

Software Architecture In Practice

Agile software development approaches have had significant impact on industrial software development practices. Today, agile software development has penetrated to most IT companies across the globe, with an intention to increase quality, productivity, and profitability. Comprehensive knowledge is needed to understand the architectural challenges involved in adopting and using agile approaches and industrial practices to deal with the development of large, architecturally challenging systems in an agile way. Agile Software Architecture focuses on gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox. Readers will learn how agile and architectural cultures can co-exist and support each other according to the context. Moreover, this book will also provide useful leads for future research in architecture and agile to bridge such gaps by developing appropriate approaches that incorporate architecturally sound practices in agile methods. Presents a consolidated view of the state-of-art and state-of-practice as well as the newest research findings Identifies gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox Explains whether or not and how agile and architectural cultures can co-exist and support each other depending upon the context Provides useful leads for future research in both architecture and agile to bridge such gaps by developing appropriate approaches, which incorporate architecturally sound practices in agile methods

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.

An understanding of psychology—specifically the psychology behind how users behave and interact with digital interfaces—is perhaps the single most valuable nondesign skill a designer can have. The most elegant design can fail if it forces users to conform to the design rather than working within the "blueprint" of how humans perceive and process the world around them. This practical guide explains how you can apply key principles in psychology to build products and experiences that are more intuitive and human-centered. Author Jon Yablonski deconstructs familiar apps and experiences to provide clear examples of how UX designers can build experiences that adapt to how users perceive and process digital interfaces. You’ll learn: How aesthetically pleasing design creates positive responses The principles from psychology most useful for designers How these psychology principles relate to UX heuristics Predictive models including Fitts’s law, Jakob’s law, and Hick’s law Ethical implications of using psychology in design A framework for applying these principles

The concepts and practice of software architecture are introduced--what a system is designed to do and how its components are meant to interact with each other. The authors cover not only essential technical topics for specifying and validating a system, but, for the first time, emphasize the importance of the business context in which large systems are designed.

Just Enough Software Architecture

Software Architecture in Practice

Design modern systems using effective architecture concepts, design patterns, and techniques with C++20

A Software Production Environment Based on Conceptual Modeling

Become a successful software architect by implementing effective architecture concepts

Software architecture in practice

Software Architecture for Product Families

From the creator of the popular website Ask a Manager and New York’s work-advice columnist comes a witty, practical guide to 200 difficult professional conversations—featuring all-new advice! There’s a reason Alison Green has been called “the Dear Abby of the work world.” Ten years as a workplace-advice columnist have taught her that people avoid awkward conversations in the office because they simply don’t know what to say. Thankfully, Green does—and in this incredibly helpful book, she tackles the tough discussions you may need to have during your career. You’ll learn what to say when • coworkers push their work on you—then take credit for it • you accidentally trash-talk someone in an email then hit “reply all” • you’re being micromanaged—or not being managed at all • you catch a colleague in a lie • your boss seems unhappy with your work • your cubemate’s loud speakerphone is making you homicidal • you got drunk at the holiday party Praise for Ask a Manager “A must-read for anyone who works . . . [Alison Green’s] advice boils down to the idea that you should be professional (even when others are not) and that communicating in a straightforward manner with candor and kindness will get you far, no matter where you work.”—Booklist (starred review) “The author’s friendly, warm, no-nonsense writing is a pleasure to read, and her advice can be widely applied to relationships in all areas of readers’ lives. Ideal for anyone new to the job market or new to management, or anyone hoping to improve their work experience.”—Library Journal (starred review) “I am a huge fan of Alison Green’s Ask a Manager column. This book is even better. It teaches us how to deal with many of the most vexing big and little problems in our workplaces—and to do so with grace, confidence, and a sense of humor.”—Robert Sutton, Stanford professor and author of The No Asshole Rule and The Asshole Survival Guide “Ask a Manager is the ultimate playbook for navigating the traditional workforce in a diplomatic but firm way.”—Erin Lowry, author of Broke Millennial: Stop Scraping By and Get Your Financial Life Together

Software architecture is foundational to the development of large, practical software-intensive applications. This brand-new text covers all facets of software architecture and how it serves as the intellectual centerpiece of software development and evolution. Critically, this text focuses on supporting creation of real implemented systems. Hence the text details not only modeling techniques, but design, implementation, deployment, and system adaptation -- as well as a host of other topics -- putting the elements in context and comparing and contrasting them with one another. Rather than focusing on one method, notation, tool, or process, this new text/reference widely surveys software architecture techniques, enabling the instructor and practitioner to choose the right tool for the job at hand. Software Architecture is intended for upper-division undergraduate and graduate courses in software architecture, software design, component-based software engineering, and distributed systems; the text may also be used in introductory as well as advanced software engineering courses. "Designing a large software system is an extremely complicated undertaking that requires juggling differing perspectives and differing goals, and evaluating differing options. Applied Software Architecture is the best book yet that gives guidance as to how to sort out and organize the conflicting pressures and produce a successful design." -- Len Bass, author of Software Architecture in Practice. Quality software architecture design has always been important, but in today's fast-paced, rapidly changing, and complex development environment, it is essential. A solid, well-thought-out design helps to manage complexity, to resolve trade-offs among conflicting requirements, and, in general, to bring quality software to market in a more timely fashion. Applied Software Architecture provides practical guidelines and techniques for producing quality software designs. It gives an overview of software architecture basics and a detailed guide to architecture design tasks, focusing on four fundamental views of architecture--conceptual, module, execution, and code. Through four real-life case studies, this book reveals the insights and best practices of the most skilled software architects in designing software architecture. These case studies, written with the masters who created them, demonstrate how the book's concepts and techniques are embodied in state-of-the-art architecture design. You will learn how to: create designs flexible enough to incorporate tomorrow's technology; use architecture as the basis for meeting performance, modifiability, reliability, and safety requirements; determine priorities among conflicting requirements and arrive at a successful solution; and use software architecture to help integrate system components. Anyone involved in software architecture will find this book a valuable compendium of best practices and an insightful look at the critical role of architecture in software development. 0201325713B07092001

The software 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.

Views and Beyond

Software Architecture: Principles and Practice

Continuous Architecture

Agile Software Architecture

Using Psychology to Design Better Products & Services

Art of the Edo Period, 1600 - 1868. (Editor : William Watson).

Ask a Manager

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. It democratizes architecture. This book seeks to make architecture relevant to all software developers. Developers need to understand how to use constraints as guiderails 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 job 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.

An enterprise Java developer's guide to learning JAX-RS, context and dependency injection, JavaServer Faces (JSF), and microservices with Eclipse MicroProfile using the latest features of Jakarta EE Key FeaturesExplore Jakarta EE's latest features and API specifications and discover their benefitsBuild and deploy microservices using Jakarta EE 8 and Eclipse MicroProfileBuild robust RESTful web services for various enterprise scenarios using the JAX-RS, JSON-P, and JSON-B APIsBook Description Jakarta EE is widely used around the world for developing enterprise applications for a variety of domains. With this book, Java professionals will be able to enhance their skills to deliver powerful enterprise solutions using practical recipes. This second edition of the Jakarta EE Cookbook takes you through the improvements introduced in its latest version and helps you get hands-on with its significant APIs and features used for server-side development. You'll use Jakarta EE for creating RESTful web services and web applications with the JAX-RS, JSON-P, and JSON-B APIs and learn how you can improve the security of your enterprise solutions. Not only will you learn how to use the most important servers on the market, but you'll also learn to make the best of what they have to offer for your project. From an architectural point of view, this Jakarta book covers microservices, cloud computing, and containers. It allows you to explore all the tools for building reactive applications using Jakarta EE and core Java features such as lambdas. Finally, you'll discover how professionals can improve their projects by engaging with and contributing to the community. By the end of this book, you'll have become proficient in developing and deploying enterprise applications using Jakarta EE. What you will learnWork with Jakarta EE's most commonly used APIs and features for server-side developmentEnable fast and secure communication in web applications with the help of HTTP2Build enterprise applications with reusable componentsBreak down monoliths into microservices using Jakarta EE and Eclipse MicroProfileImprove your enterprise applications with multithreading and concurrencyRun applications in the cloud with the help of containersGet to grips with continuous delivery and deployment for shipping your applications effectivelyWho this book is for This book is for Java EE developers who want to build enterprise applications or update their legacy apps with Jakarta EE's latest features and specifications. Some experience of working with Java EE and knowledge of web and cloud computing will assist with understanding the concepts covered in this book.

A software architecture manifests the major early design decisions, which determine the system’s development, deployment and evolution. Thus, making better architectural decisions is one of the large challenges in software engineering. Software architecture knowledge management is about capturing practical experience and translating it into generalized architectural knowledge, and using this knowledge in the communication with stakeholders during all phases of the software lifecycle. This book presents a concise description of knowledge management in the software architecture discipline. It explains the importance of sound knowledge management practices for improving software architecture processes and products, and makes clear the role of knowledge management in software architecture and software development processes. It presents many approaches that are in use in software companies today, approaches that have been used in other domains, and approaches under development in academia. After an initial introduction by the editors, the contributions are grouped in three parts on "Architecture Knowledge Management", "Strategies and Approaches for Managing Architectural Knowledge", and "Tools and Techniques for Managing Architectural Knowledge". The presentation aims at information technology and software engineering professionals, in particular software architects and software architecture researchers. For the industrial audience, the book gives a broad and concise understanding of the importance of knowledge management for improving software architecture process and building capabilities in designing and evaluating better architectures for their mission- and business-critical systems. For researchers, the book will help to understand the applications of various knowledge management approaches in an industrial setting and to identify research challenges and opportunities.

The Definitive, Practical, Proven Guide to Architecting Modern Software--Fully Updated with New Content on Mobility, the Cloud, Energy Management, DevOps, Quantum Computing, and More Updated with eleven new chapters, Software Architecture in Practice, Fourth Edition, thoroughly explains what software architecture is, why it's important, and how to design, instantiate, analyze, evolve, and manage it in disciplined and effective ways. Three renowned software architects cover the entire lifecycle, presenting practical guidance, expert methods, and tested models for use in any project, no matter how complex. You'll learn how to use architecture to address accelerating growth in requirements, system size, and abstraction, and to manage emergent quality attributes as systems are dynamically combined in new ways. With insights for utilizing architecture to optimize key quality attributes--including performance, modifiability, security, availability, interoperability, testability, usability, deployability, and more--this guide explains how to manage and refine existing architectures, transform them to solve new problems, and build reusable architectures that become strategic business assets. Discover how architecture in uences (and is influenced by) technical environments, project lifecycles, business profiles, and your own practices Leverage proven patterns, interfaces, and practices for optimizing quality through architecture Architect for mobility, the cloud, machine learning, and quantum computing Design for increasingly crucial attributes such as energy efficiency and safety Scale systems by discovering architecturally significant influences, using DevOps and deployment pipelines, and managing architecture debt Understand architecture's role in the organization, so you can deliver more value Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Continuous Architecture in Practice

REST in Practice

Software Architecture with Python

Evaluating Software Architectures

Aligning Agile Processes and Software Architectures

A Risk-Driven Approach

Software architecture—the conceptual glue that holds every phase of a project together for its many stakeholders—is widely recognized as a critical element in modern software development. Practitioners have increasingly discovered that close attention to a software system ’ s architecture pays valuable dividends. Without an architecture that is appropriate for the problem being solved, a project will stumble along or, most likely, fail. Even with a superb architecture, if that architecture is not well understood or well communicated the project is unlikely to succeed. Documenting Software Architectures, Second Edition, provides the most complete and current guidance, independent of language or notation, on how to capture an architecture in a commonly understandable form. Drawing on their extensive experience, the authors first help you decide what information to document, and then, with guidelines and examples (in various notations, including UML), show you how to express an architecture so that others can successfully build, use, and maintain a system from it. The book features rules for sound documentation, the goals and strategies of documentation, architectural views and styles, documentation for software interfaces and software behavior, and templates for capturing and organizing information to generate a coherent package. New and improved in this second edition: Coverage of architectural styles such as service-oriented architectures, multi-tier architectures, and data models Guidance for documentation in an Agile development environment Deeper treatment of documentation of rationale, reflecting best industrial practices Improved templates, reflecting years of use and feedback, and more documentation layout options A new, comprehensive example (available online), featuring documentation of a Web-based service-oriented system Reference guides for three important architecture documentation languages: UML, AADL, and SysML

This book presents a systematic model-based approach for software architecture according to three complementary viewpoints: structure, behavior, and execution. It covers a unified modeling approach and consolidates theory and practice with well-established learning outcomes. The authors cover the fundamentals of software architecture description and presents SysADL, a specialization of the OMG Standard Systems Modeling Language (SysML) with the aim of bringing together the expressive power of an Architecture Description Language (ADL) with a standard notation, widely accepted by industry and compliant with the ISO/IEC/IEEE 42010 Standard on Architecture Description in Systems and Software Engineering. The book is clearly structured in four parts: The first part focuses on the fundamentals of software architecture, exploring the concepts and constructs for modeling software architecture from differing viewpoints. Each chapter covers a specific viewpoint illustrated with examples of a real system. The second part focuses on how to design software architecture for achieving quality attributes. Each chapter covers a specific quality attribute and presents well-defined approaches to achieve it. Each architectural case study is illustrated with different examples drawn from a real-life system. The third part shows readers how to apply software architecture style to design architectures that meet the quality attributes. Each chapter covers a specific architectural style and gives insights on how to describe substyles. Each style is illustrated by variants and examples of a real-life system. The fourth part presents how to textually represent software architecture models to complement visual notation, including different examples. Software Architecture in Action is designed for teaching the required modeling techniques to both undergraduate and graduate students, giving them the practical techniques and tools needed to design the architecture of software-intensive systems. Similarly, this book will appeal to software development architects, designers, programmers and project managers too. A quick start guide to learning essential software architecture tools, frameworks, design patterns, and best practices Key Features: Apply critical thinking to your software development and architecture practices and bring structure to your approach using well-known IT standards Understand the impact of cloud-native approaches on software architecture Integrate the latest technology trends into your architectural designs Book Description: Are you a seasoned developer who likes to add value to a project beyond just writing code? Have you realized that good development practices are not enough to make a project successful, and you now want to embrace the bigger picture in the IT landscape? If so, you’re ready to become a software architect; someone who can deal with any IT stakeholder as well as add value to the numerous dimensions of software development. The sheer volume of content on software architecture can be overwhelming, however. Software Architecture for Busy Developers is here to help. Written by St é phane Eyskens, author of The Azure Cloud Native Mapbook, this book guides you through your software architecture journey in a pragmatic way using real-world scenarios. By drawing on over 20 years of consulting experience, St é phane will help you understand the role of a software architect, without the fluff or unnecessarily complex theory. You'll begin by understanding what non-functional requirements mean and how they concretely impact target architecture. The book then covers different frameworks used across the entire enterprise landscape with the help of use cases and examples. Finally, you'll discover ways in which the cloud is becoming a game changer in the world of software architecture. By the end of this book, you'll have gained a holistic understanding of the architectural landscape, as well as more specific software architecture skills. You'll also be ready to pursue your software architecture journey on your own - and in just one weekend! What You Will Learn: Understand the roles and responsibilities of a software architect Explore enterprise architecture tools and frameworks such as The Open Group Architecture Framework (TOGAF) and ArchiMate Get to grips with key design patterns used in software development Explore the widely adopted Architecture Tradeoff Analysis Method (ATAM) Discover the benefits and drawbacks of monoliths, service-oriented architecture (SOA), and microservices Stay on top of trending architectures such as API-driven, serverless, and cloud native Who this book is for: This book is for developers who want to move up the organizational ladder and become software architects by understanding the broader application landscape and discovering how large enterprises deal with software architecture practices. Prior knowledge of software development is required to get the most out of this book.

This book introduces all the relevant information required to understand and put Model Driven Architecture (MDA) into industrial practice. It clearly explains which conceptual primitives should be present in a system specification, how to use UML to properly represent this subset of basic conceptual constructs, how to identify just those diagrams and modeling constructs that are actually required to create a meaningful conceptual schema, and how to accomplish the transformation process between the problem space and the solution space. The approach is fully supported by commercially available tools.

Laws of UX

Practical recipes for enterprise Java developers to deliver large scale applications with Jakarta EE, 2nd Edition

Theory and Practice

Domain-driven Design

Software Architecture for Busy Developers

Applied Software Architecture

Fowler

Software architecture is a subfield of software development process and is also referred to as architectural decisions. It deals with the development, design, construction, deployment, maintenance, and production of structures which are the part of software systems. It includes software elements, their interrelation and their properties. This book is compiled in such a manner, that it will provide in-depth knowledge about the theory and practice of software architecture. Some of the diverse topics covered in it address the varied branches that fall under this category. This textbook is a complete source of knowledge on the present status of this important field. It is meant for those who are seeking a comprehensive analysis of the present status of this field.

An exploration of why we play video games despite the fact that we are almost certain to feel unhappy when we fail at them. We may think of video games as being “fun,” but in *The Art of Failure*, Jesper Juul claims that this is almost entirely mistaken. When we play video games, our facial expressions are rarely those of happiness or bliss. Instead, we frown, grimace, and shout in frustration as we lose, or die, or fail to advance to the next level. Humans may have a fundamental desire to succeed and feel competent, but game players choose to engage in an activity in which they are nearly certain to fail and feel incompetent. So why do we play video games even though they make us unhappy? Juul examines this paradox. In video games, as in tragic works of art, literature, theater, and cinema, it seems that we want to experience unpleasantness even if we also dislike it. Reader or audience reaction to tragedy is often explained as catharsis, as a purging of negative emotions. But, Juul points out, this doesn't seem to be the case for video game players. Games do not purge us of unpleasant emotions; they produce them in the first place. What, then, does failure in video game playing do? Juul argues that failure in a game is unique in that when you fail in a game, you (not a character) are in some way inadequate. Yet games also motivate us to play more, in order to escape that inadequacy, and the feeling of escaping failure (often by improving skills) is a central enjoyment of games. Games, writes Juul, are the art of failure: the singular art form that sets us up for failure and allows us to experience it and experiment with it. *The Art of Failure* is essential reading for anyone interested in video games, whether as entertainment, art, or education.

Describes ways to incorporate domain modeling into software development.

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 software into end-users 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 into account when deploying agile 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 the Agile and Continuous 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

Jakarta EE Cookbook

Foundations, Theory, and Practice

Support Constant Change

Software Architecture with C++

Sustainable Architecture in an Agile and Cloud-Centric World

Essential Software Architecture

The Software Architect Elevator

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

The award-winning and highly influential *Software Architecture in Practice*, Third Edition, has been substantially revised to reflect the latest developments in the field. In a real-world setting, the book once again introduces the concepts and best practices of software architecture—how a software system is structured and how that system's elements are meant to interact. Distinct from the details of implementation, algorithm, and data representation, an architecture holds the key to achieving system quality, is a reusable asset that can be applied to subsequent systems, and is crucial to a software organization's business strategy. The authors have structured this edition around the concept of architecture influence cycles. Each cycle shows how architecture influences, and is influenced by, a particular context in which architecture plays a critical role. Contexts include technical environment, the life cycle of a project, an organization's business profile, and the architect's professional practices. The authors also have greatly expanded their treatment of quality attributes, which remain central to their architecture philosophy—with an entire chapter devoted to each attribute—and broadened their treatment of architectural patterns. If you design, develop, or manage large software systems (or plan to do so), you will find this book to be a valuable resource for getting up to speed on the state of the art. Totally new material covers Contexts of software architecture: technical, project, business, and professional Architecture competence: what this means both for individuals and organizations The origins of business goals and how this affects architecture Architecturally significant requirements, and how to determine them Architecture in the life cycle, including generate-and-test as a design philosophy; architecture conformance during implementation; architecture and testing; and architecture and agile development Architecture and current technologies, such as the cloud, social networks, and end-user devices

In Continuous Architecture in Practice, three leading software architecture experts update the discipline's classic practices for today's environments, software development contexts, and applications. Coverage includes: Discover what's changed, and how the architect's role must change Reflect today's quality attributes in evolvable architectures Understand team-based software architecture, and architecture as a “flow of decisions” Architect for security, including continuous threat modeling and mitigation Explore architectural opportunities to improve performance in continuous delivery environments Architect for scalability, avoid common scalability pitfalls, and scale microservices and serverless environments Improve resilience and reliability in the face of inevitable failures Architect data for NoSQL, big data, and analytics Use architecture to promote innovation: case studies in AI/ML, chatbots, and blockchain

The practice of enterprise application development has benefited from the emergence of many new enabling technologies. Multi-tiered object-oriented platforms, such as Java and .NET, have become commonplace. These new tools and technologies are capable of building powerful applications, but they are not easily implemented. Common failures in enterprise applications often occur because their developers do not understand the architectural lessons that experienced object developers have learned. Patterns of Enterprise Application Architecture is written in direct response to the stiff challenges that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology—from Smalltalk to CORBA to Java to .NET—the same basic design ideas can be adapted and applied to solve common problems. With the help of an expert group of contributors, Martin distills over forty recurring solutions into patterns. The result is an indispensable handbook of solutions that are applicable to any enterprise application platform. This book is actually two books in one. The first section is a short tutorial on developing enterprise applications, which you can read from start to finish to understand the scope of the book's lessons. The next section, the bulk of the book, is a detailed reference to the patterns themselves. Each pattern provides usage and implementation information, as well as detailed code examples in Java or C#. The entire book is also richly illustrated with UML diagrams to further explain the concepts. Armed with this book, you will have the knowledge necessary to make important architectural decisions about building an enterprise application and the proven patterns for use when building them. The topics covered include · Dividing an enterprise application into layers · The major approaches to organizing business logic · An in-depth treatment of mapping between objects and relational databases · Using Model-View-Controller to organize a Web presentation · Handling concurrency for data that spans multiple transactions · Designing distributed object interfaces

Model-Driven Architecture in Practice

Software Architecture in the Age of Agility and Devops

An Engineering Approach

Documenting Software Architectures

Software Systems Architecture

Designing and Executing Architectural Models with SysADL Grounded on the OMG SysML Standard

Software Architecture in Action

There are no easy decisions in software architecture. Instead, there are many hard parts—difficult problems or issues with no best practices—that force you to choose among various compromises. With this book, you'll learn how to think critically about the trade-offs involved with distributed architectures. Architecture veterans and practicing consultants Neal Ford, Mark Richards, Pramod Sadalage, and Zhamak Dehghani discuss strategies for choosing an appropriate architecture. By interweaving a story about a fictional group of technology professionals—the Sysops Squad—they examine everything from how to determine service granularity, manage workflows and orchestration, manage and decouple contracts, and manage distributed transactions to how to optimize operational characteristics, such as scalability, elasticity, and performance. By focusing on commonly asked questions, this book provides techniques to help you discover and weigh the trade-offs as you confront the issues you face as an architect. Analyze trade-offs and effectively document your decisions Make better decisions regarding service granularity Understand the complexities of breaking apart monolithic applications Manage and decouple contracts between services Handle data in a highly distributed architecture Learn patterns to manage workflow and transactions when breaking apart applications

This award-winning book, substantially updated to reflect the latest developments in the field, introduces the concepts and best practices of software architecture—how a software system is structured and how that system's elements are meant to interact. Distinct from the details of implementation, algorithm, and data representation, an architecture holds the key to achieving system quality, is a reusable asset that can be applied to subsequent systems, and is crucial to a software organization's business strategy. Drawing on their own extensive experience, the authors cover the essential technical topics for designing, specifying, and validating a system. They also emphasize the importance of the business context in which large systems are designed. Their aim is to present software architecture in a real-world setting, reflecting both the opportunities and constraints that companies encounter. To that end, case studies that describe successful architectures illustrate key points of both technical and organizational discussions. Topics new to this edition include: Architecture design and analysis, including the Architecture Tradeoff Analysis Method (ATAM) Capturing quality requirements and achieving them through quality scenarios and tactics Using architecture reconstruction to recover undocumented architectures Documenting architectures using the Unified Modeling Language (UML) New case studies, including Web-based examples and a wireless Enterprise JavaBeans™ (EJB) system designed to support wearable computers The financial aspects of architectures, including use of the Cost Benefit Analysis Method (CBAM) to make decisions If you design, develop, or manage the building of large software systems (or plan to do so), or if you are interested in acquiring such systems for your corporation or government agency, use *Software Architecture in Practice*, Second Edition, to get up to speed on the current state of software architecture.

REST continues to gain momentum as the best method for building Web services, and this down-to-earth book delivers techniques and examples that show how to design and implement integration solutions using the REST architectural style.

Job titles like “Technical Architect” and “Chief Architect” nowadays abound in software industry, yet many people suspect that “architecture” is one of the most overused and least understood terms in professional software development. Gorton's book tries to resolve this dilemma. It concisely describes the essential elements of knowledge and key skills required to be a software architect. The explanations encompass the essentials of architecture thinking, practices, and supporting technologies. They range from a general understanding of structure and quality attributes through technical issues like middleware components and service-oriented architectures to recent technologies like model-driven architecture, software product lines, aspect-oriented design, and the Semantic Web, which will presumably influence future software systems. This second edition contains new material covering enterprise architecture, agile development, enterprise service bus technologies, RESTful Web services, and a case study on how to use the MeDICi integration framework. All approaches are illustrated by an ongoing real-world example. So if you work as an architect or senior designer (or want to someday), or if you are a student in software engineering, here is a valuable and yet approachable knowledge source for you.

Principles and Practice

Occupational Outlook Handbook

Tackling Complexity in the Heart of Software

Software Architecture

Software Architecture in Practice, 4th Edition

How to Navigate Clueless Colleagues, Lunch-Stealing Bosses, and the Rest of Your Life at Work

Software Architecture: The Hard Parts

The Definitive, Practical, Proven Guide to Architecting Modern Software--Now Fully Updated Now with nine new chapters, Software Architecture in Practice, Fourth Edition, thoroughly explains what software architecture is, why it's important, and how to design, instantiate, analyze, evolve, and manage it in disciplined and effective ways. Three renowned software architects cover the entire lifecycle, presenting practical guidance, expert methods, and tested models for use in any project, no matter how complex. You'll learn how to use architecture to address accelerating growth in requirements, system size, and abstraction, and to manage emergent quality attributes as systems are dynamically combined in new ways. With insights for utilizing architecture to optimize key quality attributes—including performance, modifiability, security, availability, interoperability, testability, usability, deployability, and more--this guide explains how to manage and refine existing architectures, transform them to solve new problems, and build reusable architectures that become strategic business assets. Discover how architecture influences (and is influenced by) technical environments, project lifecycles, business profiles, and your own practices Leverage proven patterns, interfaces, and practices for optimizing quality through architecture Architect for mobility, the cloud, machine learning, and quantum computing Design for increasingly crucial attributes such as energy efficiency and safety Scale systems by discovering architecturally significant influences, using DevOps and deployment pipelines, and managing architecture debt Understand architecture's role in the organization, so you can deliver more value.

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

This Book Describes Systematic Methods For Evaluating Software Architectures And Applies Them To Real-Life Cases. Evaluating Software Architectures Introduces The Conceptual Background For Architecture Evaluation And Provides A Step-By-Step Guide To The Process Based On Numerous Evaluations Performed In Government And Industry.

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.

The Art of Failure

Software Architecture Knowledge Management

Designing Software Architectures

Redefining the Architect's Role in the Digital Enterprise

Talk and Act Like a Software Architect in One Weekend

Pattern Enterpr Applica Arch

Fundamentals of Software Architecture

Apply business requirements to IT infrastructure and deliver a high-quality product by understanding architectures such as microservices, DevOps, and cloud-native using modern C++ standards and features Key FeaturesDesign scalable large-scale applications with the C++ programming languageArchitect software solutions in a cloud-based environment with continuous integration and continuous delivery (CI/CD)Achieve architectural goals by leveraging design patterns, language features, and useful toolsBook Description Software architecture refers to the high-level design of complex applications. It is evolving just like the languages we use, but there are architectural concepts and patterns that you can learn to write high-performance apps in a high-level language without sacrificing readability and maintainability. If you're working with modern C++, this practical guide will help you put your knowledge to work and design distributed, large-scale apps. You'll start by getting up to speed with architectural concepts, including established patterns and rising trends, then move on to understanding what software architecture actually is and start exploring its components. Next, you'll discover the design concepts involved in application architecture and the patterns in software development, before going on to learn how to build, package, integrate, and deploy your components. In the concluding chapters, you'll explore different architectural qualities, such as maintainability, reusability, testability, performance, scalability, and security. Finally, you will get an overview of distributed systems, such as service-oriented architecture, microservices, and cloud-native, and understand how to apply them in application development. By the end of this book, you'll be able to build distributed services using modern C++ and associated tools to deliver solutions as per your clients' requirements. What you will learnUnderstand how to apply the principles of software architectureApply design patterns and best practices to meet your architectural goalsWrite elegant, safe, and performant code using the latest C++ featuresBuild applications that are easy to maintain and deployExplore the different architectural approaches and learn to apply them as per your requirementSimplify development and operations using application containersDiscover various techniques to solve common problems in software design and developmentWho this book is for This software architecture C++ programming book is for experienced C++ developers looking to become software architects or develop enterprise-grade applications.

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 knowledge, and includes rich 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 architectural concerns. 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

with agile methods. 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 or lightweight architecture 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

Software Architecture in Practice Addison-Wesley

Architect and design highly scalable, robust, clean, and highly performant applications in Python About This Book Identify design issues and make the necessary adjustments to achieve improved performance Understand practical architectural quality attributes from the perspective of a practicing engineer and architect using Python Gain knowledge of architectural principles and how they can be used to provide accountability and rationale for architectural decisions Who This Book Is For This book is for experienced Python developers who are aspiring to become the architects of enterprise-grade applications or software architects who would like to leverage Python to create effective blueprints of applications. What You Will Learn Build programs with the right architectural attributes Use Enterprise Architectural Patterns to solve scalable problems on the Web Understand design patterns from a Python perspective Optimize the performance testing tools in Python Deploy code in remote environments or on the Cloud using Python Secure architecture applications in Python In Detail This book starts off by explaining how Python fits into an application architecture. As you move along, you will understand the architecturally significant demands and how to determine them. Later, you'll get a complete understanding of the different architectural quality requirements that help an architect to build a product that satisfies business needs, such as maintainability/reusability, testability, scalability, performance, usability, and security. You will use various techniques such as incorporating DevOps, Continuous Integration, and more to make your application robust. You will understand when and when not to use object orientation in your applications. You will be able to think of the future and design applications that can scale proportionally to the growing business. The focus is on building the business logic based on the business process documentation and which frameworks are to be used when. We also cover some important patterns that are to be taken into account while solving design problems as well as those in relatively new domains such as the Cloud. This book will help you understand the ins and outs of Python so that you can make those critical design decisions that not just live up to but also surpass the expectations of your clients. Style and approach Filled with examples and use cases, this guide takes a no-nonsense approach to help you with everything it takes to become a successful software architect.

Building Evolutionary Architectures

Software Architect's Handbook

A Practical Approach

An Essay on the Pain of Playing Video Games

Hypermedia and Systems Architecture

The Great Japan Exhibition

Software development organizations are now discovering the efficiencies that can be achieved by architecting entire software product families together. In Software Architecture for Product Families, experts from one of the world's most advanced software domain engineering projects share in-depth insights about the techniques that work -- and those that don't. The book offers a solutions-oriented, case-study approach covering the entire development lifecycle, based on advanced work done by three of Europe's leading technology companies and their academic partners. Discover the challenges that drive companies to consider architecting product families, and the new problems they encounter in doing so. Master concepts and terms that can be used to describe the architecture of a product family; then learn how to assess that architecture, and transform it into working applications. The authors also present chapter-length, real-world case studies of domain engineering projects at Nokia, Philips, and ABB.