Component Software Beyond Object Oriented Programming 2nd Edition

A catalog of solutions to commonly occurring design problems, presenting 23 patterns that allow designers to create flexible and reusable designs for object-oriented software. Describes the circumstances in which each pattern is applicable, and discusses the consequences and trade-offs of using the pattern within a larger design. Patterns are compiled from real systems, and include code for implementation in object-oriented programming languages like C++ and Smalltalk. Includes a bibliography. Annotation copyright by Book News, Inc., Portland, OR

Component SoftwareBeyond Object-oriented ProgrammingAddison-Wesley Professional

The book provides a clear understanding of what software reuse is, where the problems are, what benefits to expect, the activities, and its different forms. The reader is also given an overview of what sofware components are, different kinds of components and compositions, a taxonomy thereof, and examples of successful component reuse. An introduction to software engineering and software process models is also provided.

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 quadtrees and other spatial partitions optimize your engine, and how other classic design patterns can be used in games.

Here's a complete guide to building reliable component-based software systems. Written by world-renowned experts in the component-based software engineering field, this unique resource helps you manage complex software through the development, evaluation and integration of software components. You quickly develop a keen awareness of the benefits and risks to be considered when developing reliable systems using components. A strong software engineering perspective helps you gain a better understanding of software component design, to build systems with stronger requirements, and avoid typical errors throughout the process, leading to improved quality and time to market. From component definition, standards, objects and frameworks, to organizational development and support of the component-based life cycle, the book describes aspects of systems development using components and component development. It focuses on dependable and real-time systems, employing case studies from the process automation industry, software production, electronic consumer

equipment and office software development.

In this book, Peter Herzum and Oliver Sims present a complete component based strategy, the business component approach, that applies and extends component thinking to all aspects of the software life cycle for enterprise systems. The approach includes a conceptual framework that brings components into the world of scalable systems, and outlines the different component granularities. It also includes a methodology that goes beyond current object-oriented practices to provide the concepts required to meet the real challenges of component-based development. Using their business component approach, the authors then provide a blueprint for a business component factory--a development capability that can produce software with the quality, speed, and flexibility needed to match changing business needs. Sprinkled with guidelines, tips, and architectural patterns, this book fully prepares you for the approaching component revolution. Praise for Business Component Factory ". . . this book should be very useful for anyone considering the daunting task of adopting component software on an enterprise scale."-Clemens Szyperski (Microsoft Research), Author of the award-winning book, Component Software: Beyond Object-Oriented Programming "Herzum and Sims do an admirable job of differentiating the different component concepts, allowing this clearly written book to focus on the construction of business systems by non-software practitioners, out of business component parts developed separately (and perhaps for a commodity component marketplace). This is the future of software systems, and this book is a practical, giant step in that direction."-Richard Mark Soley, PhD,Chairman and CEO, OMG "Finally, a book that takes you from component design all the way down to the middleware on which they are deployed. It?s an important contribution to the nascent server-side component discipline written by practitioners for practitioners."-Robert Orfali, Author of Client/Server Survival Guide, Third Edition and Client/Server Programming with Java and CORBA, Second Edition (both from Wiley)

Index: side 383-411.

This book constitutes the refereed proceedings of the 6th Software Quality Days Conference (SWQD) held in Vienna, Austria, in January 2014. This professional symposium and conference offers a range of comprehensive and valuable opportunities for advanced professional training, new ideas and networking with a series of keynote speeches, professional lectures, exhibits and tutorials. The four scientific full papers accepted for SWQD were each peer reviewed by three or more reviewers and selected out of 24 high-quality submissions. Further, one keynote and ten short papers on promising research directions were also presented and included in order to spark discussions between researchers and practitioners. The papers are organized into topical sections on software process improvement and measurement, requirements management, value-based software engineering, software and systems testing, automation-supported testing and quality assurance and collaboration.

Component Software: Beyond Object-Oriented Programming explains the technical foundations of this evolving technology and its importance in the software market place. It provides in-depth discussion of both the technical and the business issues to be considered, then moves on to suggest approaches for implementing component-oriented software production and the organizational requirements for success. The author draws on his own experience to offer tried-and-tested solutions to common problems and novel approaches to potential pitfalls. Anyone responsible for developing software strategy, evaluating new technologies, buying or building software will find Clemens Szyperskiis objective and market-aware perspective of this new area invaluable. Helpful Features Include: a uniquely objective comparison of the industry front-runnersi products: Sunis Java Beans; Microsoftis DCOM and Active X; the OMGis CORBA and IIOP a description of the emerging industry standards being developed by consortia such as the OMG and the OPEN Group studies of component-oriented tools and languages, using Java and Component Pascal as examples in-depth discussion of the potential and challenges of component software (c) Clemens Szyperski 1998 0201178885B04062001

"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.

Vogel offers the ultimate developer's guide to using and building objects and components effectively and efficiently in Visual Basic. Topics include MTS, ASP, COM, DCOM, COM+, and more. From soup to nuts, Vogel also explores designing compoenents and distributing them with the Package and Deployment Wizard.

Explains how Billy Beene, the general manager of the Oakland Athletics, is using a new kind of thinking to build a successful and winning baseball team without spending enormous sums of money.

Providing all the latest on a topic of extreme commercial relevance, this book contains the refereed proceedings of the 10th International ACM SIGSOFT Symposium on Component-Based Software Engineering, held in Medford, MA, USA in July 2007. The 19 revised full papers presented were carefully reviewed and selected from 89 submissions. The papers feature new trends in global software services and distributed systems architectures to push the limits of established and tested component-based methods, tools and platforms.

Includes more than 30 percent revised material and five new chapters, covering the new 2.1 features such as EJB Timer Service and JMS as well as the latest open source Java solutions The book was developed as part of TheServerSide.com online EJB community, ensuring a built-in audience Demonstrates how to build an EJB system, program with EJB, adopt best practices, and harness advanced EJB concepts and techniques, including transactions, persistence, clustering, integration, and performance optimization Offers practical guidance on when not to use EJB and how to use simpler, less costly open source technologies in place of or in conjunction with EJB

BASIC APPROACH PLEASE PROVIDE COURSE INFORMATION

This book comprises the refereed proceedings of the International Conference, AIM/CCPE 2012, held in Bangalore, India, in April 2012. The papers presented were carefully reviewed and selected from numerous submissions and focus on the various aspects of research and development activities in computer science, information technology, computational engineering, mobile communication, control and instrumentation, communication system, power electronics and power engineering.

The book introduces the reader to computer programming, i.e. algorithms and data structures. It covers many new programming concepts that have emerged in recent years including object-oriented programming and design patterns. The book emphasizes the practical aspects of software construction without neglecting their solid theoretical foundation.

Fra bagsiden: As a platform, Java defines the services needed to connect binary components at runtime safely and reliably. To truly take advantage of alle Java has to offer, you must consider not just development, but also deployment, and not just objects, but also components. The book delves into the component-oriented features of the Java platform, thoroughly discussing class loading, refelction, serialization, native interoperation and code generation.

This new edition continues its unique approach to teaching all aspects of object-oriented programming, bringing it right up to date with the latest advances in

technology. It requires no extensive knowledge of programming languages. It is divided into four parts, each presenting the issues involved in object-oriented programming from a different perspective: software engineering and design, languages and system development, abstract data types and polymorphism, and applications and frameworks. Software engineers who want to understand the theory behind modern object-oriented technology while learning about such new topics as patterns, UML, and Java.

This book constitutes the refereed proceedings of the 11th International Conference on Software Reuse, ICSR 2009, held in Falls Church, VA, USA, in September 2009. The 28 full papers were caryfully selected from numerous submissions. 2009 was the year that ICSR went back to its roots. The theme was Formal Foundations of Reuse and Domain Engineering. The theory and formal foundations that underlie current reuse and domain engineering practice were explored and current advancements to get an idea of where the field of reuse was headed, were looked at. Many of the papers in these proceedings reflect that theme, e.g. component reuse and verification, feature modeling, generators and model-driven development, industry experience, product lines, reuse and patterns, service-oriented environments.

Shows developers how COM operates and how to use it to create efficient and stable programs consistent with the COM philosophy, allowing disparate applications and components to work together across a variety of languages, platforms, and host machines. Original. (Advanced).

This book constitutes the refereed proceedings of the 11th International ACM SIGSOFT Symposium on Component-Based Software Engineering, CBSE 2008, held in Karlsruhe, Germany in October 2008. The 20 revised full papers and 3 short papers presented were carefully reviewed and selected from 70 submissions. The papers feature new trends in global software services and distributed systems architectures to push the limits of established and tested component-based methods, tools and platforms. The papers are organized in topical sections on performance engineering; extra-functional properties: security and energy; formal methods and model checking; verification techniques; run-time infrastructures; methods of design and development; component models.

Object-oriented programming (OOP) has been the leading paradigm for developing software applications for at least 20 years. Many different methodologies, approaches, and techniques have been created for OOP, such as UML, Unified Process, design patterns, and eXtreme Programming. Yet, the actual process of building good software, particularly large, interactive, and long-lived software, is still emerging. Software engineers familiar with the current crop of methodologies are left wondering, how does all of this fit together for designing and building software in real projects? This handbook from one of the world's leading software architects and his team of software engineers presents guidelines on how to develop high-quality software in an application-oriented way. It answers questions such as: * How do we analyze an application domain

utilizing the knowledge and experience of the users? * What is the proper software architecture for large, distributed interactive systems that can utilize UML and design patterns? * Where and how should we utilize the techniques and methods of the Unified Process and eXtreme Programming? This book brings together the best of research, development, and day-to-day project work. "The strength of the book is that it focuses on the transition from design to implementation in addition to its overall vision about software development." -Bent Bruun Kristensen, University of Southern Denmark, Odense

DCOM -- the Distributed Component Object Model -- is a recent upgrade of a time-honored and well-tested technology promoted by Microsoft for distributed object programming. Now that components are playing a larger and larger part in Windows 98, Windows NT 4.0, and Windows 2000, every Windows programmer will want to understand the technology. DCOM competes with CORBA as a rich and robust method for creating expandable and flexible components, allowing you to plug in new parts conveniently and upgrade without the need for code changes to every program that uses your component.This book introduces C++ programmers to DCOM and gives them the basic tools they need to write secure, maintainable programs. While using Visual C++ development tools and wizards where appropriate, the author never leaves the results up to magic. The C++ code used to create distributed components and the communications exchanged between systems and objects are described at a level where the reader understands their significance and can use the insights for such tasks as debugging and improving performance.The first few chapters explain both the remote procedure calls that underlie DCOM's communication and the way DCOM uses C++ classes. Readers become firmly grounded in the relation between components, classes, and objects, the ways objects are created and destroyed, how clients find servers, and the basics of security and threading.After giving you a grounding in how DCOM works, this book introduces you to the Microsoft tools that make it all easy. By showing what really happens each time you choose a button in a wizard, Learning DCOM makes it possible for you to choose what you need.This book is for anyone who wants to understand DCOM. While thoroughly practical in its goals, it doesn't stint on the background you need to make your programs safe, efficient, and easy to maintain.Topics include: MIDL (Microsoft Interface Definition Language, the language for defining COM interfaces) COM error and exception handling Custom, dispatch, and dual interfaces Standard and custom factories Management of in-process versus out-of-process servers Distributed memory management Pragmatic explanation of the DCOM wire protocol Standard, custom, handler, and automation marshaling Multithreading and apartments Security at the system configuration and programming level Active Template Library (ATL), ATL wizards -- and what they don't do Writing a component that can be invoked from Visual Basic Techniques for using distributed components Creating an ActiveX control and embedding it in a Web client Authentication and the use of Windows NT security features Techniques for

merging marshaling code Connection and distributed events management An introduction to COM+ features

* Fun and easy-to-grasp, yet based on solid programming principles of object-oriented programming * Visually oriented—teaches programming by commanding turtle to move through loops, variables, procedures, and AI * Suitable for any reader, from curious children to adults, who'd like a gentle, methodical approach to core programming concepts

Test-Driven Development (TDD) is now an established technique for delivering better software faster. TDD is based on a simple idea: Write tests for your code before you write the code itself. However, this "simple" idea takes skill and judgment to do well. Now there's a practical guide to TDD that takes you beyond the basic concepts. Drawing on a decade of experience building real-world systems, two TDD pioneers show how to let tests guide your development and "grow" software that is coherent, reliable, and maintainable. Steve Freeman and Nat Pryce describe the processes they use, the design principles they strive to achieve, and some of the tools that help them get the job done. Through an extended worked example, you'll learn how TDD works at multiple levels, using tests to drive the features and the object-oriented structure of the code, and using Mock Objects to discover and then describe relationships between objects. Along the way, the book systematically addresses challenges that development teams encounter with TDD—from integrating TDD into your processes to testing your most difficult features. Coverage includes Implementing TDD effectively: getting started, and maintaining your momentum throughout the project Creating cleaner, more expressive, more sustainable code Using tests to stay relentlessly focused on sustaining quality Understanding how TDD, Mock Objects, and Object-Oriented Design come together in the context of a real software development project Using Mock Objects to guide object-oriented designs Succeeding where TDD is difficult: managing complex test data, and testing persistence and concurrency

The book describes a method for developing the testing of components in parallel with their functionality based on models. UML models are used to derive the testing architecture for an application, the testing interfaces and the component testers. The method provides a process and guidelines for modeling and developing these artifacts. The book also discusses the implications of built-in contract testing with other component-based development technologies such as product-line engineering, middleware platforms, reuse principles etc. Still further, it describes a new method for specifying and checking real-time properties of object-oriented, component-based real-time systems that are based on dynamic execution time analysis with optimization algorithms.

Component-Based Software Engineering (CBSE) is the way to produce software fast. This book presents the concepts in CBSE. While detailing both the advantages and the limitations of CBSE, it covers every aspect of component engineering, from software engineering practices to the design of software component infrastructure, technologies, and system.

This edition has been updated to cover contemporary technologies, discussing how they work, the pros and cons of each, standards, and future markets and developments. It uses the main component programming languages Java, Component Pascal and C#

Presenting the state of the art in component-based software testing, this cutting-edge resource offers you an in-depth understanding of the current issues, challenges, needs and solutions in this critical area. The book discusses the very latest advances in component-based testing and quality assurance in an accessible tutorial format, making the material easy to comprehend and benefit from no matter what your professional level. important, and how it differs from traditional software testing. From an introduction to software components, testing component-based software and validation methods for software components, to performance testing and measurement, standards and certification and verification of quality for component-based systems, you get a revealing snapshot of the key developments in this area, including

important research findings. This volume also serves as a textbook for related courses at the advanced undergraduate or graduate level.

This ground-breaking book presents a complete methodology for adaptive programming in any object-oriented programming language. Lieberherr's adaptive method signals a new approach to object-oriented program design that goes beyond object encapsulation and hard-coded navigation paths to achieve more flexible interactions among objects. Programmers using this method work at a higher, schematic level of abstraction; graph notation represents the class structure and a "propagation pattern" language tells how to distribute meaningful methods - including navigation - across the structure. Using this method, programmers can easily adapt and modify programs as they evolve. This book can be used with any object-oriented programming environment, or with the Demeter Tools Version 5.5, a complete, professional software system for creating and maintaining adaptive programs.

The Art of Objects offers an extensive overview of the long-standing principles of object technology, along with leading-edge developments in the field. It will give you a greater understanding of design patterns and the know-how to use them to find effective solutions to a wide range of design challenges. And because the book maintains an approach independent of specific programming languages, the concepts and techniques presented here can be applied to any object-oriented development environment. Using the Unified Modeling Language (UML), The Art of Objects examines numerous static and dynamic practical object design patterns, illustrated by real-life case studies that demonstrate how to put the patterns to work. You will also find discussion of basic concepts of database management and persistent objects, and an introduction to advanced topics in object modeling and interface design patterns. Moving beyond the design level, the book also covers important concepts in object-oriented architecture. Specific topics include: *Object creation and destruction, associations and links, aggregation, inheritance, and other object design fundamentals *UML notation basics for static and dyna

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.

David A. Sykes is a member of Wofford College's faculty.

Component-based software development, CBSD, is no longer just one more new paradigm in software engineering, but is effectively used in development and practice. So far, however, most of the efforts from the software engineering community have concentrated on the functional aspects of CBSD, leaving aside the treatment of the quality issues and extra-functional properties of software components and component-based systems. This book is the first one focusing on quality issues of components and

component-based systems. The 16 revised chapters presented were carefully reviewed and selected for inclusion in the book; together with an introductory survey, they give a coherent and competent survey of the state of the art in the area. The book is organized in topical parts on COTS selection, testing and certification, software component quality models, formal models to quality assessment, and CBSD management.

Describes ways to incorporate domain modeling into software development.

Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows users exactly how to spot the best opportunities for refactoring and exactly how to do it, step by step.

Business Component-Based Software Engineering, an edited volume, aims to complement some other reputable books on CBSE, by stressing how components are built for large-scale applications, within dedicated development processes and for easy and direct combination. This book will emphasize these three facets and will offer a complete overview of some recent progresses. Projects and works explained herein will prompt graduate students, academics, software engineers, project managers and developers to adopt and to apply new component development methods gained from and validated by the authors. The authors of Business Component-Based Software Engineering are academic and professionals, experts in the field, who will introduce the state of the art on CBSE from their shared experience by working on the same projects. Business Component-Based Software Engineering is designed to meet the needs of practitioners and researchers in industry, and graduate-level students in Computer Science and Engineering.

An indispensable resource for anyone working with Eiffel, this up-to-date guide provides full coverage of the most recent version of the language, focusing on Eiffel's practical use in the development of large, mission-critical software systems. In addition to a comprehensive description of Eiffel's syntax and semantics, you will find in-depth information on style guides, analysis and design, design patterns, and validation and testing. Descriptions and comparisons of available compilers and libraries will help you decide which Eiffel tools best fit your development needs. The book even includes an Eiffel resource guide. The book's most notable feature is its three large-scale case studies that demonstrate Eiffel in action, illustrating implementation techniques and showcasing Eiffel's power and effectiveness in three different realms: the MIS world, the embedded systems/telecommunications world, and the numeric world. By reading this book, you will not only obtain a knowledge of the mechanics of Eiffel programming, but you will also come away with an understanding of Eiffel's role in the field of object-oriented technology and a sense of the language's strong potential in large software development. 0201633817B04062001

Component Oriented Programming offers a unique programming-centered approach to component-based software development that delivers the well-developed training and practices you need to successfully apply this cost-effective method. Following an overview of basic theories and methodologies, the authors provide a unified component infrastructure for building component software using JavaBeans, EJB, OSGi, CORBA, CCM, .NET, and Web services. You'll learn how to develop reusable software components; build a software system of pre-built software components; design and

implement a component-based software system using various component-based approaches. Clear organization and self-testing features make Component Oriented Programming an ideal textbook for graduate and undergraduate courses in computer science, software engineering, or information technology as well as a valuable reference for industry professionals.