness and a lucid, unpretentious prose that heeds Strunk and White’s dictum: "Omit needless words".Highly recommended.-vja

Although this book comes from an academic background, I used it in a real client project in industry for the first time. The book offers a rather complete overview of software engineering in general: requirements engineering, analysis, system design, object design, implementation, testing. It also includes specialities, for instance rationale management, project management and others. I agree with a previous annotator who wrote that not all of the samples are ’perfectly helpful’. However, some are and some are quite amusing, e.g., in the Design Rationale chapter.Overall, the best collection of Software Engineering best practices I found in a single book. Really helpful for academic use as well as in industry.

This book was used in my software engineering class at college. Overall, the material in this book was presented in a very boring and complex manner, focusing on jargon definitions and a few isolated examples.Although the book does explain software engineering, it tends to do so in a painful way. Do yourself a favor and pick another book on the subject.

This is NOT a book on Unified Modeling Language (UML). It’s not a book on Object Constraint Language (OCL). It’s also not a book on Capability Maturity Models (CMM),

Class-Responsibilities-Collaborators (CRC) cards, Decision Representation Language (DRL), Extreme Programming (XP), Gantt charts, Issue-Based Information Systems (IBIS), Joint Appication Design (JAD), Key Process Areas (KPA), the Liskov Substitution Principle, Model-View-Controller (MVC) architectural styles, Nonfunctional Requirements (NFR) Frameworks, Object Design Documents (ODD), PERT charts, the Questions-Options-Criteria (QOC) model, Requirements Analysis Documents (RAD), Royce's methodology, Software Configuration Management Plans (SCMP), System Design Documents (SDD), Software Project Management Plans (SPMP), the Unified Software Development Process, User Manuals, V-Models, Work Breakdown Structures (WBS), or any of the myriad other tools introduced in the book.This IS a book to introduce newly-minted programmers to the kind of things, tools, and processes they can look forward to (with either anticipation or dread) in the real world of software development. As the authors state on page viii of the Preface:"We have observed that students are taught programming and software engineering techniques in isolation, often using small problems as examples. As a result, they are able to solve well-defined problems efficiently, but are overwhelmed by the complexity of their first real development experience, when many different techniques and tools need to be used and different people need to collaborate."It's been many years since I was involved in major software development projects (and those were all in the military). But, this book seems to have covered everything that all new programmers need to know so that they aren't simply lost when they enter their first software project. The readers certainly won't be experts in the things covered, but they'll at least have a good grounding and be able to bootstrap themselves from there (especially since the authors provide "Further Readings" and a Bibliography at the end of each chapter). For instance, on page 71, under Further Readings, they list three works on UML: one of which is the 566 page official specification, "OMG Unified Modeling Language Specification."Overall, this is an excellent book for anyone who is just entering the software development world. I rate it at 5 stars out of 5.As a side note, Florida State University (FSU) uses this book in its COP 3331: "Object-Oriented Analysis and Design" course.

We have various copies of this book in our offices - its a great book for software engineering and a must read for any software engineer, computer scientist or project manager. The material is thick and dives deep into the topic - but very much well worth it.

Object-Oriented Software Engineering: Using UML, Patterns and Java (2nd Edition) Object-Oriented Software Engineering Using UML, Patterns, and Java (3rd Edition) [Economy Edition]

Object-Oriented Software Engineering: Practical Software Development Using UML and Java 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) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition) Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) Object-oriented software development: Engineering software for reuse Object-Oriented Modeling and Design with UML (2nd Edition) Java: The Ultimate Guide to Learn Java and Python Programming (Programming, Java, Database, Java for dummies, coding books, java programming) (HTML, ... Developers, Coding, CSS, PHP) (Volume 3) JAVA: JAVA in 8 Hours, For Beginners, Learn Java Fast! A Smart Way to Learn Java, Plain & Simple, Learn JAVA Programming Language in Easy Steps, A Beginner's Guide, Start Coding Today! Patterns in Java: A Catalog of Reusable Design Patterns Illustrated with UML, 2nd Edition, Volume 1 Using UML: Software Engineering with Objects and Components (2nd Edition) Visual Object-Oriented Programming Using Delphi With CD-ROM (SIGS: Advances in Object Technology) The Object-Oriented Approach: Concepts, Systems Development, and Modeling with UML, Second Edition Object-Oriented and Classical Software Engineering Design Patterns CD: Elements of Reusable Object-Oriented Software (Professional Computing) Design Patterns: Elements of Reusable Object-Oriented Software Design Patterns: Elements of Reusable Object-Oriented Software (Adobe Reader) Object-Oriented Analysis and Design for Information Systems: Modeling with UML, OCL, and IFML