Software Design Decoded 66 Ways Experts Think

Accident investigation and risk assessment have for decades focused on the human factor, particularly ‘human error’. This bias towards performance failures leads to a neglect of normal performance. It assumes that failures and successes have different origins so there is little to be gained from studying them together. Erik Hollnagel believes this assumption is false and that safety cannot be attained only by eliminating risks and failures. The alternative is to understand why things go right and to amplify that. The ETTO Principle looks at the common trait of people at work to adjust what they do to match the conditions. It proposes that this efficiency-thoroughness trade-off (ETTO) is normal. While in some cases the adjustments may lead to adverse outcomes, these are due to the same processes that produce successes.

Join the technological revolution that’s taking the financial world by storm. Mastering Bitcoin is your guide through the seemingly complex world of bitcoin, providing the knowledge you need to participate in the internet of money. Whether you’re building the next killer app, investing in a startup, or simply curious about the technology, this revised and expanded second edition provides essential detail to get you started. Bitcoin, the first successful decentralized digital currency, is still in its early stages and yet it’s already spawned a multi-billion-dollar global economy open to anyone with the knowledge and passion to participate. Mastering Bitcoin provides the knowledge. You simply supply the passion. The second edition includes: A broad introduction of bitcoin and its underlying blockchain—ideal for non-technical users, investors, and business executives An explanation of the technical foundations of bitcoin and cryptographic currencies for developers, engineers, and software and systems architects Details of the bitcoin decentralized network, peer-to-peer architecture, transaction lifecycle, and security principles New developments such as Segregated Witness, Payment Channels, and Lightning Network A deep dive into blockchain applications, including how to combine the building blocks offered by this platform into higher-level applications User stories, analogies, examples, and code snippets illustrating key technical concepts

In this groundbreaking book Phil Barden reveals what decision science explains about people’s purchase behaviour, and specifically demonstrates its value to marketing. He shares the latest research on the motivations behind consumers’ choices and what happens in the human brain as buyers make their decisions. He deciphers the ‘secret codes’ of products, services and brands to explain why people buy them. And finally he shows how to apply this knowledge in day to day marketing to great effect by dramatically improving key factors such as relevance, differentiation and credibility. Shows how the latest insights from the fields of Behavioural Economics, psychology and neuro-economics explain why we buy what we buy Offers a pragmatic framework and guidelines for day-to-day marketing practice on how to employ this knowledge for more effective brand management - from strategy to implementation and NPD. The first book to apply Daniel Kahneman’s Nobel Prize-winning work to marketing and advertising Packed with case studies, this is a must-read for marketers, advertising professionals, web designers, R&D managers, industrial designers, graphic designers in fact anyone whose role or interest focuses on the ‘why’ behind consumer behaviour. Foreword by Rory Sutherland, Executive Creative Director and Vice-Chairman, OgilvyOne London and Vice-Chairman,Ogilvy Group UK Full colour throughout

The design and analysis of efficient data structures has long been recognized as a key component of the Computer Science curriculum. Goodrich, Tomassia and Goldwasser's approach to this classic topic is based on the object-oriented paradigm as the framework of choice for the design of data structures. For each ADT presented in the text, the authors provide an associated Java interface. Concrete data structures realizing the ADTs are provided as Java classes implementing the interfaces. The Java code implementing fundamental data structures in this book is organized in a single Java package, net.datastructures. This package forms a coherent library of data structures and algorithms in Java specifically designed for educational purposes in a way that is complimentary with the Java Collections Framework.

What Every BODY is Saying

Understanding and Using Symbols in Visual Communication

Why Smart Engineers Write Bad Code

The Elements of Computing Systems

A Circular Stroll Through the Hidden Connections of the English Language

A Review

An industry insider explains why there is so much bad software—and why academia doesn't teach programmers what industry wants them to know. Why is software so prone to bugs? So vulnerable to viruses? Why are software products so often delayed, or even canceled? Is software development really hard, or are software developers just not that good at it? In The Problem with Software, Adam Barr examines the proliferation of bad software, explains what causes it, and offers some suggestions on how to improve the situation. For one thing, Barr points out, academia doesn't teach programmers what they actually need to know to do their jobs: how to work in a team to create code that works reliably and can be maintained by somebody other than the original authors. As the size and complexity of commercial software have grown, the gap between academic computer science and industry has widened. It's an open secret that there is little engineering in software engineering, which continues to rely not on codified scientific knowledge but on intuition and experience. Barr, who worked as a programmer for more than twenty years, describes how the industry has evolved, from the era of mainframes and Fortran to today's embrace of the cloud. He explains bugs and why software has so many of them, and why today's interconnected computers offer fertile ground for viruses and worms. The difference between good and bad software can be a single line of code, and Barr includes code to illustrate the consequences of seemingly inconsequential choices by programmers. Looking to the future, Barr writes that the best prospect for improving software engineering is the move to the cloud. When software is a service and not a product, companies will have more incentive to make it good rather than “good enough to ship."

Unauthorized guide to the underpinnings of the English language.

Decoded is a book like no other: a collection of lyrics and their meanings that together tell the story of a culture, an art form, a moment in history, and one of the most provocative and successful artists of our time. Praise for Decoded “Compelling . . . provocative, evocative . . . Part autobiography, part lavishly illustrated commentary on the author’s own work, Decoded gives the reader a harrowing portrait of the rough worlds Jay-Z navigated in his youth, while at the same time deconstructing his lyrics.”—Michiko Kakutani, The New York Times “One of a handful of books that just about any hip hop fan should own.”—The New Yorker “Elegantly designed, incisively written . . . an impressive leap by a man who has never been known for small steps.”—Los Angeles Times “A riveting exploration of Jay-Z’s journey . . . So thoroughly engrossing, it reads like a good piece of cultural journalism.”—The Boston Globe “Shawn Carter’s most honest airing of the experiences he drew on to create the mythic figure of Jay-Z . . . The scenes he recounts along the way are fascinating.”—Entertainment Weekly “Hip-hop’s renaissance man drops a classic. . . . Heartfelt, passionate and slick.”—Kirkus Reviews (starred review)

Beginning with a basic primer on reverse engineering-including computer internals, operating systems, and assembly language-and then discussing the various applications of reverse engineering, this book provides readers with practical, in-depth techniques for software reverse engineering. The book is broken into two parts, the first deals with security-related reverse engineering and the second explores the more practical aspects of reverse engineering. In addition, the author explains how to reverse engineer a third-party software library to improve interfacing and how to reverse engineer a competitor's software to build a better product. * The first popular book to show how software reverse engineering can help defend against security threats, speed up development, and unlock the secrets of competitive products * Helps developers plug security holes by demonstrating how hackers exploit reverse engineering techniques to crack copy-protection schemes and identify software targets for viruses and other malware * Offers a primer on advanced reverse-engineering, delving into "disassembly"-code-level reverse engineering-and explaining how to decipher assembly language

A Practical Introduction to Hardware/Software Codesign

Decoding Design

Debugging Teams

What every programmer needs to know about cognition

Digital System Design - Use of Microcontroller

Algorithms and Information Retrieval in Java

The Shakespeare Book

This text introduces the spirit and theory of hacking as well as the science behind it all; it also provides some core techniques and tricks of hacking so you can think like a hacker, write your own hacks or thwart potential system attacks.

Based on the popular Artech House classic, Digital Communication Systems Engineering with Software-Defined Radio, this book provides a practical approach to quickly learning the software-defined radio (SDR) concepts needed for work in the field. This up-to-date volume guides readers on how to quickly prototype wireless designs using SDR for real-world testing and experimentation. This book explores advanced wireless communication techniques such as OFDM, LTE, WLA, and hardware targeting. Readers will gain an understanding of the core concepts behind wireless hardware, such as the radio frequency front-end, analog-to-digital and digital-to-analog converters, as well as various processing technologies. Moreover, this volume includes chapters on timing estimation, matched filtering, frame synchronization message decoding, and source coding. The orthogonal frequency division multiplexing is explained and details about HDL code generation and deployment are provided. The book concludes with coverage of the WLAN toolbox with OFDM beacon reception and the LTE toolbox with downlink reception. Multiple case studies are provided throughout the book. Both MATLAB and Simulink source code are included to assist readers with their projects in the field.

Strategies for building large systems that can be easily adapted for new situations with only minor programming modifications. Time pressures encourage programmers to write code that works well for a narrow purpose, with no room to grow. But the best systems are evolvable; they can be adapted for new situations by adding code, rather than changing the existing code. The authors describe techniques they have found effective--over their combined 100-plus years of programming experience--that will help programmers avoid programming themselves into corners. The authors explore ways to enhance flexibility by: • Organizing systems using combinators to compose mix-and-match parts, ranging from small functions to whole arithmetics, with standardized interfaces • Augmenting data with independent annotation layers, such as units of measurement or provenance • Combining independent pieces of partial information using unification or propagation • Separating control structure from problem domain with domain models, rule systems and pattern matching, propagation, and dependency-directed backtracking • Extending the programming language, using dynamically extensible evaluators

"A great book with deep insights into the bridge between programming and the human mind." - Mike Taylor, CGI Your brain responds in a predictable way when it encounters new or difficult tasks. This unique book teaches you concrete techniques rooted in cognitive science that will improve the way you learn and think about code. In The Programmer’s Brain: What every programmer needs to know about cognition you will learn: Fast and effective ways to master new programming languages Speed reading skills to quickly comprehend new code Techniques to unravel the meaning of complex code Ways to learn new syntax and keep it memorized Writing code that is easy for others to read Picking the right names for your variables Making your codebase more understandable to newcomers Onboarding new developers to your team Learn how to optimize your brain’s natural cognitive processes to read code more easily, write code faster, and pick up new languages in much less time. This book will help you through the confusion you feel when faced with strange and complex code, and explain a codebase in ways that can make a new team member productive in days! Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the technology Take advantage of your brain’s natural processes to be a better programmer. Techniques based in cognitive science make it possible to learn new languages faster, improve productivity, reduce the need for code rewrites, and more. This unique book will help you achieve these gains. About the book The Programmer’s Brain unlocks the way we think about code. It offers scientifically sound techniques that can radically improve the way you master new technology, comprehend code, and memorize syntax. You’ll learn how to benefit from productive struggle and turn confusion into a learning tool. Along the way, you’ll discover how to create study resources as you become an expert at teaching yourself and bringing new colleagues up to speed. What's inside Understand how your brain sees code Speed reading skills to learn code quickly Techniques to unravel complex code Tips for making codebases understandable About the reader For programmers who have experience working in more than one language. About the author Dr. Felienne Hermans is an associate professor at Leiden University in the Netherlands. She has spent the last decade researching programming, how to learn and how to teach it. Table of Contents PART 1 ON READING CODE BETTER 1 Decoding your confusion while coding 2 Speed reading for code 3 How to learn programming syntax quickly 4 How to read complex code PART 2 ON THINKING ABOUT CODE 5 Reaching a deeper understanding of code 6 Getting better at solving programming problems 7 Misconceptions: Bugs in thinking PART 3 ON WRITING BETTER CODE 8 How to get better at naming things 9 Avoiding bad code and cognitive load: Two frameworks 10 Getting better at solving complex problems PART 4 ON COLLABORATING ON CODE 11 The act of writing code 12 Designing and improving larger systems 13 How to onboard new developers

The Problem with Software

The Programmer's Brain

Big Ideas Simply Explained

Decoded

Patterns for Effective Interaction Design

Data Structures and Algorithms in Java

Software Testing and Analysis

If you’re a student studying computer science or a software developer preparing for technical interviews, this practical book will help you learn and review some of the most important ideas in software engineering—data structures and algorithms—in a way that’s clearer, more concise, and more engaging than other materials. By emphasizing practical knowledge and skills over theory, author Allen Downey shows you how to use data structures to implement efficient algorithms, and then analyze and measure their performance. You’ll explore the important classes in the Java collections framework (JCF), how they’re implemented, and how they’re expected to perform. Each chapter presents hands-on exercises supported by test code online. Use data structures such as lists and maps, and understand how they work Build an application that reads Wikipedia pages, parses the contents, and navigates the resulting data tree Analyze code to predict how fast it will run and how much memory it will require Write classes that implement the Map interface, using a hash table and binary search tree Build a simple web search engine with a crawler, an indexer that stores web page contents, and a retriever that returns user query results Other books by Allen Downey include Think Java, Think Python, Think Stats, and Think Bayes.

Teaches readers how to test and analyze software to achieve an acceptable level of quality at an acceptable cost Readers will be able to minimize software failures, increase quality, and effectively manage costs Covers techniques that are suitable for near-term application, with sufficient technical background to indicate how and when to apply them Provides balanced coverage of software testing & analysis approaches By incorporating modern topics and strategies, this book will be the standard software-testing textbook

The new RISC-V Edition of Computer Organization and Design features the RISC-V open source instruction set architecture, the first open source architecture designed to be used in modern computing environments such as cloud computing, mobile devices, and other embedded systems. With the post-PC era now upon us, Computer Organization and Design moves forward to explore this generational change with examples, exercises, and material highlighting the emergence of mobile computing and the Cloud. Updated content featuring tablet computers, Cloud infrastructure, and the x86 (cloud computing) and ARM (mobile computing devices) architectures is included. An online companion Web site provides advanced content for further study, appendices, glossary, references, and recommended reading. Features RISC-V, the first such architecture designed to be used in modern computing environments, such as cloud computing, mobile devices, and other embedded systems Includes relevant examples, exercises, and material highlighting the emergence of mobile computing and the cloud

Explore various verticals in software engineering through high-end systems using Python Key FeaturesMaster the tools and techniques used in software engineeringEvaluates available database options and selects one for the final Central Office system-componentsExperience the iterations software go through and craft enterprise-grade systemsBook Description Software Engineering is about more than just writing code—it includes a host of soft skills that apply to almost any development effort, no matter what the language, development methodology, or scope of the project. Being a senior developer all but requires awareness of how those skills, along with their expected technical counterparts, mesh together through a project's life cycle. This book walks you through that discovery by going over the entire life cycle of a multi-tier system and its related software projects. You'll see what happens before any development takes place, and what impact the decisions and designs made at each step have on the development process. The development of the entire project, over the course of several iterations based on real-world Agile iterations, will be executed, sometimes starting from nothing, in one of the fastest growing languages in the world—Python. Application of practices in Python will be laid out, along with a number of Python-specific capabilities that are often overlooked. Finally, the book will implement a high-performance computing solution, from first principles through complete foundation. What you will learnUnderstand what happens over the course of a system's life (SDLC)Establish what to expect from the pre-development life cycle stepsFind out how the development-specific phases of the SDLC affect developmentUncover what a real-world development process might be like, in an Agile wayFind out how to do more than just write the codeIdentify the existence of project-independent best practices and how to use themFind out how to design and implement a high-performance computing processWho this book is for Hands-On Software Engineering with Python is for you if you are a developer having basic understanding of programming and its paradigms and want to skill up as a senior programmer. It is assumed that you have basic Python knowledge.

Software Design and Development

Computational Support for Sketching in Design

Hidden Histories: A Spotter's Guide to the British Landscape

Mastering Bitcoin

Object-Oriented Analysis and Design

Building a Modern Computer from First Principles

An Ex-FBI Agent's Guide to Speed-Reading People

Computational Support for Sketching in Design surveys the literature on sketch based tools from journals, conference proceedings, symposia and workshops in human-computer interaction, cognitive science, design research, computer science, artificial intelligence, and engineering design.

Acknowledgments. Basic Real-Time Concepts. Computer Hardware. Languages Issues. The Software Life Cycle. Real-Time Specification and Design Techniques. Real-Time Kernels. Intertask Communication and Synchronization. Real-Time Memory Management. System Performance Analysis and Optimization. Queuing Models. Reliability, Testing, and Fault Tolerance. Multiprocessing Systems. Hardware/Software Integration. Real-Time Applications. Glossary. Bibliography. Index.

Provides information on writing a driver in Linux, covering such topics as character devices, network interfaces, driver debugging, concurrency, and interrupts.

Software Design Decoded66 Ways Experts ThinkMIT Press

Programming the Open Blockchain

The ETTO Principle: Efficiency-Thoroughness Trade-Off

Secrets of Reverse Engineering

Linux Device Drivers

Much Ado About Mothing
Reversing
Designing and Optimizing System Software

Joe Navarro, a former FBI counterintelligence officer and a recognized expert on nonverbal behavior, explains how to "speed-read" people: decode sentiments and behaviors, avoid hidden pitfalls, and look for deceptive behaviors. You'll also learn how your body language can influence what your boss, family, friends, and strangers think of you. Read this book and send your nonverbal intelligence soaring. You will discover: The ancient survival instincts that drive body language Why the face is the least likely place to gauge a person's true feelings What thumbs, feet, and eyelids reveal about moods and motives The most powerful behaviors that reveal our confidence and true sentiments Simple nonverbals that instantly establish trust Simple nonverbals that instantly communicate authority Filled with examples from Navarro's professional experience, this definitive book offers a powerful new way to navigate your world.
This title gives students an integrated and rigorous picture of applied computer science, as it comes to play in the construction of a simple yet powerful computer system.
The real challenge of programming isn't learning a language's syntax—it's learning to creatively solve problems so you can build something great. In this one-of-a-kind text, author V. Anton Spraul breaks down the ways that programmers solve problems and teaches you what other introductory books often ignore: how to Think Like a Programmer. Each chapter tackles a single programming concept, like classes, pointers, and recursion, and open-ended exercises throughout challenge you to apply your knowledge. You'll also learn how to: -Split problems into discrete components to make them easier to solve -Make the most of code reuse with functions, classes, and libraries -Pick the perfect data structure for a particular job -Master more advanced programming tools like recursion and dynamic memory -Organize your thoughts and develop strategies to tackle particular types of problems Although the book's examples are written in C++, the creative problem-solving concepts they illustrate go beyond any particular language; in fact, they often reach outside the realm of computer science. As the most skillful programmers know, writing great code is a creative art—and the first step in creating your masterpiece is learning to Think Like a Programmer.
*Over the last ten years, the ARM architecture has become one of the most pervasive architectures in the world, with more than 2 billion ARM-based processors embedded in products ranging from cell phones to automotive braking systems. A world-wide community of ARM developers in semiconductor and product design companies includes software developers, system designers and hardware engineers. To date no book has directly addressed their need to develop the system and software for an ARM-based system. This text fills that gap. This book provides a comprehensive description of the operation of the ARM core from a developer's perspective with a clear emphasis on software. It demonstrates not only how to write efficient ARM software in C and assembly but also how to optimize code. Example code throughout the book can be integrated into commercial products or used as templates to enable quick creation of productive software. The book covers both the ARM and Thumb instruction sets, covers Intel's XScale Processors, outlines distinctions among the versions of the ARM architecture, demonstrates how to implement DSP algorithms, explains exception and interrupt handling, describes the cache technologies that surround the ARM cores as well as the most efficient memory management techniques. A final chapter looks forward to the future of the ARM architecture considering ARMv6, the latest change to the instruction set, which has been designed to improve the DSP and media processing capabilities of the architecture. * No other book describes the ARM core from a system and software perspective. * Author team combines extensive ARM software engineering experience with an in-depth knowledge of ARM developer needs. * Practical, executable code is fully explained in the book and available on the publisher's Website. * Includes a simple embedded operating system.*
Real-Time Systems Design and Analysis
How to Avoid Programming Yourself into a Corner
Software-Defined Radio for Engineers
But how Do it Know?
Think Like a Programmer
Essential TypeScript
An Engineer's Handbook

An engaging, illustrated collection of insights revealing the practices and principles that expert software designers use to create great software. What makes an expert software designer? It is more than experience or innate ability. Expert software designers have specific habits, learned practices, and observed principles that they apply deliberately during their design work. This book offers sixty-six insights, distilled from years of studying experts at work, that capture what successful software designers actually do to create great software. The book presents these insights in a series of two-page illustrated spreads, with the principle and a short explanatory text on one page, and a drawing on the facing page. For example, "Experts generate alternatives" is illustrated by the same few balloons turned into a set of very different balloon animals. The text is engaging and accessible; the drawings are thought-provoking and often playful. Organized into such categories as "Experts reflect," "Experts are not afraid," and "Experts break the rules," the insights range from "Experts prefer simple solutions" to "Experts see error as opportunity." Readers learn that "Experts involve the user"; "Experts take inspiration from wherever they can"; "Experts design throughout the creation of software"; and "Experts draw the problem as much as they draw the solution." One habit for an aspiring expert software designer to develop would be to read and reread this entertaining but essential little book. The insights described offer a guide for the novice or a reference for the veteran—in software design or any design profession. A companion web site provides an annotated bibliography that compiles key underpinning literature, the opportunity to suggest additional insights, and more.
Covering more than 100 universal gardening "dos and don'ts," Decoding Gardening Advice is the first book to provide gardeners with the real answers. Jeff Gillman, the bestselling author of The Truth About Garden Remedies, and Meleah Maynard back up every good recommendation with sound horticultural and botanical science. Decoding Gardening Advice is the first and only hard-hitting, evidence-based book that every gardener needs for definitive advice on everything from bulbs, annuals, and perennials to edibles, trees, and soil care.
For the times when you're driving past a lumpy, bumpy field and you wonder what made the lumps and bumps; for when you're walking between two lines of grand trees, wondering when and why they were planted; for when you see a brown heritage sign pointing to a 'tumulus' but you don't know what to look for… Entertaining and factually rigorous, Hidden Histories will help you decipher the story of our landscape through the features you can see around you. This Spotter's Guide arms the amateur explorer with the crucial information needed to 'read' the landscape and spot the human activities that have shaped our green and pleasant land. Photographs and diagrams point out specific details and typical examples to help the curious Spotter 'get their eye in' and understand what they're looking at, or looking for. Specially commissioned illustrations bring to life the processes that shaped the landscape - from medieval ploughing to Roman road building - and stand-alone capsules explore interesting aspects of history such as the Highland Clearances or the coming of Christianity. This unique guide uncovers the hidden stories behind the country's landscape, making it the perfect companion for an exploration of our green and pleasant land.
Work with Typescript and get the most from this versatile open source language. Author Adam Freeman begins this book by describing Typescript and the benefits it offers, and goes on to show you how to use TypeScript in realistic scenarios, going in-depth to give you the knowledge you need. Starting from the nuts-and-bolts and building up to the most advanced and sophisticated features, you will learn how TypeScript builds on the JavaScript type system to create a safer and more productive development experience and understand how TypeScript can be used to create applications using popular frameworks, including Node.js, Angular, React, and Vue.js. Each topic is covered clearly and concisely and is packed with the details you need to learn to be truly effective. The most important features are given a no-nonsense in-depth treatment and chapters include common problems and details of how to avoid them. What You Will LearnGain a solid understanding of the TypeScript language and toolsUse TypeScript for client- and server-side developmentExtend and customize TypeScriptDebug and unit test your TypeScript code Who This Book Is For Developers who want to start using TypeScript, for example to create rich web applications using Angular, React, or Vue.js Adam Freeman is an experienced IT professional who has held senior positions in a range of companies, most recently serving as chief technology officer and chief operating officer of a global bank. Now retired, he spends his time writing and long-distance running.
Think Data Structures
Software Design Decoded
Hands-On Software Engineering with Python
66 Ways Experts Think
Refactoring
Information Theory, Inference and Learning Algorithms
ARM System Developer's Guide

In the course of their 20+-year engineering careers, authors Brian Fitzpatrick and Ben Collins-Sussman have picked up a treasure trove of wisdom and anecdotes about how successful teams work together. Their conclusion? Even among people who have spent decades learning the technical side of their jobs, most haven't really focused on the human component. Learning to collaborate is just as important to success. If you invest in the "soft skills" of your job, you can have a much greater impact for the same amount of effort. The authors share their insights on how to lead a team effectively, navigate an organization, and build a healthy relationship with the users of your software. This is valuable information from two respected software engineers whose popular series of talks—including "Working with Poisonous People"—has attracted hundreds of thousands of followers.
This book thoroughly explains how computers work. It starts by fully examining a NAND gate, then goes on to build every piece and part of a small, fully operational computer. The necessity and use of codes is presented in parallel with the apprioriate pieces of hardware. The book can be easily understood by anyone whether they have a technical background or not. It could be used as a textbook.
"All the world's a stage", William Shakespeare wrote, "And all the men and women merely players." Sit back as the curtain goes up on the dramas, sonnets, and life of one of the greatest writers in the English language. Shakespeare wrote or contributed to more than 40 plays, ranging from romantic comedies to the profound tragedy King Lear, as well as 154 sonnets. The Shakespeare Book has visual plot summaries of each one, with diagrams to show the intricate web of relationships in plays such as A Midsummer's Night Dream. Commentaries explain Shakespeare's sources and set each drama in context, revealing, for instance, how the warring Protestants and Catholics of his day are mirrored in Romeo and Juliet's Montagues and Capulets. Written in plain English and packed with graphics and illustrations, The Shakespeare Book illumines the Bard's world - his marriage, businesses, and friends - and explains how his works became an enduring phenomenon. Whether you need a guide through complex plots and unfamiliar language, or you're looking for a fresh perspective on his well-loved plays and sonnets, this indispensable guide will help you fully appreciate Shakespeare, the man, and the writer. Reviews: "Generous helpings of illustrations, time lines, plot diagrams, and character guides ensure that even readers in their 'salad days' will enjoy every dish at the Shakespearean feast." - Booklist "Enlightening" - YA Book Central "In this latest addition to the series, the Bard comes alive for young aficionados." - School Library Journal "Countless volumes have been written about William Shakespeare and his work, but here is a single volume that has organized his plays (and some of his sonnets) in exactly what the subtitle says: 'Big Ideas Simply Explained...a must-have.'" - VOYA magazine
Understand the Significance of Symbols in Your Design Work Our world is comprised of a handful of very simple patterns that have been a part of human design since the beginning of time and have eternal significance. Decoding Design reveals how common symbols and shapes - like circles, squares and triangles - resonate at a gut level and can lend greater meaning to a design. By deconstructing famous logos and other sample designs, you'll learn how to communicate complex information quickly and intuitively with universal and meaningful patterns. You'll also uncover how other disciplines, such as philosophy, math, and physics, influence great design and can help you present ideas in a holistic and compelling manner. Whether you're a designer, student, or marketing professional, Decoding Design will show you the deeper meaning behind the symbols you encounter everyday, and how to better use those symbols to create an impactful relationship with the viewer.
Why Things That Go Right Sometimes Go Wrong
Decoding Gardening Advice
From Beginner to Pro
Move beyond basic programming and construct reliable and efficient software with complex code
An Introduction to Creative Problem Solving
Designing Interfaces
The Science Behind the 100 Most Common Recommendations

Provides information on designing easy-to-use interfaces.
Embedded systems are today, widely deployed in just about every piece of machinery from toasters to spacecraft. Embedded system designers face many challenges. They are asked to produce increasingly complex systems using the latest technologies, but these technologies are changing faster than ever. They are asked to produce better quality designs with a shorter time-to-market. They are asked to implement increasingly complex functionality but more importantly to satisfy numerous other constraints. To achieve the current goals of design, the designer must be aware with such design constraints and more importantly, the factors that have a direct effect on them. One of the challenges facing embedded system designers is the selection of the optimum processor for the application in hand; single-purpose, general-purpose or application specific. Microcontrollers are one member of the family of the application specific processors. The book concentrates on the use of microcontroller as the embedded system's processor, and how to use it in many embedded system applications. The book covers both the hardware and software aspects needed to design using microcontroller. The book is ideal for undergraduate students and also the engineers that are working in the field of digital system design.
Table of contents
James Lowen narrates a year-long quest to see Britain's rarest and more remarkable moths. Although mostly unseen by us, moths are everywhere. And their capacity to delight astounds. Inspired by a revelatory encounter with a Poplar Hawk-moth – a huge, velvety-winged wonder wrapped in silver –
James Lowen embarks on a year-long quest to celebrate the joy of Britain's rarest and most remarkable moths. By hiking up mountains, wading through marshes and roaming by night amid ancient woodlands, James follows the trails of both Victorian collectors and present-day conservationists. Seeking to understand why they and many ordinary folk love what the general public purports to hate, his investigations reveal a heady world of criminality and controversy, derring-do and determination. From Cornwall to the Cairngorms, James explores British landscapes to coax these much-maligned creatures out from the cover of darkness and into the light. Moths are revealed to be attractive, astonishing and approachable; capable of migratory feats and camouflage mastery, moths have much to tell us on the state of the nation's wild and not-so-wild habitats. As a counterweight to his travels, James and his young daughter track the seasons through a kaleidoscope of moth species living innocently yet covertly in their suburban garden. Without even leaving home, they bond over a shared joy in the uncommon beauty of common creatures, for perhaps the greatest virtue of moths, we learn, is their accessibility. Moths may be everywhere, but above all, they are here. Quite unexpectedly, no animals may be better placed to inspire the environmentalists of the future.
A year intoxicated by Britain's rare and remarkable moths
Better Productivity Through Collaboration
The Hardware Software Interface
The Basic Principles of Computers for Everyone
Computer Organization and Design RISC-V Edition
Eh
Hacking- The art Of Exploitation

Object-oriented analysis and design (OOAD) has over the years, become a vast field, encompassing such diverse topics as design process and principles, documentation tools, refactoring, and design and architectural patterns. For most students the learning experience is incomplete without implementation. This new textbook provides a comprehensive introduction to OOAD. The salient points of its coverage are: • A sound footing on object-oriented concepts such as classes, objects, interfaces, inheritance, polymorphism, dynamic linking, etc. • A good introduction to the stage of requirements analysis. • Use of UML to document user requirements and design. • An extensive treatment of the design process. • Coverage of implementation issues. • Appropriate use of design and architectural patterns. • Introduction to the art and craft of refactoring. • Pointers to resources that further the reader's knowledge. All the main case-studies used for this book have been implemented by the authors using Java. The text is liberally peppered with snippets of code, which are short and fairly self-explanatory and easy to read. Familiarity with a Java-like syntax and a broad understanding of the structure of Java would be helpful in using the book to its full potential.
This is a practical book for computer engineers who want to understand or implement hardware/software systems. It focuses on problems that require one to combine hardware design with software design – such problems can be solved with hardware/software codesign. When used properly, hardware/software co- sign works better than hardware design or software design alone: it can improve the overall performance of digital systems, and it can shorten their design time. Hardware/software codesign can help a designer to make trade-offs between the ?exibility and the performanceof a digital system. To achieve this, a designer needs to combine two radically different ways of design: the sequential way of dec- position in time, using software, with the parallel way of decomposition in space, using hardware. Intended Audience This book assumes that you have a basic understandingof hardware that you are - miliar with standard digital hardware componentssuch as registers, logic gates, and components such as multiplexers and arithmetic operators. The book also assumes that you know how to write a program in C. These topics are usually covered in an introductory course on computer engineering or in a combination of courses on digital design and software engineering.
Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. "Refactoring" shows users exactly how to spot the best opportunities for refactoring and exactly how to do it, step by step.

Software Design for Flexibility
The Science Behind Why We Buy
Improving the Design of Existing Code
Process, Principles and Techniques
The HSC Course : Teacher Resource Kit
The Etymologicon