

Applying Uml Patterns 3 Ed Chapter 15

Second Edition of the UML video course based on the book Applying UML and Patterns. This VTC will focus on object-oriented analysis and design, not just drawing UML.

This is the eagerly-anticipated revision to one of the seminal books in the field of software architecture which clearly defines and explains the topic.

Software -- Software Engineering.

"This thoroughly updated text teaches students or industry R & D practitioners to successfully negotiate the terrain for building and maintaining large, complex software systems. The authors introduce the basic skills needed for a developer to apply software engineering techniques. Next, they focus on methods and technologies that enable developers to specify, design, and implement complex systems. Finally, the authors show how to support the system changes throughout the software life cycle."--BOOK JACKET.Title Summary field provided by Blackwell North America, Inc. All Rights Reserved

This is the definitive guide for managers and students to agile and iteratedevelopment methods: what they are, how they work, how to implement them, andwhy they should.

Object-oriented Software Engineering

Enterprise Patterns and MDA

Writing Effective Use Cases

Object-Oriented Design with UML and Java

Guide to the Unified Process featuring UML, Java and Design Patterns

Building Better Software with Archetype Patterns and UML

A Manager's Guide

Using research in neurobiology, cognitive science and learning theory, this text loads patterns into your brain in a way that lets you put them to work immediately, makes you better at solving software design problems, and improves your ability to speak the language of patterns with others on your team.

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 in short, intensive management courses. Shows students how to use both the

principles of software engineering and 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, Bruegge and Dutoit emphasize learning object-oriented software engineer through practical experience: students can apply the techniques learned in class by implementing a real-world software project. The third edition addresses new trends, in particular agile project management (Chapter 14 Project Management) and agile methodologies (Chapter 16 Methodologies).

Social scientists, whether earnest graduate students or tenured faculty members, clearly know the rules that govern good writing. But for some reason they choose to ignore those guidelines and churn out turgid, pompous, and obscure prose. Distinguished sociologist Howard S. Becker, true to his calling, looks for an explanation for this bizarre behavior not in the psyches of his colleagues but in the structure of his profession. In this highly personal and inspirational volume he considers academic writing as a social activity. Both the means and the reasons for writing a thesis or article or book are socially structured by the organization of graduate study, the requirements for publication, and the conditions for promotion, and the pressures arising from these situations create the writing style so often lampooned and lamented. Drawing on his thirty-five years' experience as a researcher, writer, and teacher, Becker exposes the foibles of the academic profession to the light of sociological analysis and gentle humor. He also offers eminently useful suggestions for ways to make social scientists better and more productive writers. Among the topics discussed are how to overcome the paralyzing fears of chaos and ridicule that lead to writer's block; how to rewrite and revise, again and again; how to adopt a persona compatible with lucid prose; how to deal with that academic bugaboo, "the literature." There is also a chapter by Pamela Richards on the personal and professional risks involved in scholarly writing. In recounting his own trials and errors Becker offers his readers not a model to be slavishly imitated but an example to inspire. Throughout, his focus is on the elusive work habits that contribute to good writing, not the more easily learned rules of grammar and punctuation. Although his examples are drawn from sociological literature, his conclusions apply to all fields of social science, and indeed to all areas of scholarly endeavor. The message is clear: you don't have to write like a social scientist

to be one.

"This book manages to convey the practical use of UML 2 in clear and understandable terms with many examples and guidelines. Even for people not working with the Unified Process, the book is still of great use. UML 2 and the Unified Process, Second Edition is a must-read for every UML 2 beginner and a helpful guide and reference for the experienced practitioner."

--Roland Leibundgut, Technical Director, Zuehlke Engineering Ltd. "This book is a good starting point for organizations and individuals who are adopting UP and need to understand how to provide visualization of the different aspects needed to satisfy it. " --Eric Naiburg, Market Manager, Desktop Products, IBM Rational Software This thoroughly revised edition provides an indispensable and practical guide to the complex process of object-oriented analysis and design using UML 2. It describes how the process of OO analysis and design fits into the software development lifecycle as defined by the Unified Process (UP). UML 2 and the Unified Process contains a wealth of practical, powerful, and useful techniques that you can apply immediately. As you progress through the text, you will learn OO analysis and design techniques, UML syntax and semantics, and the relevant aspects of the UP. The book provides you with an accurate and succinct summary of both UML and UP from the point of view of the OO analyst and designer. This book provides Chapter roadmaps, detailed diagrams, and margin notes allowing you to focus on your needs Outline summaries for each chapter, making it ideal for revision, and a comprehensive index that can be used as a reference New to this edition: Completely revised and updated for UML 2 syntax Easy to understand explanations of the new UML 2 semantics More real-world examples A new section on the Object Constraint Language (OCL) Introductory material on the OMG's Model Driven Architecture (MDA) The accompanying website provides A complete example of a simple e-commerce system Open source tools for requirements engineering and use case modeling Industrial-strength UML course materials based on the book

This comprehensive guide has been fully revised to cover UML 2.0, today's standard method for modelling software systems. Filled with concise information, it's been crafted to help IT professionals read, create, and understand system artefacts expressed using UML. Includes an example-rich tutorial for those who need familiarizing with the system.

Elements of Reusable Object-Oriented Software
An Introduction to Object-Oriented Modeling
Applying UML and Patterns Training Course
A Desktop Seminar from Craig Larman
Object-Oriented Software Engineering Using UML, Patterns, and Java
Design Patterns Applied
Advances in Conceptual Modeling

This guide will help readers learn how to employ the significant power of use cases to their software development efforts. It provides a practical methodology, presenting key use case concepts.

"One of the great things about the book is the way the authors explain concepts very simply using analogies rather than programming examples—this has been very inspiring for a product I'm working on: an audio-only introduction to OOP and software development." —Bruce Eckel "...I would expect that readers with a basic understanding of object-oriented programming and design would find this book useful, before approaching design patterns completely. Design Patterns Explained complements the existing design patterns texts and may perform a very useful role, fitting between introductory texts such as UML Distilled and the more advanced patterns books." —James Noble Leverage the quality and productivity benefits of patterns—without the complexity! Design Patterns Explained, Second Edition is the field's simplest, clearest, most practical introduction to patterns. Using dozens of updated Java examples, it shows programmers and architects exactly how to use patterns to design, develop, and deliver software far more effectively. You'll start with a complete overview of the fundamental principles of patterns, and the role of object-oriented analysis and design in contemporary software development. Then, using easy-to-understand sample code, Alan Shalloway and James Trott illuminate dozens of today's most useful patterns: their underlying concepts, advantages, tradeoffs, implementation techniques, and pitfalls to avoid. Many patterns are accompanied by UML diagrams. Building on their best-selling First Edition, Shalloway and Trott have thoroughly updated this book to reflect new software design trends, patterns, and implementation techniques. Reflecting extensive reader feedback, they have deepened and clarified coverage throughout, and reorganized content for even greater ease of understanding. New and revamped coverage in this edition includes Better ways to start "thinking in patterns" How design patterns can facilitate agile development using eXtreme Programming and other methods How to use commonality and variability analysis to design application architectures The key role of testing into a patterns-driven development process How to use factories to instantiate and manage objects more effectively The Object-Pool Pattern—a new pattern not identified by the "Gang of Four" New study/practice questions at the end of every chapter Gentle yet thorough, this book assumes no patterns experience whatsoever. It's the ideal "first book" on patterns, and a perfect complement to Gamma's classic Design Patterns. If you're a programmer or architect who wants the clearest possible understanding of design patterns—or if you've struggled to make them work for you—read this book.

Larman covers how to investigate requirements, create solutions and then translate designs into code, showing developers how to

make practical use of the most significant recent developments. A summary of UML notation is included

The practice of enterprise application development has benefited from the emergence of many new enabling technologies. Multi-tiered object-oriented platforms, such as Java and .NET, have become commonplace. These new tools and technologies are capable of building powerful applications, but they are not easily implemented. Common failures in enterprise applications often occur because their developers do not understand the architectural lessons that experienced object developers have learned. Patterns of Enterprise Application Architecture is written in direct response to the stiff challenges that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology--from Smalltalk to CORBA to Java to .NET--the same basic design ideas can be adapted and applied to solve common problems. With the help of an expert group of contributors, Martin distills over forty recurring solutions into patterns. The result is an indispensable handbook of solutions that are applicable to any enterprise application platform. This book is actually two books in one. The first section is a short tutorial on developing enterprise applications, which you can read from start to finish to understand the scope of the book's lessons. The next section, the bulk of the book, is a detailed reference to the patterns themselves. Each pattern provides usage and implementation information, as well as detailed code examples in Java or C#. The entire book is also richly illustrated with UML diagrams to further explain the concepts. Armed with this book, you will have the knowledge necessary to make important architectural decisions about building an enterprise application and the proven patterns for use when building them. The topics covered include

- Dividing an enterprise application into layers*
- The major approaches to organizing business logic*
- An in-depth treatment of mapping between objects and relational databases*
- Using Model-View-Controller to organize a Web presentation*
- Handling concurrency for data that spans multiple transactions*
- Designing distributed object interfaces*

Learn UML, the Unified Modeling Language, to create diagrams describing the various aspects and uses of your application before you start coding, to ensure that you have everything covered. Millions of programmers in all languages have found UML to be an invaluable asset to their craft. More than 50,000 previous readers have learned UML with Sams Teach Yourself UML in 24 Hours. Expert author Joe Schmuller takes you through 24 step-by-step lessons designed to ensure your understanding of UML diagrams and syntax. This updated edition includes the new features of UML 2.0 designed to make UML an even better modeling tool for modern object-oriented and component-based programming. The CD-ROM includes an electronic version of the book, and Poseidon for UML, Community Edition 2.2, a popular UML modeling tool you can use with the lessons in this book to create UML diagrams immediately.

Using UML, Patterns and Java

Practical Object-Oriented Analysis and Design

An Introduction to Object-oriented Analysis and Design and Iterative Development

Pattern Enterpr Applica Arch

A Brief Guide to the Standard Object Modeling Language

APPLYING UML & PATTERNS 3RD EDITION

A Brain-Friendly Guide

Applying UML and Patterns An Introduction to Object-oriented Analysis and Design and Iterative Development Pearson

Object-oriented analysis and design (OOAD) has over the years, become a vast field, encompassing such diverse topics as design process and principles, documentation tools, refactoring, and design and architectural patterns. For most students the learning experience is incomplete without implementation. This new textbook provides a comprehensive introduction to OOAD. The salient points of its coverage are: • A sound footing on object-oriented concepts such as classes, objects, interfaces, inheritance, polymorphism, dynamic linking, etc. • A good introduction to the stage of requirements analysis. • Use of UML to document user requirements and design. • An extensive treatment of the design process. • Coverage of implementation issues. • Appropriate use of design and architectural patterns. • Introduction to the art and craft of refactoring. • Pointers to resources that further the reader's knowledge. All the main case-studies used for this book have been implemented by the authors using Java. The text is liberally peppered with snippets of code, which are short and fairly self-explanatory and easy to read. Familiarity with a Java-like syntax and a broad understanding of the structure of Java would be helpful in using the book to its full potential.

This is a new edition of this pack which covers the three leading object modelling notations, Coad, OMT and the new Unified (Booch-Rumbaugh) methodology. It presents 177 state-of-the-art strategies and 31 patterns for object model development. The new edition includes 29 new strategies which include: using feature milestones to deliver results more quickly; extracting useful content from data models; using patterns to discover new features, separating definition from usage; when to use, or not use, inheritance; how to decide whether you need an attribute or something more; and why you should nearly always ask for more than a data value.

Object-Oriented Design with UML and Java provides an integrated introduction to object-oriented design with the Unified Modelling Language (UML) and the Java programming language. The book demonstrates how Java applications, no matter how small, can benefit from some design during their construction. Fully road-tested by students on the authors'

own courses, the book shows how these complementary technologies can be used effectively to create quality software. It requires no prior knowledge of object orientation, though readers must have some experience of Java or other high level programming language. This book covers object technology; object-oriented analysis and design; and implementation of objects with Java. It includes two case studies dealing with library applications. The UML has been incorporated into a graphical design tool called ROME, which can be downloaded from the book's website. This object modelling environment allows readers to prepare and edit various UML diagrams. ROME can be used alongside a Java compiler to generate Java code from a UML class diagram then compile and run the resulting application for hands-on learning. This text would be a valuable resource for undergraduate students taking courses on O-O analysis and design, O-O modelling, Java programming, and modelling with UML. * Integrates design and implementation, using Java and UML * Includes case studies and exercises * Bridges the gap between programming texts and high level analysis books on design

This textbook mainly addresses beginners and readers with a basic knowledge of object-oriented programming languages like Java or C#, but with little or no modeling or software engineering experience - thus reflecting the majority of students in introductory courses at universities. Using UML, it introduces basic modeling concepts in a highly precise manner, while refraining from the interpretation of rare special cases. After a brief explanation of why modeling is an indispensable part of software development, the authors introduce the individual diagram types of UML (the class and object diagram, the sequence diagram, the state machine diagram, the activity diagram, and the use case diagram), as well as their interrelationships, in a step-by-step manner. The topics covered include not only the syntax and the semantics of the individual language elements, but also pragmatic aspects, i.e., how to use them wisely at various stages in the software development process. To this end, the work is complemented with examples that were carefully selected for their educational and illustrative value. Overall, the book provides a solid foundation and deeper understanding of the most important object-oriented modeling concepts and their application in software

development. An additional website offers a complete set of slides to aid in teaching the contents of the book, exercises and further e-learning material.

Object Models

Fowler

Design Patterns Explained

Pattern Hatching

Learning UML 2.0

Core J2EE Patterns

Applying UML and Patterns

Explains how to leverage Java's architecture and mechanisms to design enterprise applications and considers code modularity, nonduplication, network efficiency, maintainability, and reusability.

The objective of the workshops associated with the ER'99 18th International Conference on Conceptual Modeling is to give participants access to high level presentations on specialized, hot, or emerging scientific topics. Three themes have been selected in this respect: — Evolution and Change in Data Management (ECDM'99) dealing with handling the evolution of data and data structure, — Reverse Engineering in Information Systems (REIS'99) aimed at exploring the issues raised by legacy systems, — The World Wide Web and Conceptual Modeling (WWWCM'99) which analyzes the mutual contribution of WWW resources and techniques with conceptual modeling. ER'99 has been organized so that there is no overlap between conference sessions and the workshops. Therefore participants can follow both the conference and the workshop presentations they are interested in. I would like to thank the ER'99 program co-chairs, Jacky Akoka and Mokrane Bouzeghoub for having given me the opportunity to organize these workshops. I would also like to thank Stephen Liddle for his valuable help in managing the evaluation procedure for submitted papers and helping to prepare the workshop proceedings for publication. August 1999 Jacques Kouloumdjian Preface for ECDM'99 The first part of this volume contains the proceedings of the First International Workshop on Evolution and Change in Data Management, ECDM'99, which was held in conjunction with the 18th International Conference on Conceptual Modeling (ER'99) in Paris, France, November 15-18, 1999.

The biggest challenge facing many game programmers is completing their game. Most game projects fizzle out, overwhelmed by the complexity of their own code. Game Programming Patterns tackles that exact problem.

Based on years of experience in shipped AAA titles, this book collects proven patterns to untangle and

optimize your game, organized as independent recipes so you can pick just the patterns you need. You will learn how to write a robust game loop, how to organize your entities using components, and take advantage of the CPUs cache to improve your performance. You'll dive deep into how scripting engines encode behavior, how quadrees and other spatial partitions optimize your engine, and how other classic design patterns can be used in games.

Uses friendly, easy-to-understand For Dummies style to help readers learn to model systems with the latest version of UML, the modeling language used by companies throughout the world to develop blueprints for complex computer systems. Guides programmers, architects, and business analysts through applying UML to design large, complex enterprise applications that enable scalability, security, and robust execution. Illustrates concepts with mini-cases from different business domains and provides practical advice and examples. Covers critical topics for users of UML, including object modeling, case modeling, advanced dynamic and functional modeling, and component and deployment modeling.

* Allen Holub is a highly regarded instructor for the University of California, Berkeley, Extension. He has taught since 1982 on various topics, including Object-Oriented Analysis and Design, Java, C++, C. Holub will use this book in his Berkeley Extension classes. * Holub is a regular presenter at the Software Development conferences and is Contributing Editor for the online magazine JavaWorld, for whom he writes the Java Toolbox. He also wrote the OO Design Process column for IBM DeveloperWorks. * This book is not time-sensitive. It is an extremely well-thought out approach to learning design patterns, with Java as the example platform, but the concepts presented are not limited to just Java programmers. This is a complement to the Addison-Wesley seminal "Design Patterns" book by the "Gang of Four".

UML @ Classroom

Holub on Patterns

Systems Analysis and Design in a Changing World

Object-Oriented Analysis and Design

Pearson New International Edition

A Simple Process for Specifying Component-based Software

Design patterns, which express relationships between recurring problems and proven solutions, have become immensely popular in the world of software development. More and more software developers are recognizing the supreme usefulness of design patterns and how

they ease the design and delivery of software applications. This book builds upon the information presented in the seminal work in this field, *Design Patterns: Elements of Reusable Object-Oriented Software*, and gives software professionals the information they need to recognize and write their own patterns. *Pattern Hatching*, written by one of the co-authors of *Design Patterns*, truly helps the software professional apply one of the most popular concepts in software development.

John Hunt's book guides you through the use of the UML and the Unified Process and their application to Java systems. Key topics focus explicitly on applying the notation and the method to Java. The book is clearly structured and written, making it ideal for practitioners. This second edition is considerably revised and extended and includes examples taken from the latest version of Rational Rose and Together. Considers how Agile Modelling fits with the Unified Process, and presents *Design Patterns Self contained* – covers both the Unified Process and UML in one book Includes real-world case studies Written by an experienced author and industry expert Ideal for students on Software Engineering courses

Enterprise Patterns and MDA teaches you how to customize any archetype pattern—such as Customer, Product, and Order—to reflect the idiosyncrasies of your own business environment. Because all the patterns work harmoniously together and have clearly documented relationships to each other, you ' ll come away with a host of reusable solutions to common problems in business-software design. This book shows you how using a pattern or a fragment of a pattern can save you months of work and help you avoid costly errors. You ' ll also discover how—when used in literate modeling—patterns can solve the difficult challenge of communicating UML models to broad audiences. The configurable patterns can be used manually to create executable code. However, the authors draw on their extensive experience to show you how to tap the significant power of MDA and UML for maximum automation. Not surprisingly, the patterns included in this book are highly valuable; a blue-chip company recently valued a similar, but less mature, set of patterns at hundreds of thousands of dollars. Use this practical guide to increase the efficiency of your designs and to create robust business applications that can be applied immediately in a business setting.

Cay Horstmann offers readers an effective means for mastering computing concepts and developing strong design skills. This book introduces object-oriented fundamentals critical to designing software and shows how to implement design techniques. The author's clear, hands-on presentation and outstanding writing style help readers to better understand the material.· *A Crash Course in Java· The Object-Oriented Design Process· Guidelines for Class Design· Interface Types and Polymorphism· Patterns and GUI Programming· Inheritance and Abstract Classes· The Java Object Model· Frameworks· Multithreading· More Design Patterns*

Refined and streamlined, *SYSTEMS ANALYSIS AND DESIGN IN A CHANGING WORLD, 7E* helps students develop the conceptual, technical, and managerial foundations for systems analysis design and implementation as well as project management principles for systems development. Using case driven techniques, the succinct 14-chapter text focuses on content that is key for success in today's market. The authors' highly effective presentation teaches both traditional (structured) and object-oriented (OO) approaches to systems analysis and design. The book highlights use cases, use diagrams, and use case descriptions required for a modeling approach, while demonstrating their application to traditional, web development, object-oriented, and service-oriented architecture approaches. The Seventh Edition's refined sequence of topics makes it easier to read and understand than ever. Regrouped analysis and design chapters provide more flexibility in course organization. Additionally, the text's running cases have been completely updated and now include a stronger focus on connectivity in applications. Important Notice: Media content referenced within the product description or the product text may not be available in the

ebook version.

Best Practices and Design Strategies

Strategies, Patterns, and Applications

Enterprise Model Patterns

Applied Java Patterns

Applied Akka Patterns

UML, Use Cases, Patterns, and Software Architectures

A Hands-On Guide to Designing Distributed Applications

Presents a step-by-step process to master object-oriented analysis and design, from requirements gathering all the way to code generation, using the latest version of the industry standard modeling language. Original. (Advanced)

Sun Microsystems experts Stelting and Maassen describe how design patterns can be applied effectively to the Java platform and present proven techniques for all types of patterns, from system architecture to single classes. Applied Java Patterns features a pattern catalog organized into four major categories - the creational, structural, behavioral, and system patterns. In addition, the authors identify patterns in the core Java APIs and present techniques for pattern use in distributed development.

Here you'll find one key to the development of a successful information system: Clearly capture and communicate both the abstract and concrete building blocks of data that describe your organization. In 1995, David Hay published Data Model Patterns: Conventions of Thought - the groundbreaking book on how to use standard data models to describe the standard business situations. Enterprise Model Patterns: Describing the World builds on the concepts presented there, adds 15 years of practical experience, and presents a more comprehensive view. You will learn how to apply both the abstract and concrete elements of your enterprise's architectural data model through four levels of abstraction: Level 0: An abstract template that underlies the Level 1 model that follows, plus two meta models: • Information Resources. In addition to books, articles, and e-mail notes, it also includes photographs, videos, and sound recordings. • Accounting. Accounting is remarkable because it is itself a modeling language. It takes a very different approach than data modelers in that instead of using entities and entity classes that represent things in the world, it is concerned with accounts that represent bits of value to the organization. Level 1: An enterprise model that is generic enough to apply to any company or government agency, but concrete enough to be readily understood by all. It describes: • People and Organization. Who is involved with the business? The people involved are not only the employees within the organization, but customers, agents, and others with whom the organization comes in contact. Organizations of interest include the enterprise itself and its own internal departments, as well as customers, competitors, government agencies, and the like. • Geographic Locations. Where is business conducted? A geographic location may be either a geographic area (defined as any bounded area on the Earth), a geographic point (used to identify a particular location), or, if you are an oil company for example, a geographic solid (such as an oil reserve). • Assets. What

tangible items are used to carry out the business? These are any physical things that are manipulated, sometimes as products, but also as the means to producing products and services. • Activities. How is the business carried out? This model not only covers services offered, but also projects and any other kinds of activities. In addition, the model describes the events that cause activities to happen. • Time. All data is positioned in time, but some more than others. Level 2: A more detailed model describing specific functional areas: • Facilities • Human Resources • Communications and Marketing • Contracts • Manufacturing • The Laboratory Level 3: Examples of the details a model can have to address what is truly unique in a particular industry. Here you see how to address the unique bits in areas as diverse as: • Criminal Justice. The model presented here is based on the “Global Justice XML Data Model” (GJXDM). • Microbiology • Banking. The model presented here is the result of working for four different banks and then adding some thought to come up with something different from what is currently in any of them. • Highways. The model here is derived from a project in a Canadian Provincial Highway Department, and addresses the question “what is a road?”

More than 300,000 developers have benefited from past editions of UML Distilled . This third edition is the best resource for quick, no-nonsense insights into understanding and using UML 2.0 and prior versions of the UML. Some readers will want to quickly get up to speed with the UML 2.0 and learn the essentials of the UML. Others will use this book as a handy, quick reference to the most common parts of the UML. The author delivers on both of these promises in a short, concise, and focused presentation. This book describes all the major UML diagram types, what they're used for, and the basic notation involved in creating and deciphering them. These diagrams include class, sequence, object, package, deployment, use case, state machine, activity, communication, composite structure, component, interaction overview, and timing diagrams. The examples are clear and the explanations cut to the fundamental design logic. Includes a quick reference to the most useful parts of the UML notation and a useful summary of diagram types that were added to the UML 2.0. If you are like most developers, you don't have time to keep up with all the new innovations in software engineering. This new edition of Fowler's classic work gets you acquainted with some of the best thinking about efficient object-oriented software design using the UML--in a convenient format that will be essential to anyone who designs software professionally.

With its clear introduction to the Unified Modeling Language (UML) 2.0, this tutorial offers a solid understanding of each topic, covering foundational concepts of object-orientation and an introduction to each of the UML diagram types.

Agile Modeling with UML

UML 2 For Dummies

Software Architecture in Practice

Head First Design Patterns

Software Engineering and Computer Systems, Part III

Second International Conference, ICSECS 2011, Kuantan, Pahang, Malaysia, June 27-29, 2011, Proceedings

Design Patterns

The UML was conceived and first implemented as a language for describing the design of object-oriented programs. Its widespread adoption and inherent flexibility has, inevitably, led to its use in other areas, including the design of component-based systems. While it is not a perfect fit for component-based development, this book describes how best to use UML 1.3 in the specification and design of medium to large systems that utilize server-side component technologies.

This book covers all you need to know to model and design software applications from use cases to software architectures in UML and shows how to apply the COMET UML-based modeling and design method to real-world problems. The author describes architectural patterns for various architectures, such as broker, discovery, and transaction patterns for service-oriented architectures, and addresses software quality attributes including maintainability, modifiability, testability, traceability, scalability, reusability, performance, availability, and security. Complete case studies illustrate design issues for different software architectures: a banking system for client/server architecture, an online shopping system for service-oriented architecture, an emergency monitoring system for component-based software architecture, and an automated guided vehicle for real-time software architecture. Organized as an introduction followed by several short, self-contained chapters, the book is perfect for senior undergraduate or graduate courses in software engineering and design, and for experienced software engineers wanting a quick reference at each stage of the analysis, design, and development of large-scale software systems.

When it comes to big data processing, we can no longer ignore concurrency or try to add it in after the fact. Fortunately, the solution is not a new paradigm of development, but rather an old one. With this hands-on guide, Java and Scala developers will learn how to embrace concurrent and distributed applications with the open source Akka toolkit. You'll learn how to put the actor model and its associated patterns to immediate and practical use. Throughout the book, you'll deal with an analogous workforce problem: how to schedule a group of people across a variety of projects while optimizing their time and skillsets. This example will help you understand how Akka uses actors, streams, and other tools to stitch your application together. Model software that reflects the real world with domain-driven design Learn principles and practices for implementing individual actors Unlock the real potential of Akka with patterns for combining multiple actors Understand the consistency tradeoffs in a distributed system Use several Akka methods for isolating and dealing with failures Explore ways to build systems that support availability and scalability Tune your Akka application for performance with JVM tools and dispatchers

This Three-Volume-Set constitutes the refereed proceedings of the Second International Conference on Software Engineering and Computer Systems, ICSECS 2011, held in Kuantan, Malaysia, in June 2011. The 190 revised full papers presented together with invited papers in the three volumes were carefully reviewed and selected from numerous submissions. The papers are organized in topical sections on software engineering; network; bioinformatics and e-health; biometrics technologies; Web engineering; neural network; parallel and distributed; e-learning; ontology; image processing; information and data management; engineering; software security; graphics and multimedia; databases; algorithms; signal processing; software design/testing; e- technology; ad hoc networks; social networks; software process modeling; miscellaneous topics in software engineering and computer systems.

This book focuses on the methodological treatment of UML/P and addresses three core topics of model-based software development: code generation, the systematic testing of programs using a model-based definition of test cases, and the evolutionary refactoring and transformation of models. For each of these topics, it first details the foundational concepts and techniques, and then presents their application with UML/P. This separation between basic principles and applications makes the content more accessible and allows the reader to transfer this knowledge directly to other model-based approaches and languages. After an introduction to the book and its primary goals in Chapter 1, Chapter 2 outlines an agile UML-based approach using UML/P as the primary development language for creating executable models, generating code from the models, designing test cases, and planning iterative evolution through refactoring. In the

interest of completeness, Chapter 3 provides a brief summary of UML/P, which is used throughout the book. Next, Chapters 4 and 5 discuss core techniques for code generation, addressing the architecture of a code generator and methods for controlling it, as well as the suitability of UML/P notations for test or product code. Chapters 6 and 7 then discuss general concepts for testing software as well as the special features which arise due to the use of UML/P. Chapter 8 details test patterns to show how to use UML/P diagrams to define test cases and emphasizes in particular the use of functional tests for distributed and concurrent software systems. In closing, Chapters 9 and 10 examine techniques for transforming models and code and thus provide a solid foundation for refactoring as a type of transformation that preserves semantics. Overall, this book will be of great benefit for practical software development, for academic training in the field of Software Engineering, and for research in the area of model-based software development. Practitioners will learn how to use modern model-based techniques to improve the production of code and thus significantly increase quality. Students will find both important scientific basics as well as direct applications of the techniques presented. And last but not least, the book will offer scientists a comprehensive overview of the current state of development in the three core topics it covers.

A UML Pattern Language

Game Programming Patterns

UML 2.0 in a Nutshell

Learning Design Patterns by Looking at Code

Enterprise Cloud Computing

Code Generation, Testing, Refactoring

ER'99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling Paris, France, November 15-18, 1999 Proceedings

Cloud computing promises to revolutionize IT and business by making computing available as a utility over the internet. This book is intended for practising software architects who need to assess the impact of such a transformation. It explains the evolution of the internet into a computing platform, describes emerging development paradigms and technologies, and discusses how these will change the way enterprises should be architected for cloud deployment. Gautam Shroff provides a technical description of cloud computing technologies, covering cloud infrastructure and platform services, programming paradigms such as MapReduce, as well as 'do-it-yourself' hosted development tools. The book describes emerging technologies critical to cloud computing. The book also covers the fundamentals of enterprise computing, including an introduction to enterprise architecture, so it will interest programmers aspiring to become software architects and serve as a reference level course in software architecture or software engineering.

Object-Oriented Design And Patterns

Agile and Iterative Development

UML Components

UML 2 and the Unified Process

A New Perspective on Object-Oriented Design

UML Distilled

Technology, Architecture, Applications