

Object Oriented Software Engineering

This book describes how object-oriented language and object-oriented ideas can be employed throughout the software project. It describes the software engineering process from requirements analysis up to acceptance testing and contains such topics as unit testing, and system design. The book uses the C++ programming language and is intended for both the undergraduate student and the industrial developer. Material on the relationship between object-oriented techniques and prototyping is also included.

This textbook provides a progressive approach to the teaching of software engineering. First, readers are introduced to the core concepts of the object-oriented methodology, which is used throughout the book to act as the foundation for software engineering and programming practices, and partly for the software engineering process itself. Then, the processes involved in software engineering are explained in more detail, especially methods and their applications in design, implementation, testing, and measurement, as they relate to software engineering projects. At last, readers are given the chance to practice these concepts by applying commonly used skills and tasks to a hands-on project. The impact of such a format is the potential for quicker and deeper understanding. Readers will master concepts and skills at the most basic levels before continuing to expand on and apply these lessons in later chapters.

EBOOK: OBJECT-ORIENTED SOFTWARE

Object-oriented Software Engineering with C++

A Hands-On Approach

Pearson New International Edition

Quantitative Approaches in Object-oriented Software Engineering

Object-oriented Software Engineering with Eiffel

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

This comprehensive and well-written book presents the fundamentals of object-oriented software engineering and discusses the recent technological developments in the field. It focuses on object-oriented software engineering in the context of an overall effort to present object-oriented concepts, techniques and models that can be applied in software estimation, analysis, design, testing and quality improvement. It applies unified modelling language notations to a series of examples with a real-life case study. The example-oriented approach followed in this book will help the readers in understanding and applying the concepts of object-oriented software engineering quickly and easily in various application domains. This book is designed for the undergraduate and postgraduate students of computer science and engineering, computer applications, and information technology. KEY FEATURES : Provides the foundation and important concepts of object-oriented paradigm. Presents traditional and object-oriented software development life cycle models with a special focus on Rational Unified Process model. Addresses important issues of improving software quality and measuring various object-oriented constructs using object-oriented metrics. Presents numerous diagrams to illustrate object-oriented software engineering models and concepts. Includes a large number of solved examples, chapter-end review questions and multiple choice questions along with their answers.

Software -- Software Engineering.

Project-based Software Engineering

Using UML, Patterns, and Java

A Sue Case Driven Approach

A Use Case Driven Approach

Project-Based Software Engineering is the first book to provide hands-on process and practice in software engineering essentials for the beginner. The book presents steps through the software development life cycle and two running case studies that develop as the steps are presented. Running parallel to the process presentation and case studies, the book supports a semester-long software development project. This book focuses on object-oriented software development, and supports the conceptualization, analysis, design and implementation of an object-oriented project. It is mostly language-independent, with necessary code examples in Java. A subset of UML is used, with the notation explained as needed to support the readers' work. Two running case studies a video game and a library check out system show the development of a software project. Both have sample deliverables and thus provide the reader with examples of the type of work readers are to create. This book is appropriate for readers looking to gain experience in project analysis, design implementation, and testing.

Addresses critical software engineering issues, showing how an object - oriented approach can provide much improved solutions over other methods. Designed as a technology tool.

Object-oriented Software EngineeringA Use Case Driven ApproachAddison-Wesley

OBJECT-ORIENTED SOFTWARE ENGINEERING

Classical and Object-oriented Software Engineering with UML and Java

An Implementor's Guide

Object-oriented and Classical Software Engineering

Growing Object-Oriented Software, Guided by Tests

Based on Objectory which is the first commercially available comprehensive object-orientd process for developing large scale industrial systems.

Venturing beyond C++ programming, this text shows how to engineer software products using object-oriented principles. It covers gathering requirements, specifying objects, object verification, defining relations between objects, translating object design into code, object testing, and software maintenance.

The object-oriented methodology OOram is new and different from all others on the market, and has been in use and development in Norway for over 12 years. This book is the authoritative account of the OOram methodology for software analysis, design, development, maintenance, and reuse.

Advances in Object-oriented Software Engineering

Object-Oriented and Classical Software Engineering

Applications and Approaches to Object-Oriented Software Design: Emerging Research and Opportunities

Working with Objects

Problems & Perspectives

A comprehensive description is provided of four areas of quantitative approaches in object-oriented software engineering: metrics collection, quality assessment, metrics validation, and process management.

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).

This is a textbook for a course in object-oriented software engineering at advanced undergraduate and graduate levels, as well as for software engineers. It contains more than 120 exercises of diverse complexity. The book discusses fundamental concepts and terminology on object-oriented software development, assuming little background on software engineering, and emphasizes design and maintenance rather than programming. It also presents up-to-date and easily understood methodologies and puts forward a software life cycle model which explicitly encourages reusability during software development and maintenance.

The Profesional Developer's Guide

Object-oriented Software and Engineering

EBOOK: OBJECT-ORIENTED SOFTWARE

Object-Oriented Software Engineering with UML

Classical and Object-oriented Software Engineering with UML and C++

The object-oriented paradigm supplements traditional software engineering by providing solutions to common problems such as modularity and reusability. Objects can be written for a specific purpose acting as an encapsulated black-box API that can work with other components by forming a complex system. This book provides a comprehensive overview of the many facets of the object-oriented paradigm and how it applies to software engineering. Starting with an in-depth look at objects, the book naturally progresses through the software engineering life cycle and shows how object-oriented concepts enhance each step. Furthermore, it is designed as a roadmap with each chapter, preparing the reader with the skills necessary to advance the project.This book should be used by anyone interested in learning about object-oriented software engineering, including students and seasoned developers. Without overwhelming the reader, this book hopes to provide enough information for the reader to understand the concepts and apply them in their everyday work. After learning about the fundamentals of the object-oriented paradigm and the software engineering life cycle, the reader is introduced to more advanced topics such as web engineering, cloud computing, agile development, and big data. In recent years, these fields have been rapidly growing as many are beginning to realize the benefits of developing on a highly scalable, automated deployment system. Combined with the speed and effectiveness of agile development, legacy systems are beginning to make the transition to a more adaptive environment.Core Features:1. Provides a thorough exploration of the object-oriented paradigm.2. Provides a detailed look at each step of the software engineering life cycle.3. Provides supporting examples and documents.4. Provides a detailed look at emerging technology and standards in object-oriented software engineering.

This book covers the essential knowledge and skills needed by a student who is specializing in software engineering. Readers will learn principles of object orientation, software development, software modeling, software design, requirements analysis, and testing. The use of the Unified Modelling Language to develop software is taught in depth. Many concepts are illustrated using complete examples, with code written in Java.

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 igrow 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

Essays on Object-oriented Software Engineering

Object-Oriented Software Engineering: An Agile Unified Methodology

Object-oriented Software Engineering with UML

The Object-Oriented Way

Elements of Reusable Object-Oriented Software

Object-Oriented Software Engineering: An Agile Unified Methodology by David Kung presents a **step-by-step methodology that integrates modeling and design, UML, patterns, test-driven development, quality assurance, configuration management, and agile principles throughout the life cycle. The overall approach is casual and easy to follow, with many practical examples that show the theory at work. The author uses his experiences as well as real-world stories to help the reader understand software design principles, patterns, and other software engineering concepts. The book also provides stimulating exercises that go far beyond the type of question that can be answered by simply copying portions of the text.**

This text provides an introduction to the process of software engineering. The revision concentrates on updating the book to reflect the most current trends and innovations in the field. The Universal Modeling Language (UML) has become an industry standard and now permeates this new edition. In this text, it is used for object-oriented analysis and design as well as when diagrams depict objects and their interrelationships. Design patterns, frameworks and software architecture have also become a popular topic in the field of software engineering and are part of a new chapter on reuse, portability, and inoperability. The inoperability material includes sections on such hot topics as OLE, COM, and CORBA. Some material from the 3rd edition has been reorganized into a new chapter on planning and estimating, including feature points and COCOMO II. While the text has been updated, the traditional features which have defined the previous three editions of Schach's book have been retained. These include a balanced coverage of the object-oriented model along with the classical model (as reflected in the title) and an emphasis on metrics. The special considerations of object-oriented life-cycle models, object-oriented analysis, and object-oriented design are also retained in this edition.

The authors analyze how the structure of a package determines its developmental complexity according to such measures as bug search times and documentation information content. The work presents arguments for why these issues impact solution cost and time more than does scalable performance. The final chapter explores the question of scalable execution and shows how scalable design relates to scalable execution. The book's focus is on program organization, which has received considerable attention in the broader software engineering community, where graphical description standards for modeling software structure and behavior have been developed by computer scientists. These discussions might be enriched by engineers who write scientific codes. This book aims to bring such scientific programmers into discussion with computer scientists. The authors do so by introducing object-oriented software design patterns in the context of scientific simulation.

Understanding Object-oriented Software Engineering

Object-Oriented Software Engineering: Practical Software Development using UML and Java

Design Patterns

Scientific Software Design

Software Engineering: A Hands-On Approach

Examines object-oriented methods, practices, terminology, and concepts

Object-Oriented Software Engineering is written for both the traditional one-semester and the newer two-semester software engineering curriculum. Part I covers the underlying software engineering theory, while Part II presents the more practical life cycle, workflow by workflow. The text is intended for the substantial object-oriented segment of the software engineering market. It

focuses exclusively on object-oriented approaches to the development of large software systems that are the most widely used. Text includes 2 running case studies, expanded coverage of agile processes and open-source development.

The object-oriented paradigm supplements traditional software engineering by providing solutions to common problems such as modularity and reusability. Objects can be written for a specific purpose acting as an encapsulated black-box API that can work with other components by forming a complex system. This book provides a comprehensive overview of the many facets of the object-oriented paradigm and how it applies to software engineering. Starting with an in-depth look at objects, the book naturally progresses through the software engineering life cycle and shows how object-oriented concepts enhance each step. Furthermore, it is designed as a roadmap with each chapter, preparing the reader with the skills necessary to advance the project.This book should be used by anyone interested in learning about object-oriented software engineering, including students and seasoned developers. Without overwhelming the reader, this book hopes to provide enough information for the reader to understand the concepts and apply them in their everyday work. After learning about the fundamentals of the object-oriented paradigm and the software engineering life cycle, the reader is introduced to more advanced topics such as web engineering, cloud computing, agile development, and big data. In recent years, these fields have been rapidly growing as many are beginning to realize the benefits of developing on a highly scalable, automated deployment system. Combined with the speed and effectiveness of agile development, legacy systems are beginning to make the transition to a more adaptive environment.Core Features: 1. Provides a thorough exploration of the object-oriented paradigm. 2. Provides a detailed look at each step of the software engineering life cycle. 3. Provides supporting examples and documents. 4. Provides a detailed look at emerging technology and standards in object-oriented software engineering.

A Hands-on Approach

Emerging Research and Opportunities

Object Oriented Software Engineering

Conquering Complex and Changing Systems

The OOram Software Engineering Method

In today's modernized environment, a growing number of software companies are changing their traditional engineering approaches in response to the rapid development of computing technologies. As these businesses adopt modern software engineering practices, they face various challenges including the integration of current methodologies and contemporary design models and the refactoring of existing systems using advanced approaches. Applications and Approaches to Object-Oriented Software Design: Emerging Research and Opportunities is a pivotal reference source that provides vital research on the development of modern software practices that impact maintenance, design, and developer productivity. While highlighting topics such as augmented reality, distributed computing, and big data processing, this publication explores the current infrastructure of software systems as well as future advancements. This book is ideally designed for software engineers, IT specialists, data scientists, business professionals, developers, researchers, students, and academicians seeking current research on contemporary software engineering methods.

Classical and Object-Oriented Software Engineering, 5/e is designed for an introductory software engineering course. This book provides an excellent introduction to software engineering fundamentals, covering both traditional and object-oriented techniques.Schach's unique organization and style makes it excellent for use in a classroom setting. It presents the underlying software engineering theory in Part I and follows it up with the more practical life-cycle material in Part II. Many software engineering books are more like reference books, which do not provide the appropriate fundamentals before inundating students with implementation details.In this edition, more practical material has been added to help students understand how to use what they are learning. This has been done through the use of "How To" boxes and greater implementation detail in the case study. Additionally, the new edition contains the references to the most current literature and includes an overview of extreme programming.The website in this edition will be more extensive. It will include Solutions, PowerPoints that incorporate lecture notes, newly developed self-quiz questions, and source code for the term project and case study.

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. Object-Oriented Software Engineering Using UML, Patterns, and Java, 3e, shows readers 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; readers 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).

Classical and Object-oriented Software Engineering

An Object-oriented Approach

Object-Oriented Software Engineering Using UML, Patterns, and Java

Practical Software Development Using UML and Java

Object-Oriented Software: Design and Maintenance

This book is based on object-oriented techniques applied to software engineering. Employing the latest technologies such as UML, Patterns, and Java, Bernd Bruegge and Allen H. Dutoit offer a cohesive, class-tested presentation of object-oriented software engineering in a step-by-step format based on ten years of teaching and real-world software engineering experience. This text teaches practical experience in developing complex software appropriate for software engineering project courses, as well as industry R & D practitioners. The reader benefits from timely exposure to state-of-the-art tools and methods. Unlike other texts based on the teaching premise of multiple classes or developing multiple systems, this book focuses on techniques and applications in a reasonably complex environment, such as multi-team development projects including 20 to 60 participants. The book is based on concrete examples from real applications such as accident management, emissions modeling, facility management, and centralized traffic control. Provides an integrated communication infrastructure for distributed development Shows the state of the art in Software Engineering: UML, Java, Design Patterns, Distributed Development, and Multiproject Management Illustrates how the reader learns to develop in a distributed team with hands-on experience on real system development problems Offers a CD-ROM containing the materials used in courses taught by the authors-problem statements, requirement analysis documents, system design documents, test manuals, prototypes, and all the artifacts produced during the development of a facility management system Presents Companion Website (www.prenhall.com/bruegge) withsupplemental material such as problem statements, requirement analysis documents, system design documents, test manuals, and solutions to exercises

Classical and Object-Oriented Software Engineering is designed for an introductory software engineering course. This book provides an excellent introduction to software engineering fundamentals, covering both traditional and object-oriented techniques. Schach's unique organization and style makes it excellent for use in a classroom setting. It presents the underlying software engineering theory in Part I and follows it up with the more practical life-cycle material in Part II. Many software engineering books are more like reference books, which do not provide the appropriate fundamentals before inundating students with implementation details. In this edition, more practical material has been added to help students understand how to use what they are learning. This has been done through the use of "How To" boxes and greater implementation detail in the case study. Additionally, the new edition contains the references to the most current literature and includes an overview of extreme programming. The website in this edition will be more extensive. It will include Solutions, PowerPoints that incorporate lecture notes, newly developed self-quiz questions, and source code for the term project and case study.

Object-Oriented Software Engineering

Designing Object-oriented Software

Object-Oriented Software Engineering