Maintenance And Refactoring Of Information Systems

On behalf of the PROFES organizing committee we are proud to present to you the proceedings of the 5th International Conference on Product Focused Software Process Improvement (PROFES 2004), held in Kansai Science City, Japan. Since 1999, PROFES has established itself as one of the recognized international process improvement conferences. In 2004 the conference left Europe for the first time and moved to Japan. Japan and its neighboring countries are intensifying their efforts to improve software engineering excellence, so it was a logical step to select Japan as the venue for PROFES 2004. The purpose of the conference is to bring to light the most recent findings and results in the area and to stimulate discussion between researchers, experienced professionals, and technology providers. The large number of participants coming from industry confirms that the conference provides a variety of up-to-date topics and tackles industry problems. The main theme of PROFES is professional software process improvement (SPI) motivated by product and service quality needs. SPI is facilitated by software process assessment, software measurement, process modeling, and technology transfer. It has become a practical tool for quality software engineering and management. The conference addresses both the solutions found in practice and the relevant research results from academia. This is reflected in the 41 full papers, which are a balanced mix of academic papers as well as industrial experience reports.
Summary The Mikado Method is a book written by the creators of this process. It describes a pragmatic, straightforward, and empirical method to plan and perform non-trivial technical improvements on an existing software system. The method has simple rules, but the applicability is vast. As you read, you'll practice a step-by-step system for identifying the scope and nature of your technical debt, mapping the key dependencies, and determining the safest way to approach the "Mikado"—your goal. About the Technology The game "pick-up sticks" is a good metaphor for the Mikado Method. You eliminate "technical debt" —the legacy problems embedded in nearly every software system— by following a set of easy-to-implement rules. You carefully extract each intertwined dependency until you expose the central issue, without collapsing the project. About the Book The Mikado Method presents a pragmatic process to plan and perform nontrivial technical improvements on an existing software system. The book helps you practice a step-by-step system for identifying the scope and nature of your technical debt, mapping the key dependencies, and determining a safe way to approach the "Mikado"—your goal. A natural by-product of this process is the Mikado Graph, a roadmap that reflects deep understanding of how your system works. This book builds on agile processes such as refactoring, TDD, and rapid feedback. It requires no special hardware or software and can be practiced by both small and large teams. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. What's Inside Understand your technical debt Surface the dependencies in legacy systems Isolate and resolve core concerns while creating minimal disruption Create a roadmap for your changes About the Authors Ola Ellnestam and Daniel Brolund are developers, coaches, and team leaders. They developed the Mikado Method in response to years of experience resolving technical debt in complex legacy systems. Table of Contents PART 1 THE BASICS OF THE MIKADO METHOD Meet the Mikado Method Hello, Mikado Method! Goals, graphs, and guidelines Organizing your work PART 2 PRINCIPLES AND PATTERNS FOR IMPROVING SOFTWARE Breaking up a monolith Emergent design Common restructuring patterns
Provides information on managing and modifying code using refractoring tools and features.
On behalf of the PROFES Organizing Committee, we are proud to present to you the proceedings of the 9th International Conference on Product-Focused Software Process Improvement (PROFES 2008) held in Frascati - Monteporzio Catone, Rome, Italy. Since 1999, PROFES has established itself as one of the recognized international process improvement conferences. The main theme of PROFES is professional so- ware process improvement (SPI) motivated by product and service quality needs. Focussing on a product to be developed, PROFES 2008 addressed both quality en- neering and management topics including processes, methods, techniques, tools, - ganizations, and enabling SPI. Both solutions found in practice and the relevant research results from academia were presented. Domains such as the automotive and mobile applications industry are growing r- idly, resulting in a strong need for professional development and improvement. Nowadays, the majority of embedded software is developed in collaboration, and distribution of embedded software development continues to increase. Thus, PROFES 2008 addressed different development modes, roles in the value chain, stakeholders' viewpoints, collaborative development, as well as economic and quality aspects. - ile development was included again as one of the themes. Since the beginning of the series of PROFES conferences, the purpose has been to bring to light the most recent findings and novel results in the area of process - provement, and to stimulate discussion among researchers, experienced professionals, and technology providers from around the world.
Security on the Web
Computational Excellence and Society 5.0
Becoming a Better Programmer
A Practitioner's Approach
Information and Communication Technology for Intelligent Systems
Principles, Heuristics and Best Practices
A Tertiary Systematic Literature Review

Provides students and engineers with the fundamental developments and common practices of software evolution and maintenance Software Evolution and Maintenance: A Practitioner's Approach introduces readers to a set of well-rounded educational materials, covering the fundamental developments in software evolution and common maintenance practices in the industry. Each chapter gives a clear understanding of a particular topic in software evolution, and discusses the main ideas with detailed examples. The authors first explain the basic concepts and then drill deeper into the important aspects of software evolution. While designed as a text in an undergraduate course in software evolution and maintenance, the book is also a great resource forsoftware engineers, information technology professionals, and graduate students in software engineering. Based on the IEEE SWEBOK (Software Engineering Body of Knowledge) Explains two maintenance standards: IEEE/EIA 1219 and ISO/IEC14764 Discusses several commercial reverse and domain engineering toolkits Slides for instructors are available online Software Evolution and Maintenance: A Practitioner's Approach equips readers with a solid understanding of the laws of software engineering, evolution and maintenance models, reengineering techniques, legacy information systems, impact analysis, refactoring, program comprehension, and reuse.
How often do you hear people say things like this? "Our JavaScript is a mess, but we're thinking about using [framework of the month]." Like it or not, JavaScript is not going away. No matter what framework or "compiles-to-js" language or library you use, bugs and performance concerns will always be an issue if the underlying quality of your JavaScript is poor. Rewrites, including porting to the framework of the month, are terribly expensive and unpredictable. The bugs won't magically go away, and can happily reproduce themselves in a new context. To complicate things further, features will get dropped, at least temporarily. The other popular method of fixing your JS is playing "JavaScript Jenga," where each developer slowly and carefully takes their best guess at how the out-of-control system can be altered to allow for new features, hoping that this doesn't bring the whole stack of blocks down. This book provides clear guidance on how best to avoid these pathological approaches to writing JavaScript: Recognize you have a problem with your JavaScript quality. Forgive the code you have now, and the developers who made it. Learn repeatable, memorable, and time-saving refactoring techniques. Apply these techniques as you work, fixing things along the way. Internalize these techniques, and avoid writing as much problematic code to begin with. Bad code doesn't have to stay that way. And making it better doesn't have to be intimidating or unreasonably expensive.
Software Evolution and MaintenanceA Practitioner's ApproachJohn Wiley & Sons
This book presents the proceedings of the International Computer Symposium 2014 (ICS 2014), held at Tunghai University, Taichung, Taiwan in December. ICS is a biennial symposium founded in 1973 and offers a platform for researchers, educators and professionals to exchange their discoveries and practices, to share research experiences and to discuss potential new trends in the ICT industry. Topics covered in the ICS 2014 workshops include: algorithms and computation theory; artificial intelligence and fuzzy systems; computer architecture, embedded systems, SoC and VLSI/EDA; cryptography and information security; databases, data mining, big data and information retrieval; mobile computing, wireless communications and vehicular technologies; software engineering and programming languages; healthcare and bioinformatics, among others. There was also a workshop on information technology innovation, industrial application and the Internet of Things. ICS is one of Taiwan's most prestigious international IT symposiums, and this book will be of interest to all those involved in the world of information technology.
Professional Refactoring in C# & ASP.NET
Improving the Design of Existing Code
9th International Conference, PROFES 2008, Monte Porzio Catone, Italy, June 23-25, 2008, Proceedings
WORK EFFECT LEG CODE _p1
Product-Focused Software Process Improvement
4th International Conference, ICMT 2011, Zurich, Switzerland, June 27-28, 2011, Proceedings
Refactoring JavaScript

This is the first book organized around code clone analysis. To cover the broad studies of code clone analysis, this book selects past research results that are important to the progress of the field and updates them with new results and future directions. The first chapter provides an introduction for readers who are inexperienced in the foundation of code clone analysis, defines clones and related terms, and discusses the classification of clones. The chapters that follow are categorized into three main parts to present 1) major tools for code clone analysis, 2) fundamental topics such as evaluation benchmarks, clone visualization, code clone searches, and code similarities, and 3) applications to actual problems. Each chapter includes a valuable reference list that will help readers to achieve a comprehensive understanding of this diverse field and to catch up with the latest research results. Code clone analysis relies heavily on computer science theories such as pattern matching algorithms, computer language, and software metrics. Consequently, code clone analysis can be applied to a variety of real-world tasks in software development and maintenance such as bug finding and program refactoring. This book will also be useful in designing an effective curriculum that combines theory and application of code clone analysis in university software engineering courses.
The capability to design quality software and implement modern information systems is at the core of economic growth in the 21st century. Nevertheless, exploiting this potential is only possible when adequate human resources are available and when modern software engineering methods and tools are used. The recent years have witnessed rapid evolution of software engineering methodologies, including the creation of new platforms and tools which aim to shorten the software design process, raise its quality and cut down its costs. This evolution is made possible through ever-increasing knowledge of software design strategies as well as through improvements in system design and code testing procedures. At the same time, the need for broad access to high-performance and high-throughput computing resources necessitates the creation of large-scale, interactive information systems, capable of processing millions of transactions per seconds. These systems, in turn, call for new, innovative distributed software design and implementation technologies. The purpose of this book is to review and analyze emerging software engineering technologies, focusing on the evolution of design and implementation platforms as well as on novel computer systems related to the development of modern information services.
The Firmware Handbook provides a comprehensive reference for firmware developers looking to increase their skills and productivity. It addresses each critical step of the development process in detail, including how to optimize hardware design for better firmware. Topics covered include real-time issues, interrupts and ISRs, memory management (including Flash memory), handling both digital and analog peripherals, communications interfacing, math subroutines, error handling, design tools, and troubleshooting and debugging. This book is not for the beginner, but rather is an in-depth, comprehensive one-volume reference that addresses all the major issues in firmware design and development, including the pertinent hardware issues. Included CD-Rom contains all the source code used in the design examples, so engineers can easily use it in their own designs
Refactoring is gaining momentum amongst the object oriented programming community. It can transform the internal dynamics of applications and has the capacity to transform bad code into good code. This book offers an introduction to refactoring.
Evolutionary Database Design (paperback)
Concepts, Methodologies, Tools, and Applications
Systems Engineering in the Fourth Industrial Revolution
Information Modelling and Knowledge Bases XXVIII
Software Engineering: Evolution and Emerging Technologies
Agile Software Development Quality Assurance
Information and Communication Technology and Applications

This book constitutes revised selected papers from the Third International Conference on Information and Communication Technology and Applications, ICTA 2020, held in Minna, Nigeria, in November 2020. Due to the COVID-19 pandemic the conference was held online. The 67 full papers were carefully reviewed and selected from 234 submissions. The papers are organized in the topical sections on Artificial Intelligence, Big Data and Machine Learning; Information Security Privacy and Trust; Information Science and Technology.
Includes articles in topic areas such as autonomic computing, operating system architectures, and open source software technologies and applications.
Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.
This book constitutes the refereed proceedings of the 4th International Conference, ICMT 2011, held in Zurich, Switzerland in June 2011. The 14 revised full papers were carefully revised and selected from 51 submissions. The scope of the contributions ranges from theoretical and methodological topics to implementation issues and applications. Topics addressed are such as transformation paradigms and languages, transformation algorithms and strategies, implementation and tools, as well as applications and case studies.
Issues in Information Science Research: 2013 Edition
Code Clone Analysis
Software Engineering and Information Technology
Proceedings of ICTIS 2020, Volume 1
Representation, Analysis, and Refactoring Techniques to Support Code Clone Maintenance
Programming and Problem Solving with C++: Brief Edition
Compiler Construction

An up-to-date guide for using massive amounts of data and novel technologies to design, build, and maintain better systems engineering Systems Engineering in the Fourth Industrial Revolution: Big Data, Novel Technologies, and Modern Systems Engineering offers a guide to the recent changes in systems engineering prompted by the current challenging and innovative industrial environment called the Fourth Industrial Revolution—INDUSTRY 4.0. This book contains advanced models, innovative practices, and state-of-the-art research findings on systems engineering. The contributors, an international panel of experts on the topic, explore the key elements in systems engineering that have shifted towards data collection and analytics, available and used in the design and development of systems and also in the later life-cycle stages of use and retirement. The contributors address the issues in a system in which the system involves data in its operation, contrasting with earlier approaches in which data, models, and algorithms were less involved in the function of the system. The book covers a wide range of topics including five systems engineering domains: systems engineering and systems thinking; systems software and process engineering; the digital factory; reliability and maintainability modeling and analytics; and organizational aspects of systems engineering. This important resource: Presents new and advanced approaches, methodologies, and tools for designing, testing, deploying, and maintaining advanced complex systems Explores effective evidence-based risk management practices Describes an integrated approach to safety, reliability, and cyber security based on system theory Discusses entrepreneurship as a multidisciplinary system Emphasizes technical merits of systems engineering concepts by providing technical models Written for systems engineers, Systems Engineering in the Fourth Industrial Revolution offers an up-to-date resource that contains the best practices and most recent research on the topic of systems engineering.
This book consists of sixty-seven selected papers presented at the 2015 International Conference on Software Engineering and Information Technology (SEIT2015), which was held in Guilin, Guangxi, China during June 26–28, 2015. The SEIT2015 has been an important event and has attracted many scientists, engineers and researchers from academia, government laboratories and industry internationally. The papers in this book were selected after rigorous review. SEIT2015 focuses on six main areas, namely, Information Technology, Computer Intelligence and Computer Applications, Algorithm and Simulation, Signal and Image Processing, Electrical Engineering and Software Engineering. SEIT2015 aims to provide a platform for the global researchers and practitioners from both academia as well as industry to meet and share cutting-edge development in the field. This conference has been a valuable opportunity for researchers to share their knowledge and results in theory, methodology and applications of Software Engineering and Information Technology. Contents: Information TechnologyComputing Intelligence and Computer ApplicationsAlgorithm and SimulationSignal and Image ProcessingElectrical EngineeringSoftware Engineering Readership: Researchers and graduate students interested in software engineering and information technology. Key Features:The proceedings collected together R&D results undertaken by researchers in six areas, namely, Information Technology, Computer Intelligence and Computer Applications, Algorithm and Simulation, Signal and Image Processing, Electrical Engineering and Software EngineeringKeywords:Information Technology;Computer Intelligence and Computer Applications;Algorithm and Simulation;Signal and Image Processing;Electrical Engineering and Software Engineering
"Software refactoring is one of the most critical aspects of software maintenance. It improves the quality of the software, reduces potential occurrence of bugs and keeps the code easier to maintain, extend and read. The process of refactoring supports and enables the developers to improve the design of software without changing the behavior. However, the automation of this process is complex for developers and software engineers since it is subjective, time and resource consuming. In this context, many literature reviews have analyzed the existing effort made by researchers to facilitate refactoring, as a core software engineering practice. This paper, aims in integrating all the existing research outcomes by performing a tertiary study on all the secondary studies, done in the area of refactoring. Based on our analysis we notice that there are many area of software refactoring that are under studied. As an outcome of this review, several classifications of existing studies were provided to showcase all the studies targeting the automation of refactoring along with explaining what metrics and objectives were used as means to drive refactoring and how it was assessed. This thesis also aims in unveiling areas of future directions for the research community in order to consolidate their efforts in improving the refactoring as a practice."--Abstract.
Getting the most out of Python to improve your codebase Key Features Save maintenance costs by learning to fix your legacy codebase Learn the principles and techniques of refactoring Apply microservices to your legacy systems by implementing practical techniques Book Description Python is currently used in many different areas such as software construction, systems administration, and data processing. In all of these areas, experienced professionals can find examples of inefficiency, problems, and other perils, as a result of bad code. After reading this book, readers will understand these problems, and more importantly, how to correct them. The book begins by describing the basic elements of writing clean code and how it plays an important role in Python programming. You will learn about writing efficient and readable code using the Python standard library and best practices for software design. You will learn to implement the SOLID principles in Python and use decorators to improve your code. The book delves more deeply into object oriented programming in Python and shows you how to use objects with descriptors and generators. It will also show you the design principles of software testing and how to resolve software problems by implementing design patterns in your code. In the final chapter we break down a monolithic application to a microservice one, starting from the code as the basis for a solid platform. By the end of the book, you will be proficient in applying industry approved coding practices to design clean, sustainable and readable Python code. What you will learn Set up tools to effectively work in a development environment Explore

how the magic methods of Python can help us write better code Examine the traits of Python to create advanced object-oriented design Understand removal of duplicated code using decorators and descriptors Effectively refactor code with the help of unit tests Learn to implement the SOLID principles in Python Who this book is for This book will appeal to team leads, software architects and senior software engineers who would like to work on their legacy systems to save cost and improve efficiency. A strong understanding of Programming is assumed.

Object-Oriented Design Knowledge: Principles, Heuristics and Best Practices

Professional Refactoring in Visual Basic

Green in Software Engineering

Working Effectively with Legacy Code

Software Evolution and Maintenance

22nd International Conference, CC 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013, Proceedings

Based off the highly successful Programming and Problem Solving with C++ which Dale is famous for, comes the new Brief Edition, perfect for the one-term course. The text was motivated by the need for a text that covered only what instructors and students are able to move through in a single semester. Important Notice: The digital edition of this book is missing some of the images or content found in the physical edition

Refactoring is an effective way to quickly uncover problematic code and fix it. In this first book to provide a hands-on approach to refactoring in C# and ASP.NET, you'll discover to apply refactoring techniques to manage and modify your code. Plus, you'll learn how to build a prototype application from scratch and discover how to refactor the prototype into a properly designed, enterprise-level application. With the help of step-by-step directions, you'll gain a better understanding of different code issues and refactoring transformations. Many of these transformations are developed from real-world scenarios that are the result of key business decisions. In addition, you'll find formal definitions of refactoring techniques that you'll be able to refer to while on the job. This book covers the refactoring techniques that will enable you to become more efficient and productive. You'll be able to use this information to respond to change and improve the design of existing code. What you will learn from this book How to assemble your own refactoring toolkit Techniques for performing unit testing Tips on refactoring to patterns How to use refactoring to upgrade legacy C# and ASP.NET code Ways to take advantage of the method extraction to eliminate duplicated code How to make code simpler, easier to modify, and more understandable All about object oriented theory and design patterns Methods for using LINQ and other C# 3.0 enhancements Who this book is for This book is for C# and ASP.NET developers who want to learn how to effectively manage and modify their code with refactoring tools and features. Wrox Professional guides are planned and written by working programmers to meet the real-world needs of programmers, developers, and IT professionals. Focused and relevant, they address the issues technology professionals face every day. They provide examples, practical solutions, and expert education in new technologies, all designed to help programmers do a better job.

Refactoring has proven its value in a wide range of development projects–helping software professionals improve system designs, maintainability, extensibility, and performance. Now, for the first time, leading agile methodologist Scott Ambler and renowned consultant Pramodkumar Sadalage introduce powerful refactoring techniques specifically designed for database systems. Ambler and Sadalage demonstrate how small changes to table structures, data, stored procedures, and triggers can significantly enhance virtually any database design–without changing semantics. You'll learn how to evolve database schemas in step with source code–and become far more effective in projects relying on iterative, agile methodologies. This comprehensive guide and reference helps you overcome the practical obstacles to refactoring real-world databases by covering every fundamental concept underlying database refactoring. Using start-to-finish examples, the authors walk you through refactoring simple standalone database applications as well as sophisticated multi-application scenarios. You'll master every task involved in refactoring database schemas, and discover best practices for deploying refactorings in even the most complex production environments. The second half of this book systematically covers five major categories of database refactorings. You'll learn how to use refactoring to enhance database structure, data quality, and referential integrity; and how to refactor both architectures and methods. This book provides an extensive set of examples built with Oracle and Java and easily adaptable for other languages, such as C#, C++, or VB.NET, and other databases, such as DB2, SQL Server, MySQL, and Sybase. Using this book's techniques and examples, you can reduce waste, rework, risk, and cost–and build database systems capable of evolving smoothly, far into the future.

This book constitutes the refereed proceedings of the Fourth International Symposium on Search-Based Software Engineering, SSBSE 2012, held in Riva del Garda, Italy in collocation with the 28th IEEE International Conference on Software Maintenance. The 15 revised full papers, 3 revised short papers, and 2 papers of the graduate track presented together with 2 keynote talks and 1 tutorial paper were carefully reviewed and selected from 38 initial submissions. Search-based Software Engineering (SBSE) studies the application of meta-heuristic optimization techniques to various software engineering problems, ranging from requirements engineering to software testing and maintenance. The papers present current research in all areas of Search Based Software Engineering, including theoretical work, research on SBSE applications, empirical studies, and reports on industrial experience.

Automated Software Maintenance Using Search-based Refactoring

The Mikado Method

Feature Based Methodology for Supporting Architecture Refactoring and Maintenance of Long Life Software Systems

Software Applications: Concepts, Methodologies, Tools, and Applications

Product Focused Software Process Improvement

Clean Code in Python

Turning Bad Code Into Good Code

Information modelling and knowledge bases are now essential, not only to academics working in computer science, but also wherever information technology is applied. This book presents papers from the 26th International Conference on Information Modelling and Knowledge Bases (formerly the European Japanese Conference - EJC), which took place in Tampere, Finland, in June 2016. The conference provides a platform to bring together researchers and practitioners working with information modelling and knowledge bases, and the 33 accepted papers cover topics including: conceptual modelling; knowledge and information modelling and discovery; linguistic modelling; cross-cultural communication and social computing; environmental modelling and engineering; and multimedia data modelling and systems. All papers were improved and resubmitted for publication after the conference. Covering state-of-the-art research and practice, the book will be of interest to all those whose work involves information modelling and knowledge bases.

This book gathers papers addressing state-of-the-art research in all areas of information and communication technologies and their applications in intelligent computing, cloud storage, data mining and software analysis. It presents the outcomes of the Fourth International Conference on Information and Communication Technology for Intelligent Systems, which was held in Ahmedabad, India. Divided into two volumes, the book discusses the fundamentals of various data analysis techniques and algorithms, making it a valuable resource for researchers and practitioners alike.

Data science is an emerging field and innovations in it need to be explored for the success of society 5.0. This book not only focuses on the practical applications of data science to achieve computational excellence, but also digs deep into the issues and implications of intelligent systems. This book highlights innovations in data science to achieve computational excellence that can optimize performance of smart applications. The book focuses on methodologies, framework, design issues, tools, architectures, and technologies necessary to develop and understand data science and its emerging applications in the present era. This book will be useful for the research community, start-up entrepreneurs, academicians, and data centered industries and professors that are interested in exploring innovations in varied applications and areas of data science.

"The software engineering community has advanced greatly in recent years and we currently have numerous defined items of knowledge, such as standards, methodologies, methods, metrics, techniques, languages, patterns, knowledge related to processes, concepts, etc.The main objective of this book is to give a unified and global vision about Micro-Architectural Design Knowledge, analyzing the main techniques, experiences and methods"--Provided by publisher.

Proceedings of the 2015 International Conference on Software Engineering and Information Technology (SEIT2015)

Refactoring

18th International Conference, PROFES 2017, Innsbruck, Austria, November 29-December 1, 2017, Proceedings

The Firmware Handbook

A Handbook for People Who Care About Code

Programming and Problem Solving with C++

Fourth International Symposium, SSBSE 2012, Riva del Garda, September 28-30, 2012, Proceedings

"This book provides the research and instruction used to develop and implement software quickly, in small iteration cycles, and in close cooperation with the customer in an adaptive way, making it possible to react to changes set by the constant changing business environment. It presents four values explaining extreme programming (XP), the most widely adopted agile methodology"--Provided by publisher.

This is the first book that presents a comprehensive overview of sustainability aspects in software engineering. Its format follows the structure of the SWEBOK and covers the key areas involved in the incorporation of green aspects in software engineering, encompassing topics from requirement elicitation to quality assurance and maintenance, while also considering professional practices and economic aspects. The book consists of thirteen chapters, which are structured in five parts. First the " Introduction " gives an overview of the primary general concepts related to Green IT, discussing what Green in Software Engineering is and how it differs from Green by Software Engineering. Next " Environments, Processes and Construction " presents green software development environments, green software engineering processes and green software construction in general. The third part, " Economic and Other Qualities, " details models for measuring how well software supports green software engineering techniques and for performing trade-off analyses between alternative green practices from an economic perspective. " Software Development Process " then details techniques for incorporating green aspects at various stages of software development, including requirements engineering, design, testing, and maintenance. In closing, " Practical Issues " addresses the repercussions of green software engineering on decision-making, stakeholder participation and innovation management. The audience for this book includes software engineering researchers in academia and industry seeking to understand the challenges and impact of green aspects in software engineering, as well as practitioners interested in learning about the state of the art in Green in Software Engineering.

Langlebige Software-Systeme durchlaufen viele bedeutende Veraenderungen im Laufe ihres Lebenszyklus, um der Weiterentwicklung der Problemdomaenen zu folgen. Normalerweise ist es schwierig eine Software-Systemarchitektur den schnellen Weiterentwicklungen einer Problemdomaene anzupassen und mit der Zeit wird der Unterschied zwischen der Problemdomaene und der Software-Systemarchitektur zu gro ß , um weitere Softwareentwicklung sinnvoll fortzufuehren. Fristgerechte Refactorings der Systemarchitektur sind notwendig, um dieses Problem zu vermeiden. Aufgrund des verhaeltnismae ß ig hohen Gefahrenpotenzials und des zeitlich stark verzoegerten Nutzens von Refactorings, werden diese Ma ß nahmen normalerweise bis zum letztmoeglichen Zeitpunkt hinausgeschoben. In der Regel ist das Management abgeneigt Architektur-Refactorings zu akzeptieren, au ß er diese sind absolut notwendig. Die bevorzugte Vorgehensweise ist, neue Systemmerkmale ad hoc hinzuzufuegen und nach dem Motto " Aendere nie etwas an einem funktionierenden System! " vorzugehen. Letztlich ist das Ergebnis ein Architekturzerfall (Architekturdrift). Die Notwendigkeit kleiner Refactoring-Schritte fuehrt zur Notwendigkeit des Architektur-Reengineerings. Im Gegensatz zum Refactoring, das eine normale Entwicklungstaetigkeit darstellt, ist Reengineering eine Form der Software- " Revolution " . Reengineeringprojekte sind sehr riskant und kostspielig. Der Nutzen des Reengineerings ist normalerweise nicht so hoch wie erwartet. Wenn nach dem Reengineering schlie ß lich die erforderlichen Architekturaenderungen statt.nden, kann dies zu spaet sein. Trotz der enormen in das Projekt gesteckten Bemuehungen erfuellen die Resultate des Reengineerings normalerweise nicht die Erwartungen. Es kann passieren, dass sehr bald ein neues, kostspieliges Reengineering erforderlich wird. In dieser Arbeit werden das Problem der Softwareevolution und der Zerfall von Softwarearchitekturen behandelt. Eine Methode wird vorgestellt, welche die Softwareentwicklung in ihrer entscheidenden Phase, dem Architekturrefactoring, unterstuetzt. Die Softwareentwicklung wird sowohl in technischer als auch organisatorischer Hinsicht unterstuetzt. Diese Arbeit hat neue Techniken entwickelt, welche die Reverse-Engineering-, Architecture-Recovery- und Architecture-Redesign-Taetigkeiten unterstuetzen. Sie schlaegt auch Aenderungen des Softwareentwicklungsprozesses vor, die fristgerechte Architekturrefactorings erzwingen koennen und damit die Notwendigkeit der Durchfuehrung eines Architektur-Reengineerings vermeiden. In dieser Arbeit wird die Merkmalmodellierung als Hauptinstrument verwendet. Merkmale werden genutzt, um die Abstraktionsluecke zwischen den Anforderungen der Problemdomaene und der Systemarchitektur zu fuellen. Merkmalmodelle werden auch als erster Grundriss fr die Wiederherstellung der verlorenen Systemarchitektur genutzt. Merkmalbasierte Analysen fuehren zu diversen, nuetzlichen Hinweisen fuer den erneuten Entwurf (das Re-Design) einer Architektur. Schlie ß lich wird die Merkmalmodellierung als Kommunikationsmittel zwischen unterschiedlichen Projektbeteiligten (Stakeholdern) im Verlauf des Softwareengineering-Prozesses verwendet und auf dieser Grundlage wird ein neuer Anforderungsde.nitionsprozess vorgeschlagen, der die erforderlichen Architekturrefactorings erzwingt.

Issues in Information Science Research / 2013 Edition is a ScholarlyEditions™ book that delivers timely, authoritative, and comprehensive information about Web and Grid Services. The editors have built Issues in Information Science Research: 2013 Edition on the vast information databases of ScholarlyNews.™ You can expect the information about Web and Grid Services in this book to be deeper than what you can access anywhere else, as well as consistently reliable, authoritative, informed, and relevant. The content of Issues in Information Science Research: 2013 Edition has been produced by the world ' s leading scientists, engineers, analysts, research institutions, and companies. All of the content is from peer-reviewed sources, and all of it is written, assembled, and edited by the editors at ScholarlyEditions™ and available exclusively from us. You now have a source you can cite with authority, confidence, and credibility. More information is available at http://www.ScholarlyEditions.com/.

Intelligent Systems and Applications

Refactor your legacy code base

Big Data, Novel Technologies, and Modern Systems Engineering

Research, Tools, and Practices

Search-based Refactoring for Software Maintenance

Data Science and Innovations for Intelligent Systems

5th International Conference, PROFES 2004, Kansai Science City, Japan, April 5-8, 2004, Proceedings

This book constitutes the proceedings of the 22nd International Conference on Compiler Construction, CC 2013, held as part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, which took place in Rome, Italy, in March 2013. The 13 papers presented in this book were carefully reviewed and selected from 53 submissions. They have been organized into five topical sections on register allocation, pointer analysis, data and information flow, machine learning, and refactoring.

This book constitutes the refereed proceedings of the 18th International Conference on Product-Focused Software Process Improvement, PROFES 2017, held in Innsbruck, Austria, in November/December 2017. The 17 revised full papers presented together with 10 short papers, 21 workshop papers. 3 posters and tool demonstrations papers, and 4 tutorials were carefully reviewed and selected from 72 submissions. The papers are organized in topical sections on : Agile software Development; Data science and analytics; Software engineering processes and frameworks; Industry relevant qualitative research; User and value centric approaches; Software startups; Serum; Software testing.

This is volume 74 of "Advances in Computers", subtitled "Recent Advances in Software Development". This series, which began in 1960, is the oldest continuously published series of books that has chronicled the ever changing landscape of information technology. Each year three volumes are published, each presenting five to seven chapters describing the latest technology in the use of computers today. In this current volume, we present six chapters that give an update on some of the major issues affecting the development of software today. The six chapters in this volume can be divided into two general categories. The first three deal with the increasing importance of security in the software we write and provide insights into how to increase that security. The three latter chapters look at software development as a whole and provide guidelines in how best to make certain decisions on a project-level basis.

Provides students and engineers with the fundamentaldevelopments and common practices of software evolution andmaintenance Software Evolution and Maintenance: A Practitioner'sApproach introduces readers to a set of well-roundededucational materials, covering the fundamental developments insoftware evolution and common maintenance practices in theindustry. Each chapter gives a clear understanding of a particulartopic in software evolution, and discusses the main ideas withdetailed examples. The authors first explain the basic concepts andthen drill deeper into the important aspects of software evolution.While designed as a text in an undergraduate course in softwareevolution and maintenance, the book is also a great resourceforsoftware engineers, information technology professionals, andgraduate students in software engineering. Based on the IEEE SWEBOK (Software Engineering Body ofKnowledge) Explains two maintenance standards: IEEE/EIA 1219 andISO/IEC14764 Discusses several commercial reverse and domain engineeringtoolkits Slides for instructors are available online Software Evolution and Maintenance: APractitioner's Approach equips readers with a solidunderstanding of the laws of software engineering, evolution andmaintenance models, reengineering techniques, legacy informationsystems, impact analysis, refactoring, program comprehension, andreuse.

Advances and Challenges in Software Refactoring

Proceedings of the International Computer Symposium (ICS) Held at Taichung, Taiwan, December 12 – 14, 2014

Refactoring Databases

Theory and Practice of Model Transformations

Search Based Software Engineering

Third International Conference, ICTA 2020, Minna, Nigeria, November 24–27, 2020, Revised Selected Papers

Completely revised and updated with the latest version of C++, the new Fifth Edition of Programming and Problem Solving with C++ provides the clearest introduction to C++, object-oriented programming, and software development available. Renowned author team Nell Dale and Chip Weems are careful to include all topics and guidelines put forth by the ACM/IEEE. A new chapter on Data Structures makes this text ideal for the one- or two-term course. New Software Maintenance Case Studies teach students how to read code in order to debug, alter, or enhance existing class or code segments. Important Notice: The digital edition of this book is missing some of the images or content found in the physical edition

If you're passionate about programming and want to get better at it, you've come to the right source. Code Craft author Pete Goodliffe presents a collection of useful techniques and approaches to the art and craft of programming that will help boost your career and your well-being. The book's standalone chapters span the range of a software developer's life--dealing with code, learning the trade, and improving performance--with no language or industry bias.