Reusable Software Components Object Oriented Embedded Systems Programming In C

"Despite continuous improvements in hardware processors, storage, and networks, developing quality software on-time and under budget remains difficult. Moreover, developing high quality, reusable software is even more challenging. The principles, practices, and skills required to develop such software are best learned by attaining mastery of patterns and frameworks. A pattern describes a reusable solution to a common problem that arises within a particular context of software design. When related patterns are woven together they provide a vocabulary and a process for the orderly resolution of software development problems. A framework is an integrated set of software components that collaborate to provide a reusable architecture for a family of related applications. Frameworks can also be viewed as concrete realizations of patterns that facilitate direct reuse of detailed designs and source code. Design Patterns in Java LiveLessons describes how to master the complexity of developing software by learning and applying object-oriented patterns and frameworks. It centers on a case study based on many of the patterns in the book Design Patterns: Elements of Reusable Object-Oriented Software (the so-called 'Gang of Four' book) that showcases pattern- and object-oriented design and programming techniques using Java. This case study will help you evaluate the limitations of alternative software development methods (such as algorithm decomposition) and demonstrate by example how patterns and object-orientation help to alleviate such limitations."--Resource description page.
Software -- Software Engineering.
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. Software Engineering discusses the major issues associated with different phases of software development life cycle. Starting from the basics, the book discusses several advanced topics. Topics like software project management, software process models, developing methodologies, software specification, software testing and quality, software implementation, software security, software maintenance and software reuse are discussed. This book also gives an introduction to the new emerging technologies, trends and practices in software engineering field. New topics such as MIMO technology, AJAX, etc. are included in the book. The topics like .NET framework, J2EE, etc. are also dealt with. Case Studies, discussions on real-life situations of dealing with IT related problems and finding their solutions in an easy manner, are given in each chapter. Elegant and simple style of presentation makes the reading of this book a pleasant experience. Students of Computer Science and Engineering, Information Technology and Computer Applications should find this book highly useful. It would also be useful for IT technology professionals who are interested to get acquainted with the latest and the newest technologies.

A Metrics-based Approach to the Automated Identification of Object Oriented Reusable Software Components

A Simple Process for Specifying Component-based Software

Based on the MET++ Application Framework

Component-Based Software Engineering

Business Component-Based Software Engineering

Design Patterns CD

- First book of its kind (case studies in CBD) - Covers different kinds of components - Covers different component models/technologies - Includes a wide scope of CBD topics - Covers both theoretical and practical work - Includes both formal and informal approaches - Provides a snapshot of current concerns and pointers to future trends

An overview of the basic issues concerning software reuse with focus on mental and supplemental tools that support the concept. Describes the processes including: components, software libraries, methodologies, Ada resuse experiences, and object-oriented computing. Acidic paper; no index. Annotation

Apply design principles to your classes, preparing them for reuse. You will use package design principles to create packages that are just right in terms of cohesion and coupling, and are user- and maintainer-friendly at the same time. The first part of this book walks you through the five SOLID principles that will help you improve the design of your classes. The second part introduces you to the best practices of package design, and covers both package cohesion principles and package coupling principles. Cohesion principles show you which classes should be put together

in a package, when to split packages, and if a combination of classes may be considered a "package" in the first place. Package coupling principles help you choose the right dependencies and prevent wrong directions in the dependency graph of your packages. What You'll LearnApply the SOLID principles of class designDetermine if classes belong in the same packageKnow whether it is safe for packages to depend on each other Who This Book Is For Software developers with a broad range of experience in the field, who are looking for ways to reuse,share, and distribute their code

From the basics to the most advanced quality of service (QoS) concepts, this all encompassing, first-of-its-kind book offers an in-depth understanding of the latest technical issues raised by the emergence of new types, classes and qualities of Internet services. The book provides end-to-end QoS guidance for real time multimedia communications over the Internet. It offers you a multiplicity of hands-on examples and simulation script support, and shows you where and when it is preferable to use these techniques for QoS support in networks and Internet traffic with widely varying characteristics and demand profiles. This practical resource discusses key standards and protocols, including real-time transport, resource reservation, and integrated and differentiated service models, policy based management, and mobile/wireless QoS. The book features numerous examples, simulation results and graphs that illustrate important concepts, and pseudo codes are used to explain algorithms. Case studies, based on freely available Linux/FreeBSD systems, are presented to show you how to build networks supporting Quality of Service. Online support material including presentation foils, lab exercises and additional exercises are available to text adopters.

Programming .NET Components

Reusable Software Components

8th International Conference, ICSR 2004, Madrid, Spain, July 5-9, 2004, Proceedings

LEGOS

SOFTWARE ENGINEERING

(Issues, Tools, Techniques, and Trends)

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.

At the now famous 1968 NATO Conference on Software Engineering, "Massproduced software components" were advocated as an answer to the software crisis. As a result, there has been a significant shift of emphasis from projects to the production of reusable software components. Today, object-oriented programming (OOP) provides the best known basis for reusable software construction. OOP is touted as being the most influential programming development of the 1990's, in the same way that structured programming was in the 1970's. Object-oriented programming is not a fad, but rather a well establish programming paradigm with the potential to significantly improve software quality. [Author's abstract].

Introducing the reuse-driven software engineering business; Architectural style; Processes; Organizing a reuse business.

It is now more than twenty-five years since object-oriented programming was “ inve- ed ” (actually, more than thirty years since work on Simula started), but, by all accounts, it would appear as if object-oriented technology has only been “ discovered ” in the past ten years! When the first European Conference on Object-Oriented Programming was held in Paris in 1987, I think it was generally assumed that Object-Oriented Progr-

ming, like Structured Programming, would quickly enter the vernacular, and that a c- ference on the subject would rapidly become superfluous. On the contrary, the range and impact of object-oriented approaches and methods continues to expand, and, - spite the inevitable oversell and hype, object-oriented technology has reached a level of scientific maturity that few could have foreseen ten years ago. Object-oriented technology also cuts across scientific cultural boundaries like p- haps no other field of computer science, as object-oriented concepts can be applied to virtually all the other areas and affect virtually all aspects of the software life cycle. (So, in retrospect, emphasizing just Programming in the name of the conference was perhaps somewhat short-sighted, but at least the acronym is pronounceable and easy to rem- ber!) This year ' s ECOOP attracted 146 submissions from around the world - making the selection process even tougher than usual. The selected papers range in topic from programming language and database issues to analysis and design and reuse, and from experience reports to theoretical contributions.

The Design and Implementation of a Reusable Component Library and a Retrieval/Integration System

Object-oriented Design Methodology to Facilitate Reuse

Design and Build .NET Applications Using Component-Oriented Programming

Object-oriented Embedded Systems Programming in C

Case Studies

Development of Application Software Hierarchy for Reuse (DASH'R)

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.

The goal of this Technology Reinvestment Program Focus Area is to radically reduce the effort required to field new software applications through the development of reusable software components. Today an estimated 85% of the installed base is a custom application, with all components written especially for that software package. "Object Oriented Software", an emerging software technology, which is becoming widely used in the development of new software, offers the promise of reusability and ease of modification for both Defense and commercial applications. However, the promise can only be realized if the use of object oriented software is created according to an established set of standards and if appropriate reusable software components are developed. Building on emerging industry standards for software object technologies, these projects will significantly accelerate development of tools to help build the infrastructure for component ware, create a pool of developers experienced with applying the new tools,

and deliver a series of demonstration applications with both commercial and defense relevance.

McClure takes software reuse beyond "good intentions", by presenting specific reuse techniques that have repeatedly helped companies lower costs and improve quality.

This thesis demonstrates and illustrates a way of developing reusable graphics software components in Ada associated with a C++/C library. The work was carried out using object-oriented software development techniques that were used to analyze, design and implement a partial flight simulator. The objective of this thesis was to present a way of building reusable software components with Ada in a graphics application environment. An object-oriented approach was taken in the development of a set of reusable graphics software components for a flight simulator domain. A selection of a set of reusable software components came from domain analysis. These components were analyzed in detail, then redesigned to demonstrate and illustrate the thesis objective. Examples from design and implementation demonstrate how Ada 83 was applied in building reusable graphics software components associated with C++ routines, the limitations of Ada 83, and how Ada9X addresses these limitations.

Architecture Process and Organization for Business Success

A Dissertation

Elements of Reusable Object-Oriented Software

Software Maintenance - A Management Perspective

Design Patterns in Java LiveLessons

Component Software: Beyond Object-Oriented Programming, 2/E

This book on the MET++ multimedia application framework provides an in-depth look at the concepts and techniques applied in an object-oriented class library to support multimedia application development. It is a reference for software designers and programmers who want to build multimedia applications by reusing components of the MET++ framework.

Today's increasingly competitive and fiscally constrained business environment is fostering the need to cut costs and justify expenditures. Usability engineering is not yet universally accepted, nor is it yet an integrated aspect of software engineering, and would-be usability champions need more help than ever to win the funding necessary to introduce and promote usability engineering techniques. Cost-Justifying Usability is the first book to address pragmatically and in detail the question of how usability engineering professionals and their managers can cost-justify their proposals and efforts. The book offers specific techniques for quantifying costs and benefits, making a convincing and successful business case for investment in usability engineering.This book comprises a thorough and well-integrated collection of chapters written by experienced and prominent usability experts. Taken together, these chapters

provide readers with:An overall framework for cost-justifying usability engineering programs that can be applied to any contextAn examination of the unique factors and issues in cost-justifying usability efforts for three very different types of organizations: vendor companies, international development organizations, and contractor companiesCase studies of successful cost-justification effortsA look at some special issues regarding cost-justification of usability, including"discount"usability engineering techniques, success factors for introducing usability engineering into development organizations, specialized tools for usability cost-justification, and a look to the future of usability engineeringPractical and effective insight for human factors professionals, interface designers, software development managers, and

Rapid prototyping with automated retrieval of reusable software components is a software development method to construct software systems expeditiously. This thesis describes a tool to enhance the practice of software reuse within the Computer Aided Prototyping System (CAPS). A software base interface provides prototype designers with the means to retrieve components and integrate them into new applications. Reusable components are retrieved from the software base using a formal specification as the search key or through a browser. The specification language used is the Prototype System Description Language (PSDL). The software base stores the reusable components in an object oriented database management system (ONTOS) with an appropriate PSDL specification. Following a query conducted by the PSDL specification, chosen retrieved components are transformed and integrated to the system under development. All software base procedures, including the storage, retrieval, and integration of the components, are conducted through a graphical user interface which is designed to demonstrate and manipulate available software base operations.

Techniques and principles; Presentation of the libraries; Class reference.

Object-oriented Implementations of Data Structures and Algorithms as Reusable Software Components

Creating Reusable Software Components

Elements of Reusable Object-oriented Software

Software Reuse: Methods, Techniques, and Tools

Component-Oriented Programming

Toward Reusable Graphics Components in Ada

Object orientation has become a “must know” subject for managers, researchers, and software practitioners interested in the design, evolution, reuse and

management of efficient software components.The book contains technical papers reflecting both theoretical and practical contributions from researchers in the field of object-oriented (OO) databases and software engineering systems. The book identifies actual and potential areas of integration of OO and database technologies, current and future research directions in software methodologies, and reflections about the OO paradigm.In providing current research and relevant information about this promising and rapidly growing field of object-oriented databases and software engineering systems, this book is invaluable to research scientists, practitioners, and graduate students working in the areas of databases and software engineering.

An estimated 85% of the installed base of software is a custom application with a production quantity of one. In practice, almost 100% of military software systems are custom software. Paradoxically, the marginal costs of producing additional units are near zero. So why hasn't the software market, a market with high design costs and low productions costs evolved like other similar custom widget industries, such as automobiles and hardware chips? The military software industry seems immune to market pressures that have motivated a multilevel supply chain structure in other widget industries: design cost recovery, improve quality through specialization, and enable rapid assembly from purchased components. The primary goal of the ComponentWare Consortium (CWC) technology plan was to overcome barriers to building and deploying mission-critical information systems by using verified, reusable software components (Component Ware). The adoption of the ComponentWare infrastructure is predicated upon a critical mass of the leading platform vendors' inevitable adoption of adopting emerging, object-based, distributed computing frameworks--initially CORBA and COM/OLE. The long-range goal of this work is to build and deploy military systems from verified reusable architectures. The promise of component-based applications is to enable developers to snap together new applications by mixing and matching prefabricated software components. A key result of this effort is the concept of reusable software architectures. A second important contribution is the notion that a software architecture is something that can be captured in a formal language and reused across multiple applications. The formalization and reuse of software architectures provide major cost and schedule improvements. The Unified Modeling Language (UML) is fast becoming the industry standard for object-oriented analysis and design notation for object-based systems. However, the lack of a standard real-time distributed object operating system, lack of a standard Computer-Aided Software Environment (CASE) tool notation and lack of a standard CASE tool repository has limited the realization of component software. The approach to fulfilling this need is the software component factory innovation. The factory approach takes advantage of emerging standards such as UML, CORBA, Java and the Internet. The key technical innovation of the software component factory is the ability to assemble and test new system configurations as well as assemble new tools on demand from existing tools and architecture design repositories.

Computational Intelligence Techniques and Their Applications to Software Engineering Problems focuses on computational intelligence approaches as applicable in varied areas of software engineering such as software requirement prioritization, cost estimation, reliability assessment, defect prediction, maintainability and quality prediction, size estimation, vulnerability prediction, test case selection and prioritization, and much more. The concepts of expert systems, case-based reasoning, fuzzy logic, genetic algorithms, swarm computing, and rough sets are introduced with their applications in software engineering. The field of knowledge discovery is explored using neural networks and data mining techniques by determining the underlying and hidden patterns in software data sets. Aimed at graduate students and researchers in computer science engineering, software engineering, information technology, this book: Covers various aspects of in-depth solutions of software engineering problems using computational intelligence techniques Discusses the latest evolutionary approaches to preliminary theory of different solve optimization problems under software engineering domain Covers heuristic as well as meta-heuristic algorithms designed to provide better and optimized solutions Illustrates applications including software requirement prioritization, software cost estimation, reliability assessment, software defect prediction, and more Highlights swarm intelligence-based optimization solutions for software testing and reliability problems

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.

Principles of Package Design

Design Patterns: Elements of Reusable Object-Oriented Software

Component-based Software Development

Guidelines and Methods

The Base Object-oriented Component Libraries

Reusable Software

This book constitutes the refereed proceedings of the 8th International Conference on Software Reuse, ICSR-8, held in Madrid, Spain in July 2004. The 28 revised full papers presented were carefully reviewed and selected from numerous submissions. The papers are organized in topical sections on software variability: requirements; testing reusable software; feature modeling; aspect-oriented software development; component and service development; code level reuse; libraries, classification, and retrieval; model-based approaches; transformation and generation; and requirements.

'Programming .NET Components', second edition, updated to cover .NET 2.0., introduces the Microsoft .NET Framework for building components on Windows platforms. From its many lessons, tips, and guidelines, readers will learn how to use the .NET Framework to program reusable, maintainable, and robust components.

Computer systems play an important role in our society. Software drives those systems. Massive investments of time and resources are made in developing and implementing these systems. Maintenance is inevitable. It is hard and costly. Considerable resources are required to keep the systems active and dependable. We cannot maintain software unless maintainability characters are built into the products and processes. There is an urgent need to reinforce software development practices based on quality and reliability principles. Though maintenance is a mini development lifecycle, it has its own problems. Maintenance issues need corresponding tools and techniques to address them. Software professionals are key players in maintenance. While development is an art and science, maintenance is a craft. We need to develop maintenance personnel to master this craft. Technology impact is very high in systems world today. We can no longer conduct business in the way we did before. That calls for reengineering systems and

software. Even reengineered software needs maintenance, soon after its implementation. We have to take business knowledge, procedures, and data into the newly reengineered world. Software maintenance people can play an important role in this migration process. Software technology is moving into global and distributed networking environments. Client/server systems and object-orientation are on their way. Massively parallel processing systems and networking resources are changing database services into corporate data warehouses. Software engineering environments, rapid application development tools are changing the way we used to develop and maintain software. Software maintenance is moving from code maintenance to design maintenance, even onto specification maintenance. Modifications today are made at specification level, regenating the software components, testing and integrating them with the system. Eventually software maintenance has to manage the evolution and evolutionary characteristics of software systems. Software professionals have to maintain not only the software, but the momentum of change in systems and software. In this study, we observe various issues, tools and techniques, and the emerging trends in software technology with particular reference to maintenance. We are not searching for specific solutions. We are identifying issues and finding ways to manage them, live with them, and control their negative impact.

Software Reuse is a state of the art book concerning all aspects of software reuse. It does away with the hype and shows the reality. Different techniques are presented which enable software reuse and the author demonstrates why object-oriented methods are better for reuse than other approaches. The book details the different factors to take into account when managing reusable components: characterisation, identification, building, verification, storage, search, adaptation, maintenance and evolution. Comparisons and description of various types of companies that could benefit from applying reuse techniques are included outlining, amongst other things, increased profitability and likely problems that might arise from the purchase and selling of reuse tools and components. Based on a real experience of software reuse in a company with a bibliography of more than 200 references provided, this book is a 'must have' for all those working in the software reuse field.

Tutorial, Software Reuse

Design Patterns

10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007, Proceedings

7th European Conference, Kaiserslautern, Germany, July 26-30, 1993. Proceedings

Reusable Software Component Retrieval Via Normalized Algebraic Specifications
UML Components

Helps real-time embedded systems designers combine the development benefits of the widely-used C language and object-oriented techniques not normally associated with C.Introduces object-oriented programming to microcontroller programmers familiar with C. Shows how objects can be written in C, and developed into classes. Presents useful objects and classes for microcontroller programs, including a class that creates instances of an asynchronous serial port. Shows how to implement components to handle timer functions and input capture. Compiles data sheets for all components derived in the book.Programmers working with real-time embedded systems.

Capturing a wealth of experience about the design of object-oriented software, four top-notch designers present a catalog of simple and succinct solutions to commonly occurring design problems. Previously undocumented, these 23 patterns allow designers to create more flexible, elegant, and ultimately reusable designs without having to rediscover the design solutions themselves.

The 23 patterns contained in the book, Design Patterns: Elements of Reusable Object-Oriented Software have become an essential resource for anyone developing reusable software designs. Now these design patterns, along with the entire text of the book, are being made available on CD. This electronic version will enable programmers to install the patterns directly onto a computer or network and create an architecture for using and building reusable components. Produced in HTML format, the CD is heavily cross-referenced with numerous links to the online text.

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.

Computational Intelligence Techniques and Their Applications to Software Engineering Problems
ECOOP '93 - Object-Oriented Programming
Software Reuse
Emerging Technology
Testing and Quality Assurance for Component-based Software
Object-oriented Technology For Database And Software Systems