Empirical Analysis Of Programming Language Adoption | cf4721d0d667ffe0b2072c545c06045d

Modern Compiler Implementation in ML

This book gathers the proceedings of the 10th International Conference on Frontier Computing, held in Singapore, on July 10–13, 2020, and provides comprehensive coverage of the latest advances and trends in information technology, science, and engineering. It addresses a number of broad themes, including communication networks, business intelligence and knowledge management, web intelligence, and related fields that inspire the development of information technology. The respective contributions cover a wide range of topics: database and data mining, networking and communications, web and Internet of things, embedded systems, soft computing, social network analysis, security and privacy, optical communication, and ubiquitous/pervasive computing. Many of the papers outline promising future research directions, and the book benefits students, researchers, and professionals alike. Further, it offers a useful reference guide for newcomers to the field.

Modern Compiler Implementation in C

Complex information structures are found in many disciplines including physics, genetics, biology and all branches of the information sciences. The current increasing, widespread use of information technology in all academic activities emphasizes the need to understand how people construct and use such structures. The practices and activities found within the community of programmers provides a rich area for study. The contents of this book are devoted to fundamental research that directly informs: the teaching community about some of the recent issues and problems in computer science education; the help readers when designing systems to support teaching, learning and using information technology; the psychology of the programming community about work in the area of learning to build, and debug programs; and the software engineering community in terms of the issues that implementors need to take into account when designing and building tools and environments for computer-based systems.

Human-computer Interaction, INTERACT ’03

This book constitutes the refereed proceedings of two workshops held at the 24th International Conference on Financial Cryptography and Data Security, FC 2020, in Kota Kinabalu, Malaysia, in February 2020. The 39 full papers and 3 short papers presented in this book were carefully reviewed and selected from 73 submissions. The papers feature four Workshops. The 1st Asian Workshop on Usable Security, AsaUSEC 2020, the 1st Workshop on Coordination of Decentralized Finance, CoDeFi 2020, the 5th Workshop on Advances in Secure Electronic Voting, VOTING 2020, and the 4th Workshop on Trusted Smart Contracts, WTSC 2020. The AsaUSEC Workshop contributes an increase of the scientific quality of research in human factors in security and privacy. In terms of improving efficacy of secure systems, the research included an extension of graphical password authentication. Further a comparative study of SpotBugs, SonarCube, Cryptoguard and CogniCrypt identified strengths in each and refined the need for improvements in security testing tools. The CoDeFi Workshop discusses multi-disciplinary approaches to understanding blockchains and cryptocurrencies and opens up discussions of decentralized finance based on permissionless blockchains. The workshop consists of two parts: presentations by all stakeholders, and unconference style discussions. The VOTING Workshop covers topics like new methods for risk-limited audits, new oaths to increase the efficiency of mixnets, verification of security of voting schemes and auditing, electronic voting system usability, and new technical designs for cryptographic protocols for voting systems, and new ways of preventing vote-stashing by de-incentivising this via smart contracts. The WTSC Workshop focuses on smart contracts, i.e., self-enforcing agreements in the form of executable programs, and other decentralized applications that are deployed to and run on top of specialized blockchains.

Innovative Teaching Strategies and New Learning Paradigms in Computer Programming

Robert Sedgewick has rewritten and substantially expanded his popular work to provide current and comprehensive coverage of important algorithms and data structures. Many new algorithms are presented, and the explanations of each algorithm are much more detailed than in previous editions. A new text design and detailed, innovative figures, with accompanying commentary, greatly enhance the presentation. The third edition retains the successful blend of theory and practice that has made Sedgewick's work an invaluable resource for more than 250,000 programmers! This particular book, Parts 1–4, represents the essential first half of Sedgewick's complete work. It provides extensive coverage of fundamental data structures and algorithms for sorting, searching, and related applications. The algorithms and data structures are expressed in concise implementations in C, so that you can both appreciate their fundamental properties and test them on real applications. Of course, the substance of the book applies to programming in any language. Highlights: Expanded coverage of arrays, linked lists, strings, trees, and other basic data structures Greater emphasis on abstract data types (ADTs) than in previous editions, including ADT implementations, symbol table ADT (searching) implementations New implementations of binomial queues, multway radix sorting, Batchet's sorting networks, randomized BSTs, splay trees, skip lists, multway tries, and much more Increased quantitative information about the algorithms, including empirical studies and basic analytic studies, giving you a basis for comparing them Over 1000 new exercises to help you learn the properties of algorithms Whether you are a student learning the algorithms for the first time or a professional interested in improving your data structures, you will find a wealth of useful information in this book.

Learning to Build and Comprehend Complex Information Structures

Parallel to the growth of computer usage in society is the growth of programming instruction in schools. This informative volume unites a wide range of perspectives on the study of novice programmers who will not only inform readers of empirical findings, but will also provide insights into how novices reason and solve problems within complex domains. The large variety of methodologies found in these studies helps to improve programming instruction and makes this an invaluable reference for researchers planning studies of their own. Topics discussed include historical perspectives, transfer, learning, bugs, and programming environments.

Computer Science 2

Encyclopedia of Microcomputers

This book constitutes the refereed proceedings of the 22nd European Symposium on Programming, ESOP 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 31 papers, presented together with a full-length invited talk, were carefully reviewed and selected from 120 full submissions. The contributions are organized according to ten topical sections on programming techniques; programming tools; separation logic; gradual typing; shared memory concurrency and verification; process calculus; timing concurrency; model checking and verification; weak memory concurrency and verification; and types, inference, and analysis.

Aspect-Oriented Programming: evaluated: A Study on the Impact that Aspect-Oriented Programming can have on Software Development Productivity

Empirical Studies of Programmers
This is an authoritative introduction to Computing Education research written by over 50 leading researchers from academia and the industry.

**Studying the Novice Programmer**

Redundancies in program source code - software clones - are a common phenomenon. Although it is often claimed that software clones decrease the maintainability of software systems and need to be managed, research in the last couple of years showed that not all clones can be considered harmful. A sophisticated assessment of the relevance of software clones and a cost-benefit analysis of clone management is needed to gain a better understanding of cloning and whether it is truly a harmful phenomenon. This thesis introduces techniques to model, analyze, and evaluate versatile aspects of software clone evolution within the history of a system. We present a mapping of non-identical clones across multiple versions of a system, that avoids possible ambiguities of previous approaches. Though processing more data to determine the context of each clone to avoid an ambiguous mapping, the approach is shown to be efficient and applicable to large systems for a retrospective analysis of software clone evolution. The approach has been used in several studies to gain insights into the phenomenon of cloning in open-source as well as industrial software systems. Our results show that non-identical clones require more attention regarding clone management compared to identical clones as they are the dominating clone type for the main share of our subject systems. Using the evolution model to investigate costs and benefits of refactorings that remove clones, we conclude that clone removals could not reduce maintenance costs for most systems under study.

**Empirical Studies on the Development of Executable Business Processes**

This work brings together papers written by researchers and practitioners actively working in the field of human-computer interaction. It should be of use to students who study information technology and computer sciences, and to professional designers who are interested in User Interface design.

**Financial Cryptography and Data Security**

In recent years, searching for source code on the web has become increasingly common among professional software developers and is emerging as an area of academic research. This volume surveys past research and presents the state of the art in the area of "code retrieval on the web." This work is concerned with the algorithms, systems, and tools to programmers to search for source code on the web and the empirical studies of these inventions and practices. It is a label that we apply to a set of related research from software engineering, information retrieval, human-computer interaction, management, as well as commercial products. The division of code retrieval on the web into snippet remixing and component reuse is driven both by empirical data, and analysis of existing search engines and tools. Contributors include leading researchers from human-computer interaction, software engineering, programming languages, and management. "Finding Source Code on the Web for Remix and Reuse" consists of five parts. Part I is titled "Programmers and Practices," and consists of a retrospective chapter and two empirical studies on how programmers search the web for source code. Part II is titled "From Data Structures to Infrastructures," and covers the creation of ground-breaking search engines for code retrieval required ingenuity in the adaptation of existing technology and in the creation of new algorithms and data structures. Part III focuses on "Reuse: Components and Projects," which are reused with minimal modification. Part IV is on "Remix: Snippets and Answers," which examines how source code from the web can also be used as solutions to problems and answers to questions. The book concludes with Part V, "Looking Ahead," that looks at future programming and the legacies of software reuse and remix and the implications of current intellectual property law on the future of software development. The story, "Richie Boss: Private Investigator Manager," was selected as the winner of a crowdfunding short story contest.

**Software Language Engineering**

Describes all phases of a modern compiler, including techniques in code generation and register allocation for imperative, functional and object-oriented languages.

**Empirical Studies of Programmers**

This volume contains the papers presented at the second workshop on Empirical Studies of Programmers. They represent a variety of approaches and topics covering the research in this area. All the chapters present research that bears on programmers. Together with the first volume edited by Elliot Soloway and Sathirama Iyengar, these chapters contribute to a growing knowledge base about how programmers go about their task and how they progress from novice to expert levels.

**Programming Languages and System Architectures**

This book collects essential research on the practical application of executable business process modeling in real-world projects, i.e., model-driven solutions for the support and automation of digital business processes that are created using languages such as BPEL or BPMN. It mainly focuses on empirical research, but also includes an up-to-date cross-section of case studies in order to assess examples of BPM’s practical impact in the industry. On the one hand, executable models are formally and precisely defined so that computers can interpret and execute them, and hence visualize what can be done. On the other, they are visualized so that humans can describe, document and optimize business processes at a higher level of abstraction than with traditional textual programming languages. While these important research areas have long been separated from one another, this book is an attempt at cross-fertilization, driven by the insight that business processes are the software behind today’s digital organizations, and that achieving a precise representation of such processes is key to their reliable execution. Consequently, the book presents various case studies and experiments that investigate questions of interest to both academia (e.g., identifying challenges for which no solution exists; sharing new insights into how existing approaches are actually used) and industry (e.g., guidelines on using certain technologies and on modeling comprehensive and executable models). Both researchers and practitioners will benefit from the presentation of how concepts are transformed into working solutions. The studies are presented in a structured manner and with sufficient rigor to be considered empirical research, further enhancing the book’s value for the research community, while practitioners will find concrete guidance on making the right decisions for their projects.

**Empirical Studies of Programmers**

Covering a variety of areas including software analysis, design, coding and maintenance, this text details the research conducted since the 1970s in this fast-developing field before going on to define a computer program from the viewpoint of computing and cognitive psychology. Two essential sides of programming, software production and software understanding, are given detailed treatment, with parallels drawn throughout between studies on processing texts written in natural language and processing computer programs. Of particular interest to researchers, practitioners and graduates in cognitive psychology, cognitive ergonomics and computer science.

**Thinking with Diagrams**

"The Encyclopedia of Microcomputers serves as the ideal companion reference to the popular Encyclopedia of Computer Science and Technology. Now in its 10th year of publication, this timely reference work details the broad spectrum of microcomputer technology, including microcomputer history; explains and illustrates the use of microcomputers throughout academia, business, government, and society in general; and assesses the future impact of this rapidly changing technology."
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 design and maintain software. Software maintenance is moving from code maintenance to design maintenance, even on specification maintenance. Modifications today are made at specification level, regenerating the software components, testing and integrating them with the system. Eventually software maintenance has to manage the evolution and variational 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.


"This comprehensive reference work provides immediate, fingertip access to state-of-the-art technology in nearly 700 self-contained articles written by over 900 international authorities. Each article in the Encyclopedia features current developments and trends in computers, software, vendors, and applications extensive bibliographies of leading figures in the field, such as Samuel Alexander, John von Neumann, and Norbert Wiener and in-depth analysis of future directions."

Programming Languages and Systems

This book constitutes the refereed proceedings of the 9th International Conference on Software Reuse, ICSR 2006, held in Torino, Italy, in June 2006. The book presents 27 revised full papers and 13 revised short papers, carefully reviewed and selected from numerous submissions. The Coverage includes COTS selection and integration; product lines, domain analysis, and variability; reengineering maintenance; programming languages and retrieval; aspect-oriented software development; approaches and models; and components.

Programming Languages and Systems

Master the art of building analytical models using R. About This Book Load, wrangle, and analyze your data using the world's most powerful statistical programming language. Build and customize publication-quality visualizations of powerful and stunning R graphs. Develop key skills and techniques with R to create and customize data mining algorithms. Use R to optimize your trading strategy and build up your own risk management system. Discover how to build machine learning algorithms, prepare data, and dig deep into data prediction techniques with R. This Book is For This course is for data scientists or quantitative analyst who are looking at learning R and take advantage of its powerful analytical design framework. It's a seamless journey in becoming a full-stack R developer. What You Will Learn Describe and visualize the behavior of data and relationships between data. Gain a thorough understanding of statistical reasoning and sampling. Handle missing data gracefully. Use multiple imputation. Create diverse types of bar charts using the default R functions. Familiarize yourself with algorithms written in R for spatial data mining, text mining, and so on. Understand relationships between market factors and their impact on your portfolio. Harness the power of R to build machine learning algorithms with real-world data science applications. Learn specialized machine learning techniques for text mining, big data, and more. In Detail The R learning path created for you has five connected modules, which are a mini-course in their own right. As you complete each one, you'll have gained key skills and be ready for the material in the next module! This course begins by looking at the Data Analysis with R module. This will help you navigate the R environment. You'll gain a thorough understanding of statistical reasoning and sampling. Finally, you'll be able to put best practices into effect to make your job easier and facilitate reproducibility. The second place to explore is R Graphs, which will help you leverage powerful default R graphics and utilize advanced graphics systems such as lattice and ggplot2. The grammar of graphics. You'll learn how to produce, customize, and publish advanced visualizations using this popular and powerful framework. With the third module, Learning Data Mining with R, you will learn how to manipulate data with R using code snippets and be introduced to mining frequent patterns, association, and correlations while working with R programs. The Mastering R for Quantitative Finance module pragmatically introduces both the quantitative finance concepts and their modeling in R, enabling you to build a tailor-made trading system on your own. By the end of the module, you will be well-versed with various financial techniques using R and will be able to place good bets while making financial decisions. Finally, we'll look at the Machine Learning with R module. With this module, you'll discover all the analytical tools you need to gain insights from complex data and learn how to choose the correct algorithm for your specific needs. You'll also learn to apply machine learning methods to common tasks, including classification, prediction, forecasting, and so on. Style and approach Learn data analysis, data visualization techniques, data mining, and machine learning all using R and also learn to build models in quantitative finance using this powerful language.

Finding Source Code on the Web for Remix and Reuse

Courses in computer programming combine a number of different concepts, from complex problem-solving to mathematical precepts such as algorithms and computational intelligence. Due to the complex nature of computer science education, teaching the novice programmer can be a challenge. Innovative Teaching Strategies and New Learning Paradigms in Computer Programming brings together pedagogical and technological methods to address the recent challenges that have developed in computer programming courses. Focusing on educational tools, computer science concepts, and educational design, this book is an essential reference source for teachers, practitioners, and scholars interested in improving the success rate of students.

Encyclopedia of Computer Science and Technology

This book constitutes the refereed proceedings of the 8th International Conference on Software Analysis, Testing, and Evolution, SATE 2018. The conference was co-located with the national Software Application Conference, NASAC 2018, and was held in Shenzhen, Guangdong, in November 2018. The 13 full papers presented were carefully reviewed and selected from 34 submissions. The papers describe results related to software analysis, testing and evolution, including theoretical research, empirical study, new technology, case study and industrial practice.

Empirical Studies of Programmers

The methods and thinking of economics permeate a large part of the IS discipline. Reciprocally, newly emerging research methods relying on the IT-enabled treatment of massive data aggregates feed economic research. As new and radical forms of IT innovation continue to energize electronic commerce, IS researchers face a daunting task in using existing empirical methods and tools to understand the threats, opportunities, risks, and rewards of these new technologies. This groundbreaking volume leads the way. It introduces new methodological approaches to data analysis as well as new techniques for collecting transactional data. The ideas it presents have broad appeal and demonstrate what is possible when new techniques and new ways of thinking are brought to bear on complex research problems.

Software Design – Cognitive Aspect

This book constitutes the proceedings of the 28th European Conference on Object-Oriented Programming, ECOOP 2014, held in Uppsala, Sweden, in July/August 2014. The 27 papers presented in this volume were carefully reviewed and selected from 101 submissions. They are organized in topical sections named: analysis; design; concurrency; types; implementation; refactoring; JavaScript, PHP and frameworks; and parallelism.

Algorithms in C, Parts 1-4

This book provides an introductory overview of the rapid growth in interdisciplinary research into Thinking with Diagrams. Diagrammatic representations are becoming more common in everyday human experience, yet they offer unique challenges to cognitive science research. Neither linguistic nor perceptual theories are sufficient to completely explain their advantages and applications. These research challenges may be part of the reason why so many diagrams are badly designed or badly used. This is ironic when the user interfaces of computer software and the worldwide web are becoming so completely dominated by graphical and diagrammatic representations. This book includes chapters commissioned from leading researchers in the major disciplines involved in diagrams research. They review the philosophical status of diagrams, the cognitive processes involved in their application, and a range of specialist fields in which diagrams are central, including education, architectural design and visual programming languages. The result is immediately relevant to researchers in cognitive science and artificial intelligence, as well as in applied technology areas such as human-computer interaction and information design.

Frontier Computing

This book constitutes the thoroughly refereed post-conference proceedings of the First International Conference on Software Language Engineering, SLE 2008, held in Toulouse, France, in September 2008. The 16 revised full papers and 1 revised short paper presented together...
Empirical Research in Software Engineering

This volume studies programmers to gain insights to facilitate improved productivity and quality software. The chapters cover a range of topics including cognitive models of programming; measuring program complexity; the effects of program style and structure on program comprehension, production and maintenance; documentation; the effects of control structures and data structures on program comprehension, production and maintenance; evaluations of program design and construction methodologies; teaching strategies; and assessment of programmer abilities.

Perspectives on Data Science for Software Engineering

Perspectives on Data Science for Software Engineering presents the best practices of seasoned data miners in software engineering. The idea for this book was created during the 2014 conference at Dagstuhl, an invitation-only gathering of leading computer scientists who meet to identify and discuss cutting-edge informatics topics. At the 2014 conference, the concept of how to transfer the knowledge of experts from seasoned software engineers and data scientists to newcomers in the field highlighted many discussions. While there are many books covering data mining and software engineering basics, they present only the fundamentals and lack the perspective that comes from real-world experience. This book offers unique insights into the wisdom of the community's leaders gathered to share hard-won lessons from the trenches. Ideas are presented in digestible chapters designed to be applicable across many domains. Toolkits can learn the tips and tricks of the trade, while more experienced data scientists will benefit from war stories that show what traps to avoid. Presents the wisdom of community experts, derived from a summit on software analytics Provides contributed chapters that share discrete ideas and technique from the trenches Covers top areas of concern, including mining security and social data, data visualization, and cloud-based data Presented in clear chapters designed to be applicable across many domains

Theory and Practice of Model Transformations

The increased availability and quality of open source software on the Web is creating more opportunities for developers to reuse software and is changing the way developers write source code. It is important to understand how developers look for source code on the Web so that tools and approaches can be suggested to better support developers' needs. Based on different approaches to understand how humans look for information, we propose a five-stage model to differentiate the stages that could take place when developers look for source code on the Web. We use this model to assess the effectiveness of existing tools.

Cognition and Computer Programming

Aspect-oriented programming is a relatively new technique that has evolved on top of the already well-established approach of object-oriented programming. When it is used correctly, it promises to remove many redundant parts of a code that appear repeatedly in an application, essentially untangling the original code. Thus, it can lead to a cleaner, more separated source code design, to greater modularity and maintainability. Time-savings in software engineering can also be huge cost-savings, and anything that increases software quality is a welcome sight in an industry that in parts, still suffers from a bad reputation concerning the quality of its products. But, used in the wrong way or with insufficient knowledge, aspect-oriented programming can result in the opposite. Unintended side effects may introduce new bugs into an application, and therefore, it may be just one of the many negative consequences. In any case, there is a huge lack of empirical research on the subject, and in software engineering in general. Due to that fact, the question arises in how far hard facts can be drawn. This book deals with part of that problem by evaluating the aspect-oriented approach in a carefully designed and controlled experiment. On the basis of different tasks that were solved by participants of this study, the aspect-oriented approach (represented by AspectJ) is compared with a plain object-oriented approach (represented by Java). The book starts with an introduction to the topic, and further, it provides on the one hand, the survey's methodology and on the other, some background information. A short chapter on aspect-oriented programming and empirical research may help readers who are unfamiliar with any of the subjects. Then, the survey itself is described in detail, i.e. its design, its implementation, and its evaluation, followed by a thorough discussion of the results. And the answer to the question 'Can aspect-oriented programming keep its promise in productivity?' is given.

Economics, Information Systems, and Electronic Commerce: Empirical Research

This book constitutes the refereed proceedings of the 8th International Conference on Model Transformation, ICMT 2015, held in L'Aquila, Italy, in July 2015, as Part of STAF 2015, the federation of a number of the leading conferences on software technologies. The 16 revised papers were carefully selected from 34 submissions. The papers are organized in topical sections on change management; reuse and industrial applications; new paradigms for model transformation; transformation validation and verification; and foundations of model transformation.

R: Data Analysis and Visualization

This new, expanded textbook describes all phases of a modern compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as functional and object-oriented languages, that are missing from most books. In addition, more advanced chapters are now included so that it can be used as the basis for two-semester or graduate course. The most accepted and successful techniques are described in a concise way, rather than as an exhaustive catalog of every possible variant. Detailed descriptions of the interfaces between modules of a compiler are illustrated with actual C header files. The first part of the book, Fundamentals of Compilation, is suitable for a one-semester first course in compiler design. The second part, Advanced Topics, which includes the advanced chapters, covers the compilation of object-oriented and functional languages, garbage collection, loop optimizations, SSA form, loop scheduling, and optimization for cache-memory hierarchies.

Software Maintenance - A Management Perspective

This book constitutes the proceedings of the 25th European Symposium on Programming, ESOP 2016, which took place in Eindhoven, The Netherlands, in April 2016, held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016. The 29 papers presented in this volume were carefully reviewed and selected from 98 submissions. Being devoted to fundamental issues in the specification, design, analysis, and implementation of programming languages and systems, ESOP features contributions on all aspects of programming language research; theoretical and/or practical advances.

Reuse of Off-the-Shelf Components

The papers in this volume represent the work presented at the 1996 workshop. One of the goals of the workshop, in 1986, was to bring together the small and disparate group of researchers who were wrestling with difficult and complex issues of programming. The text includes papers, posters, tutorials and panels used at the 1996 workshop.

Empirical Research towards a Relevance Assessment of Software Clones

The Cambridge Handbook of Computing Education Research

Offers information on past and future conferences of the Empirical Studies of Programmers (ESP), provided by the ESP Design Team at the University of Nebraska at Lincoln. Includes indexes of papers and lists of participants, as well as photographs. Links to related sites.

R: Data Analysis and Visualization

This new, expanded textbook describes all phases of a modern compiler: lexical analysis, parsing, abstract syntax, semantic actions, intermediate representations, instruction selection via tree matching, dataflow analysis, graph-coloring register allocation, and runtime systems. It includes good coverage of current techniques in code generation and register allocation, as well as functional and object-oriented languages, that are missing from most books. In addition, more advanced chapters are now included so that it can be used as the basis for two-semester or graduate course. The most accepted and successful techniques are described in a concise way, rather than as an exhaustive catalog of every possible variant. Detailed descriptions of the interfaces between modules of a compiler are illustrated with actual C header files. The first part of the book, Fundamentals of Compilation, is suitable for a one-semester first course in compiler design. The second part, Advanced Topics, which includes the advanced chapters, covers the compilation of object-oriented and functional languages, garbage collection, loop optimizations, SSA form, loop scheduling, and optimization for cache-memory hierarchies.
This volume looks at the obvious trend in the computing evolution from studies of student programmers and toward studies of real programmers performing real programming tasks. The percentage of student and professional programmers in the studies reported in papers presented at ESP 1 and ESP 5 has nearly flipped. There is now an emphasis on the practical application of the research results both in industry and in academia. The papers and posters of this workshop include a range of programming language paradigms beyond the procedural paradigm: functional, logic, visual, object-oriented, and concurrent programming. This appears to be an indication of the maturation of the field. No longer is it a question of whether to study programmers, but more a question of which aspects of the programming that make it an important area of study.

**ECOOP 2014 -- Object-Oriented Programming**

This book constitutes the refereed proceedings of the 4th Computational Methods in Systems and Software 2020 (CoMeSySo 2020) proceedings. Software engineering, computer science and artificial intelligence are crucial topics for the research within an intelligent systems problem domain. The CoMeSySo 2020 conference is breaking the barriers, being held online. CoMeSySo 2020 intends to provide an international forum for the discussion of the latest high-quality research results.

**Software Analysis, Testing, and Evolution**

Empirical research has now become an essential component of software engineering yet software practitioners and researchers often lack an understanding of how the empirical procedures and practices are applied in the field. Empirical Research in Software Engineering: Concepts, Analysis, and Applications shows how to implement empirical research processes, procedures, and practices in software engineering. Written by a leading researcher in empirical software engineering, the book describes the necessary steps to perform replicated and empirical research. It explains how to plan and design experiments, conduct systematic reviews and case studies, and analyze the results produced by the empirical studies. The book balances empirical research concepts with exercises, examples, and real-life case studies, making it suitable for a course on empirical software engineering. The author discusses the process of developing predictive models, such as defect prediction and change prediction, on data collected from source code repositories. She also covers the application of machine learning techniques in empirical software engineering, includes guidelines for publishing and reporting results, and presents popular software tools for carrying out empirical studies.

**Empirical Studies of Programmers**

Programming languages and system architectures are at the frontiers of two different worlds. The conference on which this book is based was an adventure in a land where the two worlds - the formal world of algorithms and the physical world of electronic circuits - interact. The participants explored this land under the guidance of internationally renowned researchers such as Butler W. Lampson, Susan Graham, Jan L.A. van de Snepscheut, and C.A.R. Hoare, all of whom gave invited papers. The volume includes these papers together with sixteen session papers. Subjects of special interest include: programing language design and history, programming environments, programming methods, operating systems, compiler construction, and innovative system architectures.

Copyright code : cf4721d9d67f4a1b572c545c06045d