# Doing Hard Time Developing Real Time Systems With Uml Objects Frameworks And Patterns With Cd Rom

Thoroughly updated and fully compliant with Rational Rose 2002, the latest release of the industry's most popular software modeling tool, this edition contains simplified, useful case studies and helps the reader understand the core concepts of modeling and how to use UML effectively.

Overviews the process of building and compiling executable UML models for software development. The book focuses on the BridgePoint tool suite and object action language developed by Project Technology. The authors discuss identifying system requirements, diagramming classes and attributes, constraints on the class diagram, ways of building sets of communicating statechart diagrams, and model verification. Annotation copyrighted by Book News, Inc., Portland, OR.

* Full analysis of performance characteristics of the .NET Framework, including actual benchmark results * Information on the internals of the .NET Framework and exposure to the various elements that make up the .NET Framework * Description of tools and techniques for identifying performance problems developers may encounter * References to sources of further information on various performance topics * Written by a Microsoft MVP in a technically unique style and of the highest quality

Contributions on UML address the application of UML in the specification of embedded HW/SW systems. C-Based System Design embraces the modeling of operating systems, modeling with different models of computation, generation of test patterns, and experiences from case studies with SystemC. Analog and Mixed-Signal Systems covers rules for solving general modeling problems in VHDL-AMS, modeling of multi-nature systems, synthesis, and modeling of Mixed-Signal Systems with SystemC. Languages for formal methods are addressed by contributions on formal specification and refinement of hybrid, embedded and real-time stems. Together with articles on new languages such as SystemVerilog and Software Engineering in Automotive Systems the contributions selected for this book embrace all aspects of languages and models for specification, design, modeling and verification of systems. Therefore, the book gives an excellent overview of the actual state-of-the-art and the latest research results.

I highly recommend this book for anyone who's ever tried to implement RUP on a small project. Pollice and company have demystified and effectively scaled the process while ensuring that its essence hasn't been compromised. A must-have for any RUPster's library! Chris Soskin, Process Engineering Consultant, Toyota Motor SalesDo you want to improve the process on your next project? Perhaps you'd like to combine the best practices from the Rational Unified Process (RUP) and from agile methodologies (such as Extreme Programming). If so, buy this book! Software Development for Small Teams describes an entire software development project, from the initial customer contact through delivery of the software. Through a case study, it describes how one small, distributed team designed and applied a successful process. But this is not a perfect case study. The story

includes what worked and what didn't, and describes how the team might change its process for the next project. The authors encourage you to assess their results and to use the lessons learned on your next project. Key topics covered include: Achieving a balance between people, process, and tools; recognizing that software develo

Artificial Intelligence (AI) is a scientific field of longstanding tradition, with origins in the early years of computer science. Today AI has reached a level of maturity that allows us to build highly sophisticated systems which perform very different tasks. Nevertheless, its evolution has opened up a number of new problems, ranging from specific algorithms to system integration, which remain elusive and assure a long life for this research field. Research progress in this area is today an international challenge that must be supported by world-class meetings and organizations, but in spite of this fact, there is also an objective need for meetings and organizations that support and disseminate research at other levels. This book focuses on new and original research on Artificial Intelligence.

A recent survey stated that 52% of embedded projects are late by 4-5 months. This book can help get those projects in on-time with design patterns. The author carefully takes into account the special concerns found in designing and developing embedded applications specifically concurrency, communication, speed, and memory usage. Patterns are given in UML (Unified Modeling Language) with examples including ANSI C for direct and practical application to C code. A basic C knowledge is a prerequisite for the book while UML notation and terminology is included. General C programming books do not include discussion of the contraints found within embedded system design. The practical examples give the reader an understanding of the use of UML and OO (Object Oriented) designs in a resource-limited environment. Also included are two chapters on state machines. The beauty of this book is that it can help you today. . Design Patterns within these pages are immediately applicable to your project Addresses embedded system design concerns such as concurrency, communication, and memory usage Examples contain ANSI C for ease of use with C programming code

This book constitutes the refereed proceedings of the International Conference on Embedded and Ubiquitous Computing, EUC 2004, held in Aizu-Wakamatsu City, Japan, in August 2004. The 104 revised full papers presented were carefully reviewed and selected from more than 260 submissions. The papers are organized in topical sections on embedded hardware and software; real-time systems; power-aware computing; hardware/software codesign and systems-on-chip; mobile computing; wireless communication; multimedia and pervasive computing; agent technology and distributed computing, network protocols, security, and fault-tolerance; and middleware and peer-to-peer computing.

The aim of this book is to provide new ideas, original results and practical experiences regarding service robotics. This book provides only a small example of this research activity, but it covers a great deal of what has been done in the field recently. Furthermore, it works as a valuable resource for researchers interested in this field.

The Eclipse environment solves the problem of having to maintain your own Integrated Development Environment (IDE), which is time consuming and costly. Embedded tools can also be easily integrated into Eclipse. The C/C++CDT is ideal

for the embedded community with more than 70% of embedded developers using this language to write embedded code. Eclipse simplifies embedded system development and then eases its integration into larger platforms and frameworks. In this book, Doug Abbott examines Eclipse, an IDE, which can be vital in saving money and time in the design and development of an embedded system. Eclipse was created by IBM in 2001 and then became an open-source project in 2004. Since then it has become the de-facto IDE for embedded developers. Virtually all of the major Linux vendors have adopted this platform, including MontVista, LynuxWorks, and Wind River. *Details the Eclipse Integrated Development Environment (IDE) essential to streamlining your embedded development process *Overview of the latest C/C++ Developer's Toolkit (CDT) *Includes case studies of Eclipse use including Monta Vista, LynuxWorks, and Wind River

This volume contains papers from the IFAC Workshop on Real-Time Programming. The aim of the Workshop was to bring together academic practitioners and industrialists involved in this important and expanding area of interest in order to exchange experiences on recent advances in this field. Contents include: * DEPENDABILITY AND SAFETY FOR REAL TIME SYSTEMS * REAL-TIME PROGRAMMING TECHNIQUES * SOFTWARE REQUIREMENT ENGINEERING * CONTROL SYSTEMS DESIGN * SOFTWARE DESIGN * SOFTWARE ENGINEERING AND COMPLEX ENGINEERINGSYSTEMS

Computers as Components: Principles of Embedded Computing System Design, Third Edition, presents essential knowledge on embedded systems technology and techniques. Updated for today's embedded systems design methods, this volume features new examples including digital signal processing, multimedia, and cyber-physical systems. It also covers the latest processors from Texas Instruments, ARM, and Microchip Technology plus software, operating systems, networks, consumer devices, and more. Like the previous editions, this textbook uses real processors to demonstrate both technology and techniques; shows readers how to apply principles to actual design practice; stresses necessary fundamentals that can be applied to evolving technologies; and helps readers gain facility to design large, complex embedded systems. Updates in this edition include: description of cyber-physical systems; exploration of the PIC and TI OMAP processors; high-level representations of systems using signal flow graphs; enhanced material on interprocess communication and buffering in operating systems; and design examples that include an audio player, digital camera, and cell phone. The author maintains a robust ancillary site at http://www.marilynwolf.us/CaC3e/index.html which includes a variety of support materials for instructors and students, including PowerPoint slides for each chapter; lab assignments developed for multiple systems including the ARM-based BeagleBoard computer; downloadable exercises solutions and source code; and links to resources and additional information on hardware, software, systems, and more. This book will appeal to students in an embedded systems design course as well as to researchers and savvy

professionals schooled in hardware or software design. Description of cyber-physical systems: physical systems with integrated computation to give new capabilities Exploration of the PIC and TI OMAP multiprocessors High-level representations of systems using signal flow graphs Enhanced material on interprocess communication and buffering in operating systems Design examples include an audio player, digital camera, cell phone, and more

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.

This book constitutes the thoroughly refereed post-proceedings of the Second International Workshop on Rapid Integration of Software Engineering Techniques, RISE 2005. The book presents 19 revised full papers together with the abstract of a keynote paper. Among the topics addressed are modelling safety case evolution, practical approaches in model mapping, context-aware service composition, techniques for representing product line core assets for automation, formal development of reactive fault-tolerant systems, and more.

"Highlights of this book include: the MDA framework, including the Platform Independent Model (PIM) and Platform Special Model (PSM); OMG standards and the use of UML; MDA and Agile, Extreme Programming, and Rational Unified Process (RUP) development; how to apply MDA, including PIM-to-PSM and PSM-to-code transformations for Relational, Enterprise JavaBean (EJB), and Web models; transformations, including controlling and tuning, traceability, incremental consistency, and their implications; metamodeling; and relationships between different standards, including Meta Object Facility (MOF), UML, and Object Constraint Language (OCL)."--Jacket.

Real-Time Simulation Technologies: Principles, Methodologies, and Applications is an edited compilation of work that explores fundamental concepts and basic techniques of real-time simulation for complex and diverse systems across a broad spectrum. Useful for both new entrants and experienced experts in the field, this book integrates coverage of detailed theory, acclaimed methodological approaches, entrenched technologies, and high-value applications of real-time simulation—all from the unique perspectives of renowned international contributors. Because it offers an accurate and otherwise unattainable assessment of how a system will behave over a particular time frame, real-time simulation is increasingly critical to the optimization of dynamic processes and adaptive systems in a variety of enterprises. These range in scope from the maintenance of the national power grid, to space exploration, to the development of virtual reality programs and cyber-physical systems. This book outlines how, for these and other undertakings, engineers must assimilate real-time data with computational tools for rapid decision making under uncertainty. Clarifying the central concepts behind real-time simulation tools and techniques, this one-of-a-kind resource: Discusses the state of the art,

important challenges, and high-impact developments in simulation technologies Provides a basis for the study of real-time simulation as a fundamental and foundational technology Helps readers develop and refine principles that are applicable across a wide variety of application domains As science moves toward more advanced technologies, unconventional design approaches, and unproven regions of the design space, simulation tools are increasingly critical to successful design and operation of technical systems in a growing number of application domains. This must-have resource presents detailed coverage of real-time simulation for system design, parallel and distributed simulations, industry tools, and a large set of applications.

Abstraction is the most basic principle of software engineering. Abstractions are provided by models. Modeling and model transformation constitute the core of model-driven development. Models can be refined and finally be transformed into a technical implementation, i.e., a software system. The aim of this book is to give an overview of the state of the art in model-driven software development. Achievements are considered from a conceptual point of view in the first part, while the second part describes technical advances and infrastructures. Finally, the third part summarizes experiences gained in actual projects employing model-driven development. Beydeda, Book and Gruhn put together the results from leading researchers in this area, both from industry and academia. The result is a collection of papers which gives both researchers and graduate students a comprehensive overview of current research issues and industrial forefront practice, as promoted by OMG’s MDA initiative.

This revised and enlarged edition of a classic in Old Testament scholarship reflects the most up-to-date research on the prophetic books and offers substantially expanded discussions of important new insight on Isaiah and the other prophets.

This book documents the satellite events run around the 14th European Conference on Object-Oriented Programming, ECOOP 2000 in Cannes and Sophia Antipolis in June 2000. The book presents 18 high-quality value-adding workshop reports, one panel transcription, and 15 posters. All in all, the book offers a comprehensive and thought-provoking snapshot of the current research in object-orientation. The wealth of information provided spans the whole range of object technology, ranging from theoretical and foundational issues to applications in various domains.

Written as a workbook with a set of guided exercises that teach by example, this book gives a practical, hands-on guide to using UML to design and implement embedded and real-time systems. A review of the basics of UML and the Harmony process for embedded software development: two on-going case examples to teach the concepts, a small-scale traffic light control system and a large scale unmanned air vehicle show the applications of UML to the specification, analysis and design of embedded and real-time systems in general. A building block approach: a series of progressive worked exercises with step-by-step explanations of the complete solution, clearly demonstrating how to convert concepts into actual designs. A walk through of the phases of an incremental spiral process: posing the problems and the solutions for requirements analysis, object analysis, architectural design, mechanistic design, and detailed design.

MDA Distilled is an accessible introduction to the MDA standard and its tools and technologies. The book describes the fundamental features of MDA, how they fit together, and how you can use them in your organization today. You will also learn how to define a model-driven process for a project involving multiple platforms, implement that process, and then test the resulting system.

In this project aims to calculate the proportional difference in the development of skills among students using the Computer Assisted Teaching (CAT) and those without. To this end, we propose the hypothesis that the proportional difference in the development of skills among students using the CAT and those without, to study the subject Operating Systems is 30%. This will define the basic research project as a Quasi-Experimental design and correlational form, where they took 2 samples of 89 students, forming groups: CATG, which used computer-assisted instruction, and not used, NCATG. These groups was administered as a questionnaire and obtained partial notes on the subject. To obtain the results, we evaluate the hypothesis and compared the groups formed in the development of skills and academic performance.

This book constitutes the thoroughly refereed post-proceedings of the 9th International Conference on Real-Time and Embedded Systems and Applications, RTCSA 2003, held in Tainan, Taiwan, in February 2003. The 28 revised full papers and 9 revised short papers presented were carefully reviewed and selected for inclusion in the book. The papers are organized in topical sections on scheduling, networking and communication, embedded systems and environments, pervasive and ubiquitous computing, systems and architectures, resource management, file systems and databases, performance analysis, and tools and development.

This book constitutes thoroughly revised and selected papers from the 7th International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2019, held in Prague, Czech Republic, in February 2019. The 16 thoroughly revised and extended papers presented in this volume were carefully reviewed and selected from 76 submissions. They address some of the most relevant challenges being faced by researchers and practitioners in the field of model-driven engineering and software development and cover topics like language design and tooling; programming support tools; code and text generation from models, behavior modeling and analysis; model transformations and multi-view modeling; as well as applications of MDD and its related techniques to cyber-physical systems, cyber security, IoT, autonomous vehicles and healthcare.

bull; Learn to better leverage the siginificant power of UML 2.0 and the Model-Driven Architecture standard bull; The OCL helps developers produce better software by adding vital definition to their designs bull; Updated to reflect the latest version of the standard - OCL 2.0

Covers UML 2.0.

The complexity of most real-time and embedded systems often exceeds that of other types of systems since, in addition to the usual spectrum of problems inherent in software, they need to deal with the complexities of the physical world. That world—as the proverbial Mr. Murphy tells us—is an unpredictable and often unfriendly place. Consequently, there is a

very strong motivation to investigate and apply advanced design methods and technologies that could simplify and improve the reliability of real-time software design and implementation. As a result, from the first versions of UML issued in the mid 1990's, designers of embedded and real-time systems have taken to UML with vigour and enthusiasm. However, the dream of a complete, model-driven design flow from specification through automated, optimised code generation, has been difficult to realise without some key improvements in UML semantics and syntax, specifically targeted to the real-time systems problem. With the enhancements in UML that have been proposed and are near standardisation with UML 2. 0, many of these improvements have been made. In the Spring of 2003, adoption of a formalised UML 2. 0 specification by the members of the Object Management Group (OMG) seems very close. It is therefore very appropriate to review the status of UML as a set of notations for embedded real-time systems - both the state of the art and best practices achieved up to this time with UML of previous generations - and where the changes embodied in the 2.

Learn how to apply agile methods in the development of real-time and embedded systems * * Introduces the Harmony Process, the first agile method that fully reflects the unique challenges of real-time/embedded systems. * Learn to continuously validate analysis and design models and optimize processes throughout project execution. * Apply MDA in an agile fashion * By Bruce Powel Douglass, a renowned expert on improving real-time and embedded systems development. Real-time and embedded systems face the same challenges as traditional software development: shrinking budgets and shorter timeframes. However, these systems can be even more difficult to develop successfully, due to their additional requirements for timeliness, minimal resource usage, safety, and high reliability - and in some cases, their requirements to support rigorous industry standards. In Real-Time Agility, leading embedded systems consultant Bruce Powel Douglass reveals how to leverage the best practices of agile development to address all of these challenges. Douglass introduces the Harmony Process, a proven, start-to-finish approach to software development that can reduce costs, save time -- and most importantly, eliminate potential defects. Replete with examples, this book serves as an ideal tutorial in agile methods for real-time/embedded systems developers. It has been designed to serve equally well as a reference guide that professionals can rely on while they're 'in the heat of battle,' working to move a project forward to a successful conclusion.

This book constitutes the refereed proceedings of the 12th SIGSAND/PLAIS EuroSymposium 2019 held in Gdansk, Poland, on September 19, 2019. The objective of the EuroSymposium on Systems Analysis and Design is to promote and develop high quality research on all issues related to information systems (IS) and in particular in systems analysis and design (SAND). The 12 papers presented in this volume were carefully reviewed and selected from 32 submissions.

They were organized in topical sections named: information systems in business; health informatics and life-long-learning; IT security; agile methods and software engineering.

"Designing a large software system is an extremely complicated undertaking that requires juggling differing perspectives and differing goals, and evaluating differing options. Applied Software Architecture is the best book yet that gives guidance as to how to sort out and organize the conflicting pressures and produce a successful design." -- Len Bass, author of Software Architecture in Practice. Quality software architecture design has always been important, but in today's fast-paced, rapidly changing, and complex development environment, it is essential. A solid, well-thought-out design helps to manage complexity, to resolve trade-offs among conflicting requirements, and, in general, to bring quality software to market in a more timely fashion. Applied Software Architecture provides practical guidelines and techniques for producing quality software designs. It gives an overview of software architecture basics and a detailed guide to architecture design tasks, focusing on four fundamental views of architecture--conceptual, module, execution, and code. Through four real-life case studies, this book reveals the insights and best practices of the most skilled software architects in designing software architecture. These case studies, written with the masters who created them, demonstrate how the book's concepts and techniques are embodied in state-of-the-art architecture design. You will learn how to: create designs flexible enough to incorporate tomorrow's technology; use architecture as the basis for meeting performance, modifiability, reliability, and safety requirements; determine priorities among conflicting requirements and arrive at a successful solution; and use software architecture to help integrate system components. Anyone involved in software architecture will find this book a valuable compendium of best practices and an insightful look at the critical role of architecture in software development. 0201325713B07092001

Apress????????

A classic treatise that defined the field of applied demand analysis, Consumer Demand in the United States: Prices, Income, and Consumption Behavior is now fully updated and expanded for a new generation. Consumption expenditures by households in the United States account for about 70% of Americaâ__s GDP. The primary focus in this book is on how households adjust these expenditures in response to changes in price and income. Econometric estimates of price and income elasticities are obtained for an exhaustive array of goods and services using data from surveys conducted by the Bureau of Labor Statistics, providing a better understanding of consumer demand. Practical models for forecasting future price and income elasticities are also demonstrated. Fully revised with over a dozen new chapters and appendices, the book revisits the original Taylor-Houthakker models while examining new material as well, such as the use of quantile regression and the stationarity of consumer preference. It also explores the emerging connection between

neuroscience and consumer behavior, integrating the economic literature on demand theory with psychology literature. The most comprehensive treatment of the topic to date, this volume will be an essential resource for any researcher, student or professional economist working on consumer behavior or demand theory, as well as investors and policymakers concerned with the impact of economic fluctuations.

Reliable Software Technologies is an annual series of international conferences devoted to the promotion and advancement of all aspects of reliable software technologies. The objective of this series of conferences, initiated and sponsored by Ada-Europe, the European federation of national Ada societies, is to provide a forum to promote the development of reliable softwares both as an industrial technique and an academic discipline. Previous editions of the Reliable Software Technologies conference were held in: Porto (Portugal) in 2006, York (UK) in 2005, Palma de Mallorca (Spain) in 2004,Toulouse (France) in 2003,Vienna (Austria) in 2002,Leuven (Belgium) in 2001,Potsdam(Germany)in2000,Santander(Spain)in1999,Uppsala(Sweden) in 1998, London (UK) in 1997 and Montreux (Switzerland) in 1996. The 12th International Conference on Reliable Software Technologies took place in Geneva, Switzerland, June 25-29, 2007, under the continued sponsoring ofAda-Europe,incooperationwithACMSIGAda.Itwasorganizedbymembers of the University of Applied Sciences, Western Switzerland (Engineering School of Geneva), in collaboration with colleagues from various places in Europe. The 13th conference, in 2008, will take place in Venice, Italy.

Doing Hard Time is written to facilitate the daunting process of developing real-time systems. It presents an embedded systems programming methodology that has been proven successful in practice. The process outlined in this book allows application developers to apply practical techniques - garnered from the mainstream areas of object-oriented software development - to meet the demanding qualifications of real-time programming. Bruce Douglass offers ideas that are up-to-date with the latest concepts and trends in programming. By using the industry standard Unified Modeling Language (UML), as well as the best practices from object technology, he guides you through the intricacies and specifics of real-time systems development. Important topics such as schedulability, behavioral patterns, and real-time frameworks are demystified, empowering you to become a more effective real-time programmer.

Doing Hard TimeDeveloping Real-time Systems with UML, Objects, Frameworks, and PatternsAddison-Wesley Professional

Design and Analysis of Distributed Embedded Systems is organized similar to the conference. Chapters 1 and 2 deal with specification methods and their analysis while Chapter 6 concentrates on timing and performance analysis. Chapter 3 describes approaches to system verification at different levels of abstraction. Chapter 4 deals with fault tolerance and detection. Middleware and software reuse aspects are treated in Chapter 5. Chapters 7 and 8 concentrate on the distribution related topics such as partitioning, scheduling and communication. The book closes with a chapter on design methods and frameworks.

In this fourth book in the CHDL Series, a selection of the best papers presented in FDL'02 is published. System Specification and Design Languages contains outstanding research contributions in the four areas mentioned above. So, The Analog and Mixed-

Signal system design contributions cover the new methodological approaches like AMS behavioral specification, mixed-signal modeling and simulation, AMS reuse and MEMs design using the new modeling languages such as VHDL-AMS, Verilog-AMS, Modelica and analog-mixed signal extensions to SystemC. UML is the de-facto standard for SW development covering the early development stages of requirement analysis and system specification. The UML-based system specification and design contributions address latest results on hot-topic areas such as system profiling, performance analysis and UML application to complex, HW/SW embedded systems and SoC design.C/C++-for HW/SW systems design is entering standard industrial design flows. Selected papers cover system modeling, system verification and SW generation. The papers from the Specification Formalisms for Proven design workshop present formal methods for system modeling and design, semantic integrity and formal languages such as ALPHA, HANDLE and B.

More than ever, mission-critical and business-critical applications depend on object-oriented (OO) software. Testing techniques tailored to the unique challenges of OO technology are necessary to achieve high reliability and quality. "Testing Object-Oriented Systems: Models, Patterns, and Tools" is an authoritative guide to designing and automating test suites for OO applications. This comprehensive book explains why testing must be model-based and provides in-depth coverage of techniques to develop testable models from state machines, combinational logic, and the Unified Modeling Language (UML). It introduces the test design pattern and presents 37 patterns that explain how to design responsibility-based test suites, how to tailor integration and regression testing for OO code, how to test reusable components and frameworks, and how to develop highly effective test suites from use cases. Effective testing must be automated and must leverage object technology. The author describes how to design and code specification-based assertions to offset testability losses due to inheritance and polymorphism. Fifteen micro-patterns present oracle strategies--practical solutions for one of the hardest problems in test design. Seventeen design patterns explain how to automate your test suites with a coherent OO test harness framework. The author provides thorough coverage of testing issues such as: The bug hazards of OO programming and differences from testing procedural code How to design responsibility-based tests for classes, clusters, and subsystems using class invariants, interface data flow models, hierarchic state machines, class associations, and scenario analysis How to support reuse by effective testing of abstract classes, generic classes, components, and frameworks How to choose an integration strategy that supports iterative and incremental development How to achieve comprehensive system testing with testable use cases How to choose a regression test approach How to develop expected test results and evaluate the post-test state of an object How to automate testing with assertions, OO test drivers, stubs, and test frameworks Real-world experience, world-class best practices, and the latest research in object-oriented testing are included. Practical examples illustrate test design and test automation for Ada 95, C++, Eiffel, Java, Objective-C, and Smalltalk. The UML is used throughout, but the test design patterns apply to systems developed with any OO language or methodology. 0201809389B04062001

As the application of object technology--particularly the Java programming language--has become commonplace, a new problem

has emerged to confront the software development community. Significant numbers of poorly designed programs have been created by less-experienced developers, resulting in applications that are inefficient and hard to maintain and extend. Increasingly, software system professionals are discovering just how difficult it is to work with these inherited, "non-optimal" applications. For several years, expert-level object programmers have employed a growing collection of techniques to improve the structural integrity and performance of such existing software programs. Referred to as "refactoring," these practices have remained in the domain of experts because no attempt has been made to transcribe the lore into a form that all developers could use. . .until now. In Refactoring: Improving the Design of Existing Code, renowned object technology mentor Martin Fowler breaks new ground, demystifying these master practices and demonstrating how software practitioners can realize the significant benefits of this new process. With proper training a skilled system designer can take a bad design and rework it into well-designed, robust code. In this book, Martin Fowler shows you where opportunities for refactoring typically can be found, and how to go about reworking a bad design into a good one. Each refactoring step is simple--seemingly too simple to be worth doing. Refactoring may involve moving a field from one class to another, or pulling some code out of a method to turn it into its own method, or even pushing some code up or down a hierarchy. While these individual steps may seem elementary, the cumulative effect of such small changes can radically improve the design. Refactoring is a proven way to prevent software decay. In addition to discussing the various techniques of refactoring, the author provides a detailed catalog of more than seventy proven refactorings with helpful pointers that teach you when to apply them; step-by-step instructions for applying each refactoring; and an example illustrating how the refactoring works. The illustrative examples are written in Java, but the ideas are applicable to any object-oriented programming language.

A guide to using UML describes major UML diagrams, their creation, and how to decipher them.