

## Building Evolutionary Architectures

Whether you're building a social media site or an internal-use enterprise application, this hands-on guide shows you the connection between MongoDB and the business problems it's designed to solve. You'll learn how to apply MongoDB design patterns to several challenging domains, such as ecommerce, content management, and online gaming. JavaScript code examples, you'll discover how MongoDB lets you scale your data model while simplifying the development process. Many businesses launch NoSQL databases without understanding the techniques for using their features most effectively. This book demonstrates the benefits of document embedding, polymorphic schemas, and other patterns for tackling specific big data use cases, including: Operational intelligence: Perform real-time analytics of business data Ecommerce: Use MongoDB as a product catalog master or inventory management system Content management: Learn methods for storing content nodes, binary assets, and discussions Online advertising network: Frequency capping ad impressions, and keyword targeting and bidding Social networking: Learn how to store a complex social graph, modeled after Google+ Online gaming: Provide concurrent access to character and world data for a multiplayer role-playing game

EDGE: The Agile Operating Model That Will Help You Successfully Execute Your Digital Transformation "[The authors'] passion for technology allows them to recognize that for most enterprises in the 21st century, technology is THE business. This is what really separates the EDGE approach. It is a comprehensive operating model with techno- The Foreword by Heidi Musser, Executive Vice President and Principal Consultant, Leading Agile; retired, Vice President and CIO, USAA Maximum innovation happens at the edge of chaos: the messy, risky, and uncertain threshold between randomness and structure. Operating there is uncomfortable but it's where organizations "invent the future." fast, iterative, adaptive, lightweight, and value-driven tools to achieve digital transformation, and EDGE: Value-Driven Digital Transformation is your guide to using this operating model for innovation. Jim Highsmith is one of the world's leading agile pioneers and a coauthor of the Agile Manifesto. He, Linda Liu, and David Robinson know from experience that sustainable digital transformation requires far more than adopting isolated agile practices or conventional portfolio management. This hard, indispensable work involves changing culture and mindset, and going beyond transforming the IT department. EDGE embraces an adaptive mindset in the face of market uncertainty, a vibrant portfolio approach that encourages continual value linkages from vision to detailed initiatives, incremental funding that shifts as strategies evolve, collaborative decision-making, and better risk mitigation. This guide shows leaders how to use the breakthrough EDGE approach to go beyond incremental improvement in a world of exponential organization that adapts fast enough to thrive. Clear away unnecessary governance processes, obsolete "command and control" leadership approaches, and slow budgeting/planning cycles. Improve collaboration when major, fast-paced responses are necessary. Continually optimize investment allocation and monitoring based on your vision and product for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Building Evolutionary ArchitecturesSupport Constant Change"O'Reilly Media, Inc." Continuous Architecture provides a broad architectural perspective for continuous delivery, and describes a new architectural approach that supports and enables it. As the pace of innovation and software releases increases, IT departments are tasked to deliver value quickly and inexpensively to their business partners. With a focus on getting hands faster, the ultimate goal of daily software updates is in sight to allow teams to ensure that they can release every change to the system simply and efficiently. This book presents an architectural approach to support modern application delivery methods and provide a broader architectural perspective, taking architectural concerns or continuous delivery approaches. The authors explain how to solve the challenges of implementing continuous delivery at the project and enterprise level, and the impact on IT processes including application testing, software deployment and software architecture. Covering the application of enterprise and software architecture concepts to Delivery models Explains how to create an architecture that can evolve with applications Incorporates techniques including refactoring, architectural analysis, testing, and feedback-driven development Provides insight into incorporating modern software development when structuring teams and organizations

Don't engineer by coincidence-design it like you mean it! Filled with practical techniques, Design IT! is the perfect introduction to software architecture for programmers who are ready to grow their design skills. Lead your team as a software architect, ask the right stakeholders the right questions, explore design options, and help your team promotes the right "ilities. Share your design decisions, facilitate collaborative design workshops that are fast, effective, and fun-and develop more awesome software! With dozens of design methods, examples, and practical know-how, Design IT! shows you how to become a software architect. Walk through the core concepts every architect should apply them, and learn a variety of skills that will make you a better programmer, leader, and designer. Uncover the big ideas behind software architecture and gain confidence working on projects big and small. Plan, design, implement, and evaluate software architectures and collaborate with your team, stakeholders, and other architects. Learn to understand their needs, dig for architecturally significant requirements, write amazing quality attribute scenarios, and make confident decisions. Choose technologies based on their architectural impact, facilitate architecture-centric design workshops, and evaluate architectures using lightweight, effective methods. Write lean architectural documents that are easy to read. Run an architecture design studio, implement the architecture you've designed, and grow your team's architectural knowledge. Good design requires good communication. Talk about your software architecture with stakeholders using whiteboards, documents, and code, and apply architecture-focused design methods in your day-to-day work.

Designing Software Architectures will teach you how to design any software architecture in a systematic, predictable, repeatable, and cost-effective way. This book introduces a practical methodology for architecture design that any professional software engineer can use, provides structured methods supported by reusable chunks of design patterns, and case studies that demonstrate how to use the methods. Using realistic examples, you'll master the powerful new version of the proven Attribute-Driven Design (ADD) 3.0 method and will learn how to use it to address key drivers, including quality attributes, such as modifiability, usability, and availability, along with functional requirements and non-functional requirements.

Drawing on their extensive experience, Humberto Cervantes and Rick Kazman guide you through crafting practical designs that support the full software life cycle, from requirements to maintenance and evolution. You'll learn how to successfully integrate design in your organizational context, and how to design systems that will be built well over time. Comprehensive coverage includes Understanding what architecture design involves, and where it fits in the full software development life cycle Mastering core design concepts, principles, and processes Understanding how to perform the steps of the ADD method Scaling design and analysis up or down, including design for pre-sale processes

Reviews Recognizing and optimizing critical relationships between analysis and design Utilizing proven, reusable design primitives and adapting them to specific problems and contexts Solving design problems in new domains, such as cloud, mobile, or big data

Often referred to as the "black art" because of its complexity and uncertainty, software estimation is not as difficult or puzzling as people think. In fact, generating accurate estimates is straightforward—once you understand the art of creating them. In his highly anticipated book, acclaimed author Steve McConnell unravels the mystery of software estimation—distilling academic information and real-world experience into a practical guide for working software professionals. Instead of arcane treatises and rigid modeling techniques, this guide highlights a proven set of procedures, understandable formulas, and heuristics that individuals and development teams can apply to their projects. Discover how to: Estimate schedule and cost—or estimate the functionality that can be delivered within a given time frame Avoid common software estimation mistakes Learn estimation techniques for you, your team, and your organization \* Estimate specific project activities—including development, management, and defect correction

Approaches to any type of project—small or large, agile or traditional Navigate the shark-infested political waters that surround project estimates When many corporate software projects are failing, McConnell shows you what works for successful software estimation. People have been finding inspiration in nature in solving their problems, from the very beginning of their existence. In the most general sense, biomimicry, defined as "inspire from the nature," has brought together the engineers and designers nowadays. This collaboration creates innovative and creative outcomes that encourage people with their own ideas. Relationships. Accordingly, the aim of this book is to bring together different works or developments on biomimetics in interdisciplinary relationship between different areas, especially biomimicry, engineering, and design. The twenty-first century has conceived many new and amazing designs. The book in your hands will surely be an important one in the future possibilities.

[Domain-driven Design](#)

[EDGE](#)

[Design IT!](#)

[Redefining the Architect's Role in the Digital Enterprise](#)

[Evolutionary and Neo-Schumpeterian Approaches to Economics](#)

[Pricing Beauty](#)

[The War of the Worlds](#)

[Evolutionary Patterns to Transform Your Monolith](#)

[Welcome to the Desert of the Real](#)

[Become a successful software architect by implementing effective architecture concepts](#)

[Working With Stakeholders Using Viewpoints and Perspectives](#)

[Demystifying the Black Art](#)

[Software Systems Architecture](#)

[From Programmer to Software Architect](#)

Describes ways to incorporate domain modeling into software development.

This is a practical guide for software developers, and different than other software architecture books. Here's why: It teaches risk-driven architecting. There is no need for meticulous designs when risks are small, nor any excuse for sloppy designs when risks threaten your success. This book describes a way to do just enough architecture. It avoids the one-size-fits-all process tar pit with advice on how to tune your design effort based on the risks you face. It democratizes architecture. This book seeks to make architecture relevant to all software developers. Developers need to understand how to use constraints as guidelines that ensure desired outcomes, and how seemingly small changes can affect a system's properties. It cultivates declarative knowledge. There is a difference between being able to hit a ball and knowing why you are able to hit it, what psychologists refer to as procedural knowledge versus declarative knowledge. This book will make you more aware of what you have been doing and provide names for the concepts. It emphasizes the engineering. This book focuses on the technical parts of software development and what developers do to ensure the system works not titles or processes. It shows you how to build models and analyze architectures so that you can make principled design tradeoffs. It describes the techniques software designers use to reason about medium to large sized problems and points out where you can learn specialized techniques in more detail. It provides practical advice. Software design decisions influence the architecture and vice versa. The approach in this book embraces drill-down/pop-up behavior by describing models that have various levels of abstraction, from architecture to data structure design.

This book takes a humorous slant on the programming practice manual by reversing the usual approach: under the pretence of teaching you how to become the world's worst programmer who generally causes chaos, the book teaches you how to avoid the kind of bad habits that introduce bugs or cause code contributions to be rejected. Why be a code monkey? Why be a chaos monkey? OK, so you want to become a terrible programmer. You want to write code that gets vigorously rejected in review. You look forward to reading feedback plastered in comments like "WTF????". Even better, you fantasize about your bug-ridden changes sneaking through and causing untold chaos in the codebase. You want to build a reputation as someone who writes crazy, messy, error-prone garbage that frustrates your colleagues. Bad Programming Practices 101 will help you achieve that goal a whole lot quicker by teaching you an array of bad habits that will allow you to cause maximum chaos. Alternatively, you could use this book to identify those bad habits and learn to avoid them. The bad practices are organized into topics that form the basis of programming (layout, variables, loops, modules, and so on). It's been remarked that to become a good programmer, you must first write 10,000 lines of bad code to get it all out of your system. This book is aimed at programmers who have so far written only a small portion of that. By learning about poor programming habits, you will learn good practices. In addition, you will find out the motivation behind each practice, so you can learn why it is considered good and not simply get a list of rules. What You'll Learn Become a better coder by learning how (not) to program Choose your tools wisely Think of programming as problem solving Discover the consequences of a program's appearance and overall structure Explain your use of variables in programs Avoid bad habits and common mistakes when using conditionals and loops See how poor error-handling makes for unstable programs Sidestep bad practices related specifically to object-oriented programming Mitigate the effects of ineffectual and inadequate bug location and testing Who This Book Is For Those who have some practical programming knowledge (can program in at least one programming language), but little or no professional experience, which they would like to quickly build up. They are either still undergoing training in software development, or are at the beginning of their programming career. They have at most 1-2 years of professional experience.

Herbert Simon's classic work on artificial intelligence in the expanded and updated third edition from 1996, with a new introduction by John E. Laird. Herbert Simon's classic and influential The Sciences of the Artificial declares definitively that there can be a science not only of natural phenomena but also of what is artificial. Exploring the commonalities of artificial systems, including economic systems, the business firm, artificial intelligence, complex engineering projects, and social plans, Simon argues that designed systems are a valid field of study, and he proposes a science of design. For this third edition, originally published in 1978, Simon added new material that takes into account advances in cognitive psychology and the science of design while confirming and extending the book's basic thesis: that a physical symbol system has the necessary and sufficient means for intelligent action. Simon won the Nobel Prize for Economics in 1978 for his research into the decision-making process within economic organizations and the Turing Award (considered by some the computer science equivalent to the Nobel) with Allen Newell in 1975 for contributions to artificial intelligence, the psychology of human cognition, and list processing. The Sciences of the Artificial distills the essence of Simon's thought accessibly and coherently. This release of the third edition makes a pioneering work available to a new audience.

Domain-Driven Design (DDD) software modeling delivers powerful results in practice, not just in theory, which is why developers worldwide are rapidly moving to adopt it. Now, for the first time, there's an accessible guide to the basics of DDD: What it is, what problems it solves, how it works, and how to quickly gain value from it. Concise, readable, and actionable, Domain-Driven Design Distilled never buries you in detail-it focuses on what you need to know to get started. Vaughn Vernon, author of the best-selling Implementing Domain-Driven Design, draws on his twenty years of experience applying DDD principles to real-world situations. He is uniquely well-qualified to demystify its complexities, illuminate its subtleties, and help you solve the problems you might encounter. Vernon guides you through each core DDD technique for building better software. You'll learn how to segregate domain models using the powerful Bounded Contexts pattern, to develop a Ubiquitous Language within an explicitly bounded context, and to help domain experts and developers work together to create that language. Vernon shows how to use Subdomains to handle legacy systems and to integrate multiple Bounded Contexts to define both team relationships and technical mechanisms. Domain-Driven Design Distilled brings DDD to life. Whether you're a developer, architect, analyst, consultant, or customer, Vernon helps you truly understand it so you can benefit from its remarkable power. Coverage includes What DDD can do for you and your organization—and why it's so important. The cornerstones of strategic design with DDD: Bounded Contexts and Ubiquitous Language Strategic design with Subdomains Context Mapping: helping teams work together and integrate software more strategically Tactical design with Aggregates and Domain Events Using project acceleration and management tools to establish and maintain team cadence

Presents essays exploring the philosophical themes of the motion picture "The Matrix," which portrays a false world created from nothing but perceptions.

The Comprehensive, Proven Approach to IT Scalability—Updated with New Strategies, Technologies, and Case Studies In The Art of Scalability, Second Edition, leading scalability consultants Martin L. Abbott and Michael T. Fisher cover everything you need to know to smoothly scale products and services for any requirement. This extensively revised edition reflects new technologies, strategies, and lessons, as well as new case studies from the authors' pioneering consulting practice, AKF Partners. Writing for technical and nontechnical decision-makers, Abbott and Fisher cover everything that impacts scalability, including architecture, process, people, organization, and technology. Their insights and recommendations reflect more than thirty years of experience at companies ranging from eBay to Visa, and from Salesforce.com to Apple. You'll find updated strategies for structuring organizations to maximize agility and scalability, as well as new insights into the cloud (IaaS/PaaS) transition, NoSQL, DevOps, business metrics, and more. Using this guide's tools and advice, you can systematically clear away obstacles to scalability—and achieve unprecedented IT and business performance. Coverage includes • Why scalability problems start with organizations and people, not technology, and what to do about it • Actionable lessons from real successes and failures • Staffing, structuring, and leading the agile, scalable organization • Scaling processes for hyper-growth environments • Architecting scalability: proprietary models for clarifying needs and making choices—including 15 key success principles • Emerging technologies and challenges: data cost, datacenter planning, cloud evolution, and customer-aligned monitoring • Measuring availability, capacity, load, and performance

This accessible and engaging text is the first to offer a comprehensive critical history and analysis of the greening of architecture through accumulative reduction of negative environmental effects caused by buildings, urban designs and settlements. Describing the progressive development of green architecture from 1960 to 2010, it illustrates how it is ever evolving and ameliorated through alterations in form, technology, materials and use and it examines different places worldwide that represent a diversity of cultural and climatic contexts.

[Software Architecture in Practice](#)

[Designing Fine-Grained Systems](#)

[The Art of Scalability](#)

[The Matrix and Philosophy](#)

[MongoDB Applied Design Patterns](#)

[Building Evolutionary Architectures](#)

[Refactoring Databases](#)

[Lessons of Love, Strength, and What Makes a Family](#)

[Presentation Patterns](#)

[Domain-Driven Design Distilled](#)

[An Engineering Approach](#)

[The Greening of Architecture](#)

[Building Microservices](#)

[Become a Better Coder by Learning How \(Not\) to Program](#)

Computer organization and architecture is becoming an increasingly important core subject in the areas of computer science and its applications, and information technology constantly steers the relentless revolution going on in this discipline. This textbook demystifies the state of the art using a simple and step-by-step development from traditional fundamentals to the most advanced concepts entwined with this subject. The book presents the basic theoretical principles, numerous design approaches, and their actual practical implementations. Being driven by the diversified knowledge gained directly from working in the constantly changing environment of the information technology (IT) industry, the author sets the stage by describing the modern issues in different areas of this subject. He then continues to effectively provide a comprehensive source of material with exciting new developments using a wealth of concrete examples related to recent regulatory changes in the modern design and architecture of different categories of computer systems associated with real-life instances as case studies, ranging from micro to mini, supermini, mainframes, cluster architectures, massively parallel processing (MPP) systems, and even supercomputers with commodity processors. Many of the topics that are briefly discussed in this book to conserve space for new materials are elaborately described from the design perspective to their ultimate practical implementations with representative schematic diagrams available on the book's website. Key Features Microprocessor evolutions and their chronological improvements with illustrations taken from Intel, Motorola, and other leading families Multicore concept and subsequent multicore processors, a new standard in processor design Cluster architecture, a vibrant organizational and architectural development in building up massively distributed/parallel systems InfiniBand, a high-speed link for use in cluster system architecture providing a single-system image FireWire, a high-speed serial bus used for both isochronous real-time data transfer and asynchronous applications, especially needed in multimedia and mobile phones Evolution of embedded systems and their specific characteristics Real-time systems and their major design issues in brief Improved main memory technologies with their recent releases of DDR2, DDR3, Rambus DRAM, and Cache DRAM, widely used in all types of modern systems, including large clusters and high-end servers DVD optical disks and flash drives (pen drives) RAID, a common approach to configuring multiple-disk arrangements used in large server-based systems A good number of problems along with their solutions on different topics after their delivery Exhaustive material with respective figures related to the entire text to illustrate many of the computer design, organization, and architecture issues with examples are available online at <http://crpress.com/9780367255732>

The book serves as a textbook for graduate-level courses for computer science engineering, information technology, electrical engineering, electronics engineering, computer science, BCA, MCA, and other similar courses.

A resounding testament to individuality and the power of family in all forms from the young man who "lit up the Internet" (Ellen DeGeneres) On January 31, 2011, Zach Wahls addressed the Iowa House Judiciary Committee in a public forum regarding full marriage equality. The nineteen-year-old son of a same-sex couple, Wahls proudly proclaimed, "The sexual orientation of my parents has had zero effect on the content of my character." Hours later, his speech was posted on YouTube, where it went viral, quickly receiving more than two million views. By the end of the week, everyone knew his name and wanted to hear more from the boy with two moms. Same-sex marriage will be a major—possibly the defining—issue in this year's election cycle, and Wahls speaks to that, but also to a broader issue. Sure, he's handsome and athletic, an environmental engineering student, and an Eagle Scout. Yet, growing up with two moms, he knows what it's like to feel different and to fear being made fun of or worse. In the inspirational spirit of It Gets Better edited by Dan Savage and Terry Miller, My Two Moms also delivers a reassuring message to same-sex couples, their kids, and anyone who's ever felt like an outsider: "You are not alone."

This book uncovers the guiding principles behind Tsui's evolutionary approach to explore the many design lessons that can be learned from nature and share the impressive results of their application to architectural projects.

As the development ecosystem is constantly changing, providing a constant stream of new tools, frameworks, techniques, and paradigms. Over the past few years, incremental developments in core engineering practices for software development have created the foundations for rethinking how architecture changes over time, along with ways to protect important architectural characteristics as it evolves. This practical guide ties those parts together with a new way to think about architecture and time.

not gentle to the capitalists" (Schumpeter, 1919). Thus, by instead portraying the conflict between entrepreneurial activity and the sociology of the modern state, he came quite close to the analysis carried out by Thorstein Veblen some decades earlier, who emphasized the conflict between p-ressive technology and the institutions of a contemporary "predatory dynamic State of early modern times, superficially altered by a suffusion of democratic and parliamentary institutions" (Veblen, 1964, p. 398). Modern neo-Schumpeterian approaches have continued to build on this groundwork provided by their master. During recent years there has been a great upsurge of discussion on technology, innovations, technological regimes, etc. from the dynamic perspective provided by Schumpeter (Dosi, 1984, Rosegger, 1985; Dosi et al., 1988). Thus the search process for (i- poral) extra profits has been stressed and has been used for modelling attempts. The wider institutional framework for technological change and innovation activity has also been strongly developed more recently. Hence emphasis has grown in the study of technological and industrial regimes, path dependency, and the network approach, developed recently, that social relationships structure the opportunities and constraints that face firms and agents that, for example, carry out innovations (Snehota, 1990).

Annotation: Over the past 10 years, distributed systems have become more fine-grained. From the large multi-million line long monolithic applications, we are now seeing the benefits of smaller self-contained services. Rather than heavy-weight, hard to change Service Oriented Architectures, we are now seeing systems consisting of collaborating microservices. Easier to change, deploy, and if required rewire, organizations which are in the right position to take advantage of them are yielding significant benefits. This book takes an holistic view of the things you need to be cognizant of in order to pull this off. It covers just enough understanding of technology, architecture, operations and organization to show you how to move towards finer-grained systems.

When a meteorite lands in Surrey, the locals don't know what to make of it. But as Martians emerge and begin killing bystanders, it quickly becomes clear—England is under attack. Armed soldiers converge on the scene to ward off the invaders, but meanwhile, more Martian cylinders land on Earth, bringing reinforcements. As war breaks out across England, the locals must fight for their lives, but life on Earth will never be the same. This is an unabridged version of one of the first fictional accounts of extraterrestrial invasion. H. G. Wells's military science fiction novel was first published in book form in 1898, and is considered a classic of English literature.

Discover how different software architectural models can help you solve problems, and learn best practices for the software development cycle Key Features Learn concepts related to software architecture and embrace them using the latest features of Spring 5 Discover architectural models and learn when to apply them Gain knowledge of architectural principles and how they can be used to provide accountability and rationale for architectural decisions Book Description Spring 5 and its ecosystem can be used to build robust architectures effectively. Software architecture is the underlying piece that helps us accomplish our business goals whilst supporting the features that a product demands. This book explains in detail how to choose the right architecture and apply best practices during your software development cycle to avoid technical debt and support evolution. The book covers the right architecture model to support your business requirements is one of the key decisions you need to take when a new product is being created from scratch or is being refactored to support new business demands. This book gives you insights into the most common architectural models and guides you when and where they can be used. During this journey, you'll see cutting-edge technologies surrounding the Spring products, and understand how to use agile techniques such as DevOps and continuous delivery to take your software to production effectively. By the end of this book, you'll not only know the ins and outs of Spring, but also be able to make critical design decisions that surpass your clients' expectations. What you will learn Understand the key principles of software architecture Uncover the most common architectural models available Analyze scenarios where an architecture model should be used Implement agile techniques to take your software to production Secure the products you are working on Master tricks that will help you build high-performance applications Use cutting-edge technologies to build products Who this book is for If you're an experienced Spring developer aspiring to become an architect of enterprise-grade applications, this book is for you. It's also ideal for software architects who want to leverage Spring to create effective application blueprints.

[A Guide for Agile Teams](#)

[Bad Programming Practices 101](#)

[The Software Architect Elevator](#)

[Nature as a Basis for Design](#)

[Evolutionary Architecture](#)

[Computer Organisation and Architecture](#)

[Evolutionary Concepts, Principles, and Designs](#)

[Software Engineering at Google](#)

[A Critical History and Survey of Contemporary Sustainable Architecture and Urban Design](#)

[Meta-heuristic and Evolutionary Algorithms for Engineering Optimization](#)

[Evolutionary Database Design \(paperback\)](#)

[Lessons Learned from Programming Over Time](#)

[Sustainable Architecture in an Agile and Cloud-Centric World](#)

[My Two Moms](#)

This is the eagerly-anticipated revision to one of the seminal books in the field of software architecture which clearly defines and explains the topic.

Today, software engineers need to know not only how to program effectively but also how to develop proper engineering practices to make their codebase sustainable and healthy. This book emphasizes this difference between programming and software engineering. How can software engineers manage a living codebase that evolves and responds to changing requirements and demands over the length of its life? Based on their experience at Google, software engineers Titus Winters and Hyrum Wright, along with technical writer Tom Manshreck, present a candid and insightful look at how some of the world's leading practitioners construct and maintain software. This book covers Google's unique engineering culture, processes, and tools and how these aspects contribute to the effectiveness of an engineering organization. You'll explore three fundamental principles that software organizations should keep in mind when designing, architecting, writing, and maintaining code: How time affects the sustainability of software and how to make your code resilient over time How scale affects the viability of software practices within an engineering organization What trade-offs a typical engineer needs to make when evaluating design and development decisions

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

Presentation Patterns is the first book on presentations that categorizes and organizes the building blocks (or patterns) that you'll need to communicate effectively using presentation tools like Keynote and PowerPoint. Patterns are like the lower-level steps found inside recipes; they are the techniques you must master to be considered a master chef or master presenter. You can use the patterns in this book to construct your own recipes for different contexts, such as business meetings, technical demonstrations, scientific expositions, and keynotes, just to name a few. Although there are no such things as antirecipes, this book shows you lots of antipatterns!things you should avoid doing in presentations. Modern presentation tools often encourage ineffective presentation techniques, but this book shows you how to avoid them. Each pattern is introduced with a memorable name, a definition, and a brief explanation of motivation. Readers learn where the pattern applies, the consequences of applying it, and how to apply it. The authors also identify critical antipatterns: clichés, fallacies, and design mistakes that cause presentations to disappoint. These problems are easy to avoid once you know how. Presentation Patterns will help you Plan what you'll say, who you'll say it to, how long you'll talk, and where you'll present Perfectly calibrate your presentation to your audience Use the storyteller's [narrative arc] to full advantage Strengthen your credibility,and avoid mistakes that hurt it Hone your message before you ever touch presentation software Incorporate visuals that support your message instead of hindering it Create highly effective [infodecks]: that work when you're not able to deliver a talk in person Construct slides that really communicate and avoid [Ant Fonts],[Floodmarks],[Alienating Artifacts], and other errors Master 13 powerful techniques for delivering your presentation with power, authority, and clarity Whether you use this book as a handy reference or read it from start to finish, it will be a revelation: an entirely new language for systematically planning, creating, and delivering more powerful presentations. You'll quickly find it indispensable!no matter what you're presenting, who your audiences are, or what message you're driving home.

Evolutionary architecture attempts to evolve form and structure in emulation of the evolutionary processes of nature. It considers architecture as a form of artificial life. This approach has formed the basis for the author's teaching programme for AA Diploma Unit II.

As the digital economy changes the rules of the game for enterprises, the role of software and IT architects is also transforming. Rather than focus on technical decisions alone, architects and senior technologists need to combine organizational and technical knowledge to effect change in their company's structure and processes. To accomplish that, they need to connect the IT engine room to the penthouse, where the business strategy is defined. In this guide, author Gregor Hohpe shares real-world advice and hard-learned lessons from actual IT transformations. His anecdotes help architects, senior developers, and other IT professionals prepare for a more complex but rewarding role in the enterprise. This book is ideal for:

Software architects and senior developers looking to shape the company's technology direction or assist in an organizational transformation Enterprise architects and senior technologists searching for practical advice on how to navigate technical and organizational topics CTOs and senior technical architects who are devising an IT strategy that impacts the way the organization works IT managers who want to learn what's worked and what hasn't in large-scale transformation

Salary surveys worldwide regularly place software architect in the top 10 best jobs, yet no real guide exists to help developers become architects. Until now. This book provides the first comprehensive overview of software architecture's many aspects. Aspiring and existing architects alike will examine architectural characteristics, architectural patterns, component determination, diagramming and presenting architecture, evolutionary architecture, and many other topics. Mark Richards and Neal Ford/hands-on practitioners who have taught software architecture classes professionally for years!focus on architecture principles that apply across all technology stacks. You'll explore software architecture in a modern light, taking into account all the innovations of the past decade. This book examines: Architecture patterns: The technical basis for many architectural decisions Components: Identification, coupling, cohesion, partitioning, and granularity Soft skills: Effective team management, meetings, negotiation, presentations, and more Modernity: Engineering practices and operational approaches that have changed radically in the past few years Architecture as an engineering discipline: Repeatable results, metrics, and concrete valuations that add rigor to software architecture

A book for Tech Leads, from Tech Leads. Discover how more than 35 Tech Leads find the delicate balance between the technical and non-technical worlds. Discover the challenges a Tech Lead faces and how to overcome them. You may be surprised by the lessons they have to share.

[From Novices to Practitioners](#)

[The Sciences of the Artificial](#)

[An Evolutionary Architecture](#)

[Design and architect highly scalable, robust, and high-performance Java applications](#)

[Interdisciplinary Expansions in Engineering and Design With the Power of Biomimicry](#)

[The Retrospective Handbook](#)

[Talking with Tech Leads](#)

[Continuous Architecture](#)

[Fundamentals of Software Architecture](#)

[A Risk-Driven Approach](#)

[Tackling Complexity in the Heart of Software](#)

[Software Architecture with Spring 5.0](#)

[Value-Driven Digital Transformation](#)

How do you detangle a monolithic system and migrate it to a microservice architecture? How do you do it while maintaining business-as-usual? As a companion to Sam Newman's extremely popular Building Microservices, this new book details a proven method for transitioning an existing monolithic system to a microservice architecture. With many illustrative examples, insightful migration patterns, and a bevy of practical advice to transition your monolith enterprise into a microservice operation, this practical guide covers multiple scenarios and strategies for a successful migration, from initial planning all the way through application and database decomposition. You'll learn several tried and tested patterns and techniques that you can use as you migrate your existing architecture. Ideal for organizations looking to transition to microservices, rather than rebuild Helps companies determine whether to migrate, when to migrate, and where to begin Addresses communication, integration, and the migration of legacy systems Discusses multiple migration patterns and where they apply Provides database migration examples, along with synchronization strategies Explores application decomposition, including several architectural refactoring patterns Delves into details of database decomposition, including the impact of breaking referential and transactional integrity, new failure modes, and more

Overview of optimization -- Introduction to meta-heuristic and evolutionary algorithms -- Pattern search (PS) -- Genetic algorithm (GA) -- Simulated annealing (SA) -- Tabu search (TS) -- Ant colony optimization (ACO) -- Particle swarm optimization (PSO) -- Differential evolution (DE) -- Harmony search (HS) -- Shuffled frog-leaping algorithm (SFLA) -- Honey-bee mating optimization (HBMO) -- Invasive weed optimization (IWO) -- Central force optimization (CFO) -- Biogeography-based optimization (BBO) -- Firefly algorithm (FA) -- Gravity search algorithm (GSA) -- Bat algorithm (BA) -- Plant propagation algorithm (PPA) -- Water cycle algorithm (WCA) -- Symbiotic organisms search (SOS) -- Comprehensive evolutionary algorithm (CEA)

Software Systems Architecture is a practitioner-oriented guide to designing and implementing effective architectures for information systems. It is both a readily accessible introduction to software architecture and an invaluable handbook of well-established best practices. It shows why the role of the architect is central to any successful information-systems development project, and, by presenting a set of architectural viewpoints and perspectives, provides specific direction for improving your own and your organization's approach to software systems architecture. With this book you will learn how to Design an architecture that reflects and balances the different needs of its stakeholders Communicate the architecture to stakeholders and demonstrate that it has met their requirements Focus on architecturally significant aspects of design, including frequently overlooked areas such as performance, resilience, and location Use scenarios and patterns to drive the creation and validation of your architecture Document your architecture as a set of related views Use perspectives to ensure that your architecture exhibits important qualities such as performance, scalability, and security The architectural viewpoints and perspectives presented in the book also provide a valuable long-term reference source for new and experienced architects alike. Whether you are an aspiring or practicing software architect, you will find yourself referring repeatedly to the practical advice in this book throughout the lifecycle of your projects. A supporting Web site containing further information can be found at [www.viewpoints-and-perspectives.info](http://www.viewpoints-and-perspectives.info)

A comprehensive guide to exploring software architecture concepts and implementing best practices Key Features Enhance your skills to grow your career as a software architect Design efficient software architectures using patterns and best practices Learn how software architecture relates to an organization as well as software development methodology Book Description The Software Architect's Handbook is a comprehensive guide to help developers, architects, and senior programmers advance their career in the software architecture domain. This book takes you through all the important concepts, right from design principles to different considerations at various stages of your career in software architecture. The book begins by covering the fundamentals, benefits, and purpose of software architecture. You

will discover how software architecture relates to an organization, followed by identifying its significant quality attributes. Once you have covered the basics, you will explore design patterns, best practices, and paradigms for efficient software development. The book discusses which factors you need to consider for performance and security enhancements. You will learn to write documentation for your architectures and make appropriate decisions when considering DevOps. In addition to this, you will explore how to design legacy applications before understanding how to create software architectures that evolve as the market, business requirements, frameworks, tools, and best practices change over time. By the end of this book, you will not only have studied software architecture concepts but also built the soft skills necessary to grow in this field. What you will learn Design software architectures using patterns and best practices Explore the different considerations for designing software architecture Discover what it takes to continuously improve as a software architect Create loosely coupled systems that can support change Understand DevOps and how it affects software architecture Integrate, refactor, and re-architect legacy applications Who this book is for The Software Architect's Handbook is for you if you are a software architect, chief technical officer (CTO), or senior developer looking to gain a firm grasp of software architecture.

Technologists who want their ideas heard, understood, and funded are often told to speak the language of business—without really knowing what that is. This book's toolkit provides architects, product managers, technology managers, and executives with a shared language—in the form of repeatable, practical patterns and templates—to produce great technology strategies. Author Eben Hewitt developed 39 patterns over the course of a decade in his work as CTO, CIO, and chief architect for several global tech companies. With these proven tools, you can define, create, elaborate, refine, and communicate your architecture goals, plans, and approach in a way that executives can readily understand, approve, and execute. This book covers: Architecture and strategy: Adopt a strategic architectural mindset to make a meaningful material impact Creating your strategy: Define the components of your technology strategy using proven patterns Communicating the strategy: Convey your technology strategy in a compelling way to a variety of audiences Bringing it all together: Employ patterns individually or in clusters for specific problems: use the complete framework for a comprehensive strategy

Are you running retrospectives regularly? Perhaps you run retrospectives once a week, or fortnightly. Do you feel like you could be getting more out of your retrospectives and fuelling continuous improvement in your teams? You may already find retrospectives valuable, but suspect there are ways of making them better. This book condenses down eight years of experience working with the retrospective practice within the context of real agile teams. It offers you practice advice on how to make your retrospectives even more effective including topics such as: Best methods to prepare for a retrospective Picking just the right materials Facilitating retrospectives with ease Dealing with common retrospective smells Retrospectives in different contexts including distributed, large and small groups A checklist for preparation Ensuring retrospectives result in change

Sociologist Ashley Mears takes us behind the brightly lit runways and glossy advertisements of the fashion industry in this insider's study of the world of modeling. Mears, who worked as a model in New York and London, draws on observations as well as extensive interviews with male and female models, agents, clients, photographers, stylists, and others, to explore the economics and politics—and the arbitrariness— behind the business of glamour. Exploring a largely hidden arena of cultural production, she shows how the right "look" is discovered, developed, and packaged to become a prized commodity. She examines how models sell themselves, how agents promote them, and how clients decide to hire them. An original contribution to the sociology of work in the new cultural economy, Pricing Beauty offers rich, accessible analysis of the invisible ways in which gender, race, and class shape worth in the marketplace.

[Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise](#)

[Technology Strategy Patterns](#)

[A Practical Approach](#)

[Just Enough Software Architecture](#)

[Practical Use Cases with the Leading NoSQL Database](#)

[The Making of a Fashion Model](#)

[Software Architect's Handbook](#)

[Techniques for Crafting Better Presentations](#)

[Architecture as Strategy](#)

[Support Constant Change](#)

[Software Estimation](#)

[Monolith to Microservices](#)

[Designing Software Architectures](#)