More Programming Pearls Confessions Of A Coder Confessions Of A Coder Acm Press

With approximately 600 problems and 35 worked examples, this supplement provides a collection of practical problems on the design, analysis and verification of algorithms. The book focuses on the important areas of algorithm design and analysis: background material; algorithm design techniques; advanced data structures and NP-completeness; and miscellaneous problems. Algorithms are expressed in Pascal-like pseudocode supported by figures, diagrams, hints, solutions, and comments. This book is a detailed account of the Synthesizer Generator, a system for creat ing specialized editors that are customized for editing particular languages. The book is intended for those with an interest in software tools and in methods for building interactive systems. It is a must for people who are using the Syn thesizer Generator to build editors because it provides extensive discussions of how to write editor specifications. The book should also be valuable for people who are building specialized editors "by hand," without using an editor generating tool. The need to manage the development of large software

systems is one of the most pressing problems faced by computer programmers. An important aspect of this problem is the design of new tools to aid interactive program develop ment. The Synthesizer Generator permits one to create specialized editors that are tailored for editing a particular language. In program editors built with the Synthesizer Generator, knowledge about the language is used to continuously assess whether a program contains errors and to determine where such errors occur. The information is then displayed on the terminal screen to provide feed back to the programmer as the program is developed and modified.

Project managers, technical leads, and Windows programmers throughout the industry share an important concern--how to get their development schedules under control. Rapid Development addresses that concern head-on with philosophy, techniques, and tools that help shrink and control development schedules and keep projects moving. The style is friendly and conversational--and the content is impressive.

When programmers list their favorite books, Jon Bentley's collection of programming pearls is commonly included among the classics. Just as natural pearls grow from grains of sand that irritate oysters, programming pearls have grown from real problems that have irritated real programmers. With origins beyond solid engineering, in the realm of

insight and creativity, Bentley's pearls offer unique and clever solutions to those nagging problems. Illustrated by programs designed as much for fun as for instruction, the book is filled with lucid and witty descriptions of practical programming techniques and fundamental design principles. It is not at all surprising that Programming Pearls has been so highly valued by programmers at every level of experience. In this revision, the first in 14 years, Bentley has substantially updated his essays to reflect current programming methods and environments. In addition, there are three new essays on testing, debugging, and timing set representations string problems All the original programs have been rewritten, and an equal amount of new code has been generated. Implementations of all the programs, in C or C++, are now available on the Web. What remains the same in this new edition is Bentley's focus on the hard core of programming problems and his delivery of workable solutions to those problems. Whether you are new to Bentley's classic or are revisiting his work for some fresh insight, the book is sure to make your own list of favorites.

Once a worker for an idealistic aid group, a disillusioned Moises Froissard now spends his life seeking the world's most beautiful men, women, and children from among the illegal immigrants, refugees, and dispossessed for Club Olympus, an

exclusive international sex club, in a compelling novel about the international sex trade. Reprint. 10,000 first printing.

Are you doing all you can to further your career as a software developer? With today's rapidly changing and ever-expanding technologies, being successful requires more than technical expertise. To grow professionally, you also need soft skills and effective learning techniques. Honing those skills is what this book is all about. Authors Dave Hoover and Adewale Oshineye have cataloged dozens of behavior patterns to help you perfect essential aspects of your craft. Compiled from years of research, many interviews, and feedback from O'Reilly's online forum, these patterns address difficult situations that programmers, administrators, and DBAs face every day. And it's not just about financial success. Apprenticeship Patterns also approaches software development as a means to personal fulfillment. Discover how this book can help you make the best of both your life and your career. Solutions to some common obstacles that this book explores in-depth include: Burned out at work? "Nurture Your Passion" by finding a pet project to rediscover the joy of problem solving. Feeling overwhelmed by new information? Re-explore familiar territory by building something you've built before, then use "Retreat into Competence" to move forward again. Stuck in your learning? Seek a team of experienced and talented

developers with whom you can "Be the Worst" for a while. "Brilliant stuff! Reading this book was like being in a time machine that pulled me back to those key learning moments in my career as a professional software developer and, instead of having to learn best practices the hard way, I had a guru sitting on my shoulder guiding me every step towards master craftsmanship. I'll certainly be recommending this book to clients. I wish I had this book 14 years ago!"-Russ Miles, CEO, OpenCredo

As an open operating system, Unix can be improved on by anyone and everyone: individuals, companies, universities, and more. As a result, the very nature of Unix has been altered over the years by numerous extensions formulated in an assortment of versions. Today, Unix encompasses everything from Sun's Solaris to Apple's Mac OS X and more varieties of Linux than you can easily name. The latest edition of this bestselling reference brings Unix into the 21st century. It's been reworked to keep current with the broader state of Unix in today's world and highlight the strengths of this operating system in all its various flavors. Detailing all Unix commands and options, the informative guide provides generous descriptions and examples that put those commands in context. Here are some of the new features you'll find in Unix in a Nutshell, Fourth Edition: Solaris 10, the latest version of the SVR4-based operating system, GNU/Linux, and Mac OS X Bash shell

(along with the 1988 and 1993 versions of ksh) tsch shell (instead of the original Berkeley csh) Package management programs, used for program installation on popular GNU/Linux systems, Solaris and Mac OS X GNU Emacs Version 21 Introduction to source code management systems Concurrent versions system Subversion version control system GDB debugger As Unix has progressed, certain commands that were once critical have fallen into disuse. To that end, the book has also dropped material that is no longer relevant, keeping it taut and current. If you're a Unix user or programmer, you'll recognize the value of this complete, up-to-date Unix reference. With chapter overviews, specific examples, and detailed command.

This newly expanded and updated second edition of the best-selling classic continues to take the "mystery" out of designing algorithms, and analyzing their efficacy and efficiency. Expanding on the first edition, the book now serves as the primary textbook of choice for algorithm design courses while maintaining its status as the premier practical reference guide to algorithms for programmers, researchers, and students. The reader-friendly Algorithm Design Manual provides straightforward access to combinatorial algorithms technology, stressing design over analysis. The first part, Techniques, provides accessible instruction on methods for designing and analyzing computer

algorithms. The second part, Resources, is intended for browsing and reference, and comprises the catalog of algorithmic resources, implementations and an extensive bibliography. NEW to the second edition: • Doubles the tutorial material and exercises over the first edition • Provides full online support for lecturers, and a completely updated and improved website component with lecture slides, audio and video • Contains a unique catalog identifying the 75 algorithmic problems that arise most often in practice, leading the reader down the right path to solve them • Includes several NEW "war stories" relating experiences from real-world applications • Provides up-to-date links leading to the very best algorithm implementations available in C, C++, and Java

Implementing physical simulations for real-time games is a complex task that requires a solid understanding of a wide range of concepts from the fields of mathematics, physics, and software engineering. This book is a gems-like collection of practical articles in the area of game physics. Each provides hands-on detail that can be used in practical

Literate programming is a programming methodology that combines a programming language with a documentation language, making programs more easily maintained than programs written only in a high-level language. A literate programmer is an essayist who writes programs for humans to understand. When programs are written in the recommended style they can be transformed into documents by a document compiler and into efficient code by an algebraic compiler. This anthology of essays includes Knuth's

early papers on related topics such as structured programming as well as the Computer Journal article that launched literate programming. Many examples are given, including excerpts from the programs for TeX and METAFONT. The final essay is an example of CWEB, a system for literate programming in C and related languages. Index included.

Computer-Mediated Communication Systems: Status and Evaluation synthesizes current knowledge about computerized conferencing systems, electronic mail, and office information-communication systems. It should be of interest both to students and researchers studying this new form of electronic communication and to organizations that are planning the installation of electronic mail or other computer-mediated communication systems and that need to be aware of the information gleaned from the studies presented here. The book is organized into four main sections, focusing on the following issues: (1) What are the important considerations in designing software or choosing a system from the many available options and capabilities? (2) What factors determine whether such systems are likely to be accepted or rejected? (3) What are the likely impacts of such systems upon the individuals, groups, and organizations which use them? It is not the economic costs and benefits, but the social problems and ""payoffs"" in the form of enhanced performance and organizational efficiency that should be the main considerations in deciding whether or not to use a computer-mediated communication system. (4) Given the conditional nature of many of the possible impacts, no system should be implemented without formal evaluation and feedback from users to guide the implementation. The major kinds of evaluational strategies that have been successfully employed are described in this book.

Dive deep into the Go language and become an expert Go

developer Key Features Second edition of the bestselling guide to advanced Go programming, expanded to cover machine learning, more Go packages and a range of modern development techniques Completes the Go developer's education with real-world guides to building high-performance production systems Packed with practical examples and patterns to apply to your own development work Clearly explains Go nuances and features to remove the frustration from Go development Book Description Often referred to (incorrectly) as Golang, Go is the high-performance systems language of the future. Mastering Go, Second Edition helps you become a productive expert Go programmer, building and improving on the groundbreaking first edition. Mastering Go, Second Edition shows how to put Go to work on real production systems. For programmers who already know the Go language basics, this book provides examples, patterns, and clear explanations to help you deeply understand Go's capabilities and apply them in your programming work. The book covers the nuances of Go, with in-depth guides on types and structures, packages, concurrency, network programming, compiler design, optimization, and more. Each chapter ends with exercises and resources to fully embed your new knowledge. This second edition includes a completely new chapter on machine learning in Go, guiding you from the foundation statistics techniques through simple regression and clustering to classification, neural networks, and anomaly detection. Other chapters are expanded to cover using Go with Docker and Kubernetes, Git, WebAssembly, JSON, and more. If you take the Go programming language seriously, the second edition of this book is an essential guide on expert techniques. What you will learn Clear guidance on using Go for production systems Detailed explanations of how Go internals work, the design choices behind the language, and how to optimize your Go

code A full guide to all Go data types, composite types, and data structures Master packages, reflection, and interfaces for effective Go programming Build high-performance systems networking code, including server and client-side applications Interface with other systems using WebAssembly, JSON, and gRPC Write reliable, high-performance concurrent code Build machine learning systems in Go, from simple statistical regression to complex neural networks Who this book is for Mastering Go, Second Edition is for Go programmers who already know the language basics, and want to become expert Go practitioners.

In this #1 New York Times bestseller, Nora Roberts takes readers deep into the rugged hills of South Dakota, where the shadows keep secrets, hunters stalk the land, and a friendship matures into something more.... Cooper Sullivan spent the summers of his youth on his grandparents' South Dakota ranch, sharing innocent games and stolen kisses with the neighbor girl, Lil Chance. Now, twelve years after they last walked together hand in hand, fate has brought them back to the Black Hills. Though the memory of Coop's touch still haunts her, Lil has let nothing stop her dream of opening the Chance Wildlife Refuge, but something—or someone—has been keeping a close watch. When small pranks and acts of destruction escalate into a heartless attack on Lil's beloved cougar, memories of an unsolved murder have Coop springing to action to keep Lil safe. Both of them know the natural dangers that lurk in the wild landscape of the Black Hills. But a killer of twisted and unnatural instincts has singled them out as prey....

Classic on practical methods of optimizing programs: This book gives practical advice on improving the efficiency (optimizing) programs and the limits there of. While showing how to trade off speed for space or vice-versa, the author points out the limits that can be expected to gain. His list of

techniques is a collection of practical approaches rather than theoretical possibilities. At 158 pages (not counting index) this book is eminently readable, accessable and useful. Clearly written and well organized this is a book to keep on your shelf for when a program needs improving. It is also a book to read before a program as a reminder not to make things complicated with optimization that aren't needed.

Multi-Paradigm Design for C++ offers insight into an analysis and design process that takes advantage of C++'s multiple paradigm capability. It uses understandable notation and readable explanations to help all C++ programmers - not just system architects and designers - combine multiple paradigms in their application development for more effective, efficient, portable, robust, and reusable software. Readers will gain an understanding of domain engineering methods that support multi-paradigm design. This book reveals how to analyze the application domain, using principles of commonality and variation, to define subdomains according to the most appropriate paradigm for each. Multi-paradigm design digs deeper than any single technology or technique to address fundamental questions of software abstraction and design.

Drawing on 20+ years helping software teams succeed in nearly 150 organizations, Karl Wiegers presents 60 concise lessons and practical recommendations students can apply to all kinds of projects, regardless of application domain, technology, development lifecycle, or platform infrastructure. Embodying both wisdom for deeper understanding and guidance for practical use, this book represent an invaluable complement to the technical nuts and bolts software developers usually study. Software Development Pearls covers multiple crucial domains of project success: requirements, design, project management, culture and teamwork, quality, and process improvement. Each chapter

suggests several first steps and next steps to help you begin immediately applying the author's hard-won lessons--and writing code that is more successful in every way that matters.

An off-the-beaten-path tour of the city's hidden highlights, and the stories behind them. London is full of curiosities. Who knew that beneath the Albert Memorial lies a chamber resembling a church crypt? Or that there are catacombs under Camden? Who would expect to find a lighthouse in East London, sphinxes in South London, dummy houses in West London, or a huge bust of film director Alfred Hitchcock in North London? How many of those who walk past Cleopatra's Needle pause to consider why a 3,000-year-old Egyptian monument stands beside the Thames? How many know that what was once London's smallest police station can be seen in Trafalgar Square? Or that pineapples are used in the architectural design of so many buildings? Or why there are memorials to the Mayflower and Pilgrim Fathers in Rotherhithe? Learn more about the capital of curiosities in this delightful guide for lovers of history, trivia, and travel.

An exploration of how design might be led by marginalized communities, dismantle structural inequality, and advance collective liberation and ecological survival. What is the relationship between design, power, and social justice? "Design justice" is an approach to design that is led by marginalized

communities and that aims expilcitly to challenge, rather than reproduce, structural inequalities. It has emerged from a growing community of designers in various fields who work closely with social movements and community-based organizations around the world. This book explores the theory and practice of design justice, demonstrates how universalist design principles and practices erase certain groups of people—specifically, those who are intersectionally disadvantaged or multiply burdened under the matrix of domination (white supremacist heteropatriarchy, ableism, capitalism, and settler colonialism)—and invites readers to “build a better world, a world where many worlds fit; linked worlds of collective liberation and ecological sustainability.” Along the way, the book documents a multitude of real-world community-led design practices, each grounded in a particular social movement. Design Justice goes beyond recent calls for design for good, user-centered design, and employment diversity in the technology and design professions; it connects design to larger struggles for collective liberation and ecological survival.

Modern number theory began with the work of Euler and Gauss to understand and extend the many unsolved questions left behind by Fermat. In the course of their investigations, they uncovered new phenomena in need of explanation, which over time led to the discovery of field theory and its intimate

connection with complex multiplication. While most texts concentrate on only the elementary or advanced aspects of this story, Primes of the Form x2 + ny2 begins with Fermat and explains how his work ultimately gave birth to quadratic reciprocity and the genus theory of quadratic forms. Further, the book shows how the results of Euler and Gauss can be fully understood only in the context of class field theory. Finally, in order to bring class field theory down to earth, the book explores some of the magnificent formulas of complex multiplication. The central theme of the book is the story of which primes p can be expressed in the form x2 + ny2. An incomplete answer is given using quadratic forms. A better though abstract answer comes from class field theory, and finally, a concrete answer is provided by complex multiplication. Along the way, the reader is introduced to some wonderful number theory. Numerous exercises and examples are included. The book is written to be enjoyed by readers with modest mathematical backgrounds. Chapter 1 uses basic number theory and abstract algebra, while chapters 2 and 3 require Galois theory and complex analysis, respectively.

In the twenty years since its publication, Celebration of Discipline has helped over a million seekers discover a richer spiritual life infused with joy, peace, and a deeper understanding of God. For this special twentieth anniversary edition, Richard J. Foster has

added an introduction, in which he shares the story of how this beloved and enduring spiritual guidebook came to be. Hailed by many as the best modern book on Christian spirituality, Celebration of Discipline explores the "classic Disciplines," or central spiritual practices, of the Christian faith. Along the way, Foster shows that it is only by and through these practices that the true path to spiritual growth can be found. Dividing the Disciplines into three movements of the Spirit, Foster shows how each of these areas contribute to a balanced spiritual life. The inward Disciplines of meditation, prayer, fasting, and study, offer avenues of personal examination and change. The outward Disciplines of simplicity, solitude, submission, and service, help prepare us to make the world a better place. The corporate Disciplines of confession, worship, guidance, and celebration, bring us nearer to one another and to God. Foster provides a wealth of examples demonstrating how these Disciplines can become part of our daily activities-and how they can help us shed our superficial habits and "bring the abundance of God into our lives." He offers crucial new insights on simplicity, demonstrating how the biblical view of simplicity, properly understood and applied, brings joy and balance to our inward and outward lives and "sets us free to enjoy the provision of God as a gift that can be shared with others." The discussion of celebration, often the most neglected

of the Disciplines, shows its critical importance, for it stands at the heart of the way to Christ. Celebration of Discipline will help motivate Christians everywhere to embark on a journey of prayer and spiritual growth.

Covers Expression, Structure, Common Blunders, Documentation, & Structured Programming Techniques

Do you know what "quatrefoil" and "impolitic" mean? What about "halcyon" or "narcolepsy"? This book is a handy, easy-to-read reference guide to the proper parlance for any situation. In this book you will find: Words You Absolutely Should Know (covert, exonerate, perimeter); Words You Should Know But Probably Don't (dour, incendiary, scintilla); Words Most People Don't Know (schlimazel, thaumaturgy, epergne); Words You Should Know to Sound Overeducated (ad infinitum, nugatory, garrulity); Words You Probably Shouldn't Know (priapic, damnatory, labia majora); and more. Whether writing an essay, studying for a test, or trying to impress friends, family, and fellow cocktail party guests with their prolixity, you will achieve magniloquence, ebullience, and flights of rhetorical brilliance.

Percolation theory is the study of an idealized random medium in two or more dimensions. The emphasis of this book is upon core mathematical material and the presentation of the shortest and most accessible proofs. Much new material appears

in this second edition including dynamic and static renormalization, strict inequalities between critical points, a sketch of the lace expansion, and several essays on related fields and applications.

Now in the 5th edition, Cracking the Coding Interview gives you the interview preparation you need to get the top software developer jobs. This book provides: 150 Programming Interview Questions and Solutions: From binary trees to binary search, this list of 150 questions includes the most common and most useful questions in data structures, algorithms, and knowledge based questions. 5 Algorithm Approaches: Stop being blind-sided by tough algorithm questions, and learn these five approaches to tackle the trickiest problems. Behind the Scenes of the interview processes at Google, Amazon, Microsoft, Facebook, Yahoo, and Apple: Learn what really goes on during your interview day and how decisions get made. Ten Mistakes Candidates Make -- And How to Avoid Them: Don't lose your dream job by making these common mistakes. Learn what many candidates do wrong, and how to avoid these issues. Steps to Prepare for Behavioral and Technical Questions: Stop meandering through an endless set of questions, while missing some of the most important preparation techniques. Follow these steps to more thoroughly prepare in less time.

More Programming PearlsConfessions of a CoderAddison-

Wesley Professional

Until now, no other book examined the gap between the theory of algorithms and the production of software programs. Focusing on practical issues, A Programmer's Companion to Algorithm Analysis carefully details the transition from the design and analysis of an algorithm to the resulting software program. Consisting of two main complementary

Richard Bird takes a radical approach to algorithm design, namely, design by calculation. These 30 short chapters each deal with a particular programming problem drawn from sources as diverse as games and puzzles, intriguing combinatorial tasks, and more familiar areas such as data compression and string matching. Each pearl starts with the statement of the problem expressed using the functional programming language Haskell, a powerful yet succinct language for capturing algorithmic ideas clearly and simply. The novel aspect of the book is that each solution is calculated from an initial formulation of the problem in Haskell by appealing to the laws of functional programming. Pearls of Functional Algorithm Design will appeal to the aspiring functional programmer, students and teachers interested in the principles of algorithm design, and anyone seeking to master the techniques of reasoning about programs in an equational style.

This book describes the paleomagnetism of sediments and sedimentary rocks, how sediments and sedimentary rocks become magnetized, and how the physical and chemical processes involved can affect the accuracy of paleomagnetism. Topics covered include depositional and post-depositional remanence acquisition, the detection and correction of compaction-caused inclination shallowing, reduction diagenesis of magnetic minerals, chemical remagnetization, and rotation of remanence by grain-scale rock strain. The book also has a chapter on environmental

paleomagnetism, including examples of the new technique of high-resolution rock magnetic cyclostratigraphy and its application to sedimentary sequences. By emphasising the accuracy of sedimentary paleomagnetism and the magnitude of post-depositional processes that can affect it, the book will be invaluable in the geologic interpretation of sedimentary paleomagnetic data. Paleomagnetism of Sedimentary Rocks will be welcomed by paleomagnetists, students of paleomagnetism and all Earth scientists who use sedimentary paleomagnetic data in their research. Additional resources for this book can be found at:
www.wiley.com/go/kodama/paleomagnetism.

Fleeing home from his military service in Afghanistan when his wife dies in an apparent freak household accident, Dr. Mike Scanlon struggles with the tragedy, his inability to bond with his new baby daughter and a downsizing in his medical practice only to discover a shocking secret that changes his understanding of everything. By the Edgar Award-winning author of Come Home. 300,000 first printing.

Shell scripting skills never go out of style. It's the shell that unlocks the real potential of Unix. Shell scripting is essential for Unix users and system administrators-a way to quickly harness and customize the full power of any Unix system. With shell scripts, you can combine the fundamental Unix text and file processing commands to crunch data and automate repetitive tasks. But beneath this simple promise lies a treacherous ocean of variations in Unix commands and standards. Classic Shell Scripting is written to help you reliably navigate these tricky waters.Writing shell scripts requires more than just a knowledge of the shell language, it also requires familiarity with the individual Unix programs: why each one is there, how to use them by themselves, and in combination with the other programs. The authors are intimately familiar with the tips and tricks that can be used to

create excellent scripts, as well as the traps that can make your best effort a bad shell script. With Classic Shell Scripting you'll avoid hours of wasted effort. You'll learn not only write useful shell scripts, but how to do it properly and portably.The ability to program and customize the shell quickly, reliably, and portably to get the best out of any individual system is an important skill for anyone operating and maintaining Unix or Linux systems. Classic Shell Scripting gives you everything you need to master these essential skills.

Do you have a real relationship with God, or do you just have a religion? Do you know God, or do you just know about God? In How Big Is Your God? Paul Coutinho, SJ, challenges us to grow stronger and deeper in our faith and in our relationship with God—a God whose love knows no bounds. To help us on our way, Coutinho introduces us to people in various world religions—from Hindu friends to Buddhist teachers to St. Ignatius of Loyola—who have shaped his spiritual life and made possible his deep, personal relationship with God.

Covers advanced features of Perl, how the Perl interpreter works, and presents areas of modern computing technology such as networking, user interfaces, persistence, and code generation.

This collection of essays drawn from Plauger's popular "Programming on Purpose" column in the magazine Computer Language, focuses on the technology of writing computer software. Plauger's style is clear without being simplistic, reducing complex themes to bite-size chunks. KEY TOPICS: Covers a number of important technical themes such as computer arithmetic, approximating math functions, human perception and artificial intelligence, encrypting data and clarifying documentation.

Every team needs a leader, but why do we so often take that to mean that the appropriate workplace team needs to consist

of one gem of a worker complemented with a bunch of obedient order-takers and yes men? What if the complementary fits between the team members were not with how well they performed the tasks handed down to them but with how they all used their unique strengths to share knowledge, push the envelope, and lead together in the challenge before them?The team of authors behind A Team of Leaders wants to show readers how to design systems within their organization and management procedures that nurture the leadership potential of every employee, not just the ones they ear-marked as having potential for promotion. The proven principles and techniques within these invaluable pages include:• The Five-Stage Team Development Model that maps the transition from traditional to self-directed teams• Best practices in team process design • A Team Value Creation Tool that allows members to appreciate the significance of what they contribute each day• Visual Management • And moreThe key to your company's success is creating successful teams of leaders combining their individual talents and strengths into a single, unstoppable driving force. The fresh approach taught in this indispensable guide will transform passive groups of disparate people into the effective teams of leaders you didn't know was possible to have.

Why bibliometrics is useful for understanding the global dynamics of science but generate perverse effects when applied inappropriately in research evaluation and university rankings. The research evaluation market is booming. "Ranking," "metrics," "h-index," and "impact factors" are reigning buzzwords. Government and research administrators want to evaluate everything—teachers, professors, training programs, universities—using quantitative indicators. Among the tools used to measure "research excellence," bibliometrics—aggregate data on publications and

citations—has become dominant. Bibliometrics is hailed as an "objective" measure of research quality, a quantitative measure more useful than "subjective" and intuitive evaluation methods such as peer review that have been used since scientific papers were first published in the seventeenth century. In this book, Yves Gingras offers a spirited argument against an unquestioning reliance on bibliometrics as an indicator of research quality. Gingras shows that bibliometric rankings have no real scientific validity, rarely measuring what they pretend to. Although the study of publication and citation patterns, at the proper scales, can yield insights on the global dynamics of science over time, ill-defined quantitative indicators often generate perverse and unintended effects on the direction of research. Moreover, abuse of bibliometrics occurs when data is manipulated to boost rankings. Gingras looks at the politics of evaluation and argues that using numbers can be a way to control scientists and diminish their autonomy in the evaluation process. Proposing precise criteria for establishing the validity of indicators at a given scale of analysis, Gingras questions why universities are so eager to let invalid indicators influence their research strategy.

Most software project problems are sociological, not technological. Peopleware is a book on managing software projects.

How humans and technology evolve together in a creative partnership. In this book, Edward Ashford Lee makes a bold claim: that the creators of digital technology have an unsurpassed medium for creativity. Technology has advanced to the point where progress seems limited not by physical constraints but the human imagination. Writing for both literate technologists and numerate humanists, Lee makes a case for engineering—creating technology—as a deeply intellectual and fundamentally creative process.

Explaining why digital technology has been so transformative and so liberating, Lee argues that the real power of technology stems from its partnership with humans. Lee explores the ways that engineers use models and abstraction to build inventive artificial worlds and to give us things that we never dreamed of—for example, the ability to carry in our pockets everything humans have ever published. But he also attempts to counter the runaway enthusiasm of some technology boosters who claim everything in the physical world is a computation—that even such complex phenomena as human cognition are software operating on digital data. Lee argues that the evidence for this is weak, and the likelihood that nature has limited itself to processes that conform to today's notion of digital computation is remote. Lee goes on to argue that artificial intelligence's goal of reproducing human cognitive functions in computers vastly underestimates the potential of computers. In his view, technology is coevolving with humans. It augments our cognitive and physical capabilities while we nurture, develop, and propagate the technology itself. Complementarity is more likely than competition.

Software -- Software Engineering.

The topic is of prime importance to software professionals involved in large development efforts such as databases, operating systems, compilers, and frameworks. This volume explains the process of decomposing large systems into physical (not inheritance) hierarchies of small, manageable components. Concepts and techniques are illustrated with "war stories" from the development firm, Mentor Graphics, as well as with a large-scale example comprising some 12,000 lines of code. Annotation copyright by Book News, Inc., Portland, OR