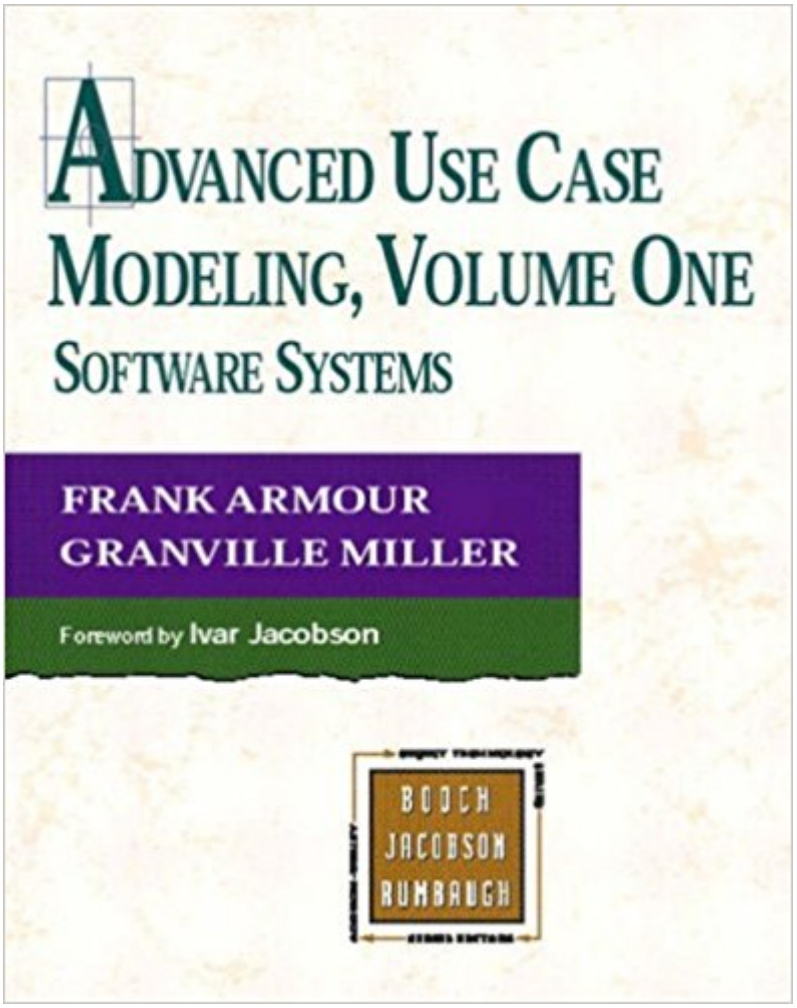
Advanced Use Case Modeling: Software Systems (v. 1)



Synopsis

used their wealth of experience to produce an excellent and insightful collection of detailed examples, explanations, and advice on how to work with use cases. Maria Ericsson The toughest challenge in building a software system that meets the needs of your audience lies in clearly understanding the problems that the system must solve. Advanced Use Case Modeling presents a framework for discovering, identifying, and modeling the problem that the software system will ultimately solve. Software developers often employ use cases to specify what should be performed by the system theyre constructing. Although use case-driven analysis, design, and testing of software systems has become increasingly popular, little has been written on the role of use cases in the complete software cycle. This book fills that need by describing how to create use case models for complex software development projects, using practical examples to explain conceptual information. The authors extend the work of software visionary Ivar Jacobson, using the Unified Modeling Language (UML) as the notation to describe the books models. Aimed primarily at software professionals, Advanced Use Case Modeling also include

Book Information

Paperback: 464 pages

Publisher: Addison-Wesley Professional; 1 edition (January 8, 2001)

Language: English

ISBN-10: 0201615924

ISBN-13: 978-0201615920

Product Dimensions: 7.3 x 1.1 x 9 inches

Shipping Weight: 1.5 pounds (View shipping rates and policies)

Average Customer Review: 4.6 out of 5 stars 8 customer reviews

Best Sellers Rank: #403,162 in Books (See Top 100 in Books) #159 inÂ Books > Textbooks > Computer Science > Object-Oriented Software Design #493 inÂ Books > Textbooks > Computer Science > Software Design & Engineering #544 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Object-Oriented Design

Customer Reviews

In this rapidly changing business and technological environment, use case modeling has emerged as one of the premier techniques for defining business processes and software systems. Business engineers now employ use cases to define complex business processes across lines of business and even to define entire businesses. Use cases are also the standard for defining requirements for

the software systems created using today's object-oriented development languages such as Java, Smalltalk, and C++. In the field of software components, a very young industry whose market is estimated to be more than $12 billion in 2001 Hanscome 1998, use cases are rapidly becoming a method of communication between suppliers and vendors. The users of this technique for defining systems are as diverse as its uses. Use case modeling is already being employed by most Fortune 1000 companies and is being taught at many academic institutions all over the world, and the popularity of this modeling technique continues to grow. Business process and software requirements engineering are rapidly evolving fields. Research in these areas continues to propose new methods of dealing with potential problems, even while actual practice is slow to adopt only a fraction of those proposed. This slow-moving partial adoption has been termed the "researchâ "practice gap" Berry 1998. Creating yet another use case book without an extensive experience base would merely add to this gap. Our approach is significant because we present a practitioner's approach firmly grounded in the real world. Goals Over the past six years, we have worked on some large, ambitious projects involving software development and business engineering. To create the best possible use case models, we found it necessary to extend the seminal work of Ivar Jacobson in certain areas. This book details our extensions, which complement Ivar's ongoing work. The flexibility of use case modeling and the Unified Modeling Language, which we use to describe these models, allows us to produce extensions to solve real-world problems successfully. The goal of this book is to further the advancement of use case modeling in software and business engineering. To achieve this goal, the book provides a comprehensive yet readable guide to use case modeling for the practitioner. Specifically, it explains advanced use case modeling concepts, describes a process for implementing use case modeling, and discusses various use case modeling issues. Audience The audience for this book is anyone involved in the conceptualization, development, testing, management, modeling, and use of software products and business processes. Although it contains a sizable amount of content related to business processes, this book is geared toward all of us in the software industry. Software professionals are the largest body of use case engineers because use case development was first introduced as a software requirements vehicle. Business analysts will agree that use case engineering has undergone the greatest transformations on their front. Business analysts and their software process brethren are quickly learning that automation via software is not the only reason for employing use cases. In fact, more and more of business process modeling using use cases is not geared toward the generation and production of new software but is being done to understand, and in some cases, standardize and optimize key business processes across multiple lines of business. Many of the

techniques described in this book transcend the software or business arenas of the reader community. The well-established link between business use cases and software system use cases is described as we illustrate the ways in which software systems can be derived from a business process. The only thing we ask is that our business readers be patient as we start on the software side. Academic institutions will also find this book useful. This book can be used as a text in an object-oriented analysis (OOA) course in which use cases play a key role. How to Use This Book The theory of use case development often differs from the actual practice of use case development. One reason for this difference is that very few software development projects are "green fields

"This book isn't just another introduction to use cases. The authors have used their wealth of experience to produce an excellent and insightful collection of detailed examples, explanations, and advice on how to work with use cases." â ”Maria Ericsson The toughest challenge in building a software system that meets the needs of your audience lies in clearly understanding the problems that the system must solve. Advanced Use Case Modeling presents a framework for discovering, identifying, and modeling the problem that the software system will ultimately solve. Software developers often employ use cases to specify what should be performed by the system they're constructing. Although use case-driven analysis, design, and testing of software systems has become increasingly popular, little has been written on the role of use cases in the complete software cycle. This book fills that need by describing how to create use case models for complex software development projects, using practical examples to explain conceptual information. The authors extend the work of software visionary Ivar Jacobson, using the Unified Modeling Language (UML) as the notation to describe the book's models. Aimed primarily at software professionals, Advanced Use Case Modeling also includes information that relates use case technique to business processes. This book presents a process for creating and maintaining use case models in a framework that can be fully customized for your organization. The authors, pioneers in the application of use cases in software development, bring their extensive experience to cover topics such as: A process model for applying a use case model How to keep your use case modeling effort on track Tips and pitfalls in use case modeling How to organize your use case model for large-system development Similarities between Advanced Use Case Modeling and the Rational Unified Process framework Effect of use cases on user interface design Guidelines for quality use case modeling 0201615924B04062001

Great book for class and came fast! It's easy to read/comprehend and good to have for reference.

Good Book

This is a very good book on use cases. Concepts are well-explained. Use cases are placed properly into the UML framework. Typical forms and diagrams are presented and discussed. The process of developing and refining use cases gets full and detailed attention, as does testing.So - good instructional material, and the book is compact and well-organized enough to also serve as a reference. It is crystal clear and very readable.What follows below is not more book review. If my experiences and commentary regarding use cases interests you, read on, but be aware that it doesn't bear directly on the book itself.My background first: I started as a developer, moved into technical management, project management, business analysis management, IT senior management, and currently serve in an IT governance role in one of the largest companies in the US.I'm not an enthusiastic use case advocate. Having managed or had oversight for more development activities than I care to recall over the last 30 years, I've only run into a handful of people who were competent enough to produce a set of use cases usable and complete enough to feed into the next design stage. (Understanding, of course, that this methodology was in its infancy during part of that period.) Many claim to know use cases and modeling. Then again, most developers involved in any kind of distributed systems, web, n-tier, PC-based development, etc. claim to be OO competent, too (Uh-huh ...). I've written use cases and developed apps from them and still only consider myself marginally competent. I used a consultant to help then, and would do so again. Use cases are exceedingly difficult to write well for non-trivial applications and, because of their text content, deceptively simple-looking. Don't be fooled.The few I've encountered who could produce good use cases have invariably been knowledgeable technical leads who were competent across the entire span of activities from requirements through post-deployment support. They were also intimately familiar with the particular business space. This is not an easy combination to find.That those who best understand use cases are typically highly technical is kind of perverse, as the intent of use cases is to keep them firmly in the end-user's conceptual space. In trying - and failing repeatedly - to get end users and business analysts trained in use cases - and, for that matter, trying to get the business to understand and accept them, I finally came up with something that worked. That is, develop conventional, text-based project charters and requirements documents, then have the dev teams and leads develop the use cases from the business docs.This results in an interative process between use case development and the requirements - adding work - but it also cuts the inexpert out of the parts of the process that they never quite fathom anyway,

such as alternative flows, extensions and generalization, includes relationships, etc. Even pre and post-conditions, which business people intuitively understand, are problematic because they typically can't get enough rigor into a use case without assistance, so the iteration helps with that also. What you wind up with is use case documention in the technical space, not the users'. That turns the process on its head but results in usable use cases. You also have to adjust how you handle non-interface content such as verification when end users aren't directly participating in use case creation. Such workflow and documentation issues lead to the next point:Further complicating the adoption of use case modeling is the fact that use cases (and the UML) are easiest to adopt when supported by an end-to-end software suite, like Rational's. Anyone who has tried to implement such a suite knows that it is excruciating. Rational themselves will tell you that the setup of the software and underlying database is critically dependent on the specific forms approach you use, your methodology variation and workflow, even your org structure. The message is that you have to have a very full understanding of what you're doing in order to implement. Most bring Rational in on a consulting basis as a result.Adoption is also complicated in large firms by the simple fact that most activity is on legacy systems, and that modern modeling and design methodologies aren't well-suited to those systems.The final comment I'd make is on value. I haven't found that the adoption of use cases - even good ones - yields better systems or even better documentation than other means of systems design and interface functional specification.Returning again to the book itself - by all means buy it if you are looking for a single volume focused on use case development. It really is a good use case text. Beware the pitfalls going down the use case adoption route, however. If ever there was something that deserved a pilot, this would be it.

Granville Miller and Frank Armour have created an essential text for understanding Use Case Modeling theory. This book explains the basics of UML in the initial chapters, but quickly moves on to detail advanced Use Case theories and the best ways to apply those theories. I approached the book as a beginner, but had little difficulty understanding the presented ideas and theories. I would recommend this book for anyone involved with Use Case Modeling - from beginner to advanced. For beginners, this book offers a solid introduction and quickly prepares you for the advanced topics. For intermediate to advanced users, this book offers a compilation of theories and practices and is certain to give you insight to pieces of the UML puzzle.Each step of the design process is explained thoroughly, and several alternative procedures are presented. Also, the appendices are valuable references of themselves. They contain a Use Case Development Review Checklist and a complete Development Case, which outlines each major step on the use case modeled

development cycle.The authors have also done an excellent job in bringing together information from outside sources to compile their work. Rather than preaching a specific format or model, the reader is presented with many different customizable options for applying the theories in the book. The carefully cited sources also give excellent direction for further reading.I was disappointed to find a flaw with the printing of the book. In my copy the pages containing the table of contents were out of order. However, I found that I was more disappointed because a potential reader might skip this book on the shelf just because of a printing mishap. This information within is too valuable to miss.A real danger with theory books is the potential to either underestimate the reader or talk over the reader's head. This book walks that fine line with ease. I felt that the ideas and terms were presented in a logical and clear manner. It is a valued reference for my work.

I have found this book of tremendous help in my work.My first books on use cases focused more on UML rather than use cases. I did not give a hoot on use cases, because they look so simple on paper (and that's why I didn't buy a book specifically on use cases!) But as I grew as a developer, I began to believe that use case modelling if done well can significantly reduce development effort and bring about quality solutions. Use cases are the foundation to the understanding of the system that you are trying to develop. Use cases deserve serious attention.The main problem with use cases is that you either don't know how to start or when to stop. This book tells you both. It tells you how to develop your use case model systematically from scratch and how to make provisions so that your use case model can grow. IMO, that's the main draw for this book.The authors also give good insights on the possible approaches the reader can take to expand his/her use case model iteratively. It cautions the modeller to keep a balanced model so that stakeholders can understand, rather than one that specifies everything but gets bogged down by the details.Semantics, you can get it elsewhere, but this book discusses it pretty well too. The examples are clear and relevant.All in all, Frank and Granville did an excellent job covering the topic.

(Addison-Wesley Object Technology (Paperback)) Rapid Prototyping Software for Avionics Systems: Model-oriented Approaches for Complex Systems Certification (Iste) The Software Requirements Memory Jogger: A Pocket Guide to Help Software And Business Teams Develop And Manage Requirements (Memory Jogger) Head First Software Development: A Learner's Companion to Software Development Agile Project Management: Agile Revolution, Beyond Software Limits: A Practical Guide to Implementing Agile Outside Software Development (Agile Business Leadership, Book 4) Don't Buy Software For Your Small Business Until You Read This Book: A guide to choosing the right software for your SME & achieving a rapid return on your investment Software Agreements Line by Line, 2nd ed.: A Detailed Look at Software Agreements and How to Draft Them to Meet Your Needs IEC 62304 Ed. 1.0 b:2006, Medical device software - Software life cycle processes Agile Software Development with Scrum (Series in Agile Software Development) Scooby-Doo Set of 8 Mystery Chapter Books (Haunted Castle ~ Snow Monster ~ Fairground Phantom ~ Spooky Strikeout ~ Case of the Haunted Hound ~ Case of the Living Doll ~ Case of the Spinning Spider ~ The Creepy Camp) The Model's Bible & Global Modeling Agency Contact List - An Insider's Guide on How to Break into the Fashion Modeling Industry Modeling Agency Tips: Get Listed with Fashion Modeling Agencies and Find Your Dream Job 3ds Max Modeling for Games: Insider's Guide to Game Character, Vehicle, and Environment Modeling: Volume I Atmospheric and Space Flight Dynamics: Modeling and Simulation with MATLABÂ® and SimulinkÂ® (Modeling and Simulation in Science, Engineering and Technology)