Documenting Software Architectures Views And Beyond Sei Series In Software Engineering Hardcover

Abstract: "Architecture documentation has emerged as an important architecture-related practice. In 2002, researchers at the Carnegie Mellon[registered trademark] Software Engineering Institute completed Documenting Software Architectures: Views and Beyond (V & B), an approach that holds that documenting a software architecture is a matter of choosing a set of relevant views of the architecture, documenting each of those views, and then documenting information that applies to more than one view or to the set of views as a whole. Details of the approach include a method for choosing the most relevant views, standard templates for documenting views and the information beyond them, and definitions of the templates' content. At about the same time, the Institute of Electrical and Electronics Engineers (IEEE) was developing a recommended best practice for describing architectures for software-intensive systems -- ANSI/IEEE Std. 1471-2000. Like V & B, that standard takes a multi-view approach to the task of architecture documentation, and it establishes a conceptual framework for architectural description and defines the content of an architectural description. This technical note summarizes the two approaches and shows how a software architecture document prepared using the V & B approach can be made compliant with Std. 1471-2000."

Over the past 20 years, software architectures have significantly contributed to the development of complex and distributed systems. Nowadays, it is recognized that one of the critical problems in the design and development of any complex software system is its architecture, i.e. the organization of its architectural elements. Software Architecture presents the software architecture paradigms based on objects, components, services and models, as well as the various architectural techniques and methods, the analysis of architectural qualities, models of representation of architectural templates and styles, their formalization, validation and testing and finally the engineering approach in which these consistent and autonomous elements can be tackled.

Architecture is crucial to the success of any large software system -- but even a superb architecture will fail if it isn't communicated well. Now, there's a language- and notation-independent guide to capturing architecture so it can be used successfully by every analyst, software designer, and developer. The authors review the diverse goals and uses of software architecture documentation, providing documentation strategies for several common scenarios. They identify the basic unit of software architecture documentation: the viewtype, which specifies the type of information to be provided in an architectural view. For each viewtype -- Modules, Component-and-Connectors, and Allocation -- they offer detailed guidance on documenting what really matters. Next, they demonstrate how to package architecture documentation in

coherent, usable form: augmenting architectural views with documentation of interfaces and behavior; accounting for architectural variability and dynamic systems; and more.

With this practical book, architects, CTOs, and CIOs will learn a set of patterns for the practice of architecture, including analysis, documentation, and communication. Author Eben Hewitt shows you how to create holistic and thoughtful technology plans, communicate them clearly, lead people toward the vision, and become a great architect or Chief Architect. This book covers each key aspect of architecture comprehensively, including how to incorporate business architecture, information architecture, data architecture, application (software) architecture together to have the best chance for the system’s success. Get a practical set of proven architecture practices focused on shipping great products using architecture Learn how architecture works effectively with development teams, management, and product management teams through the value chain Find updated special coverage on machine learning architecture Get usable templates to start incorporating into your teams immediately Incorporate business architecture, information architecture, data architecture, and application (software) architecture together

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 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.

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

Abstract: "This report compares the Software Engineering Institute's Views and Beyond approach for documenting software architectures with the documentation philosophy embodied in agile software-development methods. This report proposes an approach for capturing architecture information in a way that is consistent with agile methods."

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.

Part of the new series, Advanced Topics in Science and Technology in China, this book aims to introduce the theoretical foundations, various sub-fields, current research, and practical methods of software architecture. First off, readers can acquire a basic knowledge of software architecture, including why software architecture is necessary. They are then shown how to describe a system’s architecture with formal language. The authors continue by delineating which architecture styles are popular in practice.

Document the architecture of your software easily with this highly practical, open-source template. Key Features Get to grips with leveraging the features of arc42 to create insightful documents Learn the concepts of software architecture documentation through real-world examples

Discover techniques to create compact, helpful, and easy-to-read documentation Book Description When developers document the architecture of their systems, they often invent their own specific ways of articulating structures, designs, concepts, and decisions. What they need is a template that enables simple and efficient software architecture documentation. arc42 by Example shows how it's done through several real-world examples. Each example in the book, whether it is a chess engine, a huge CRM system, or a cool web system, starts with a brief description of the problem domain and the quality requirements. Then, you'll discover the system context with all the external interfaces. You'll dive into an overview of the solution strategy to implement the building blocks and runtime scenarios. The later chapters also explain various cross-cutting concerns and how they affect other aspects of a program. What you will learn Utilize arc42 to document a system's physical infrastructure Learn how to identify a system's scope and boundaries Break a system down into building blocks and illustrate the relationships between them Discover how to describe the runtime behavior of a system Know how to document design decisions and their reasons Explore the risks and technical debt of your system Who this book is for This book is for software developers and solutions architects who are looking for an easy, open-source tool to document their systems. It is a useful reference for those who are already using arc42. If you are new to arc42, this book is a great learning resource. For those of you who want to write better technical documentation will benefit from the general concepts covered in this book.

A Comprehensive Process for Defining Software Architectures That Work A good software architecture is the foundation of any successful software system. Effective architecting requires a clear understanding of organizational roles, artifacts, activities performed, and the optimal sequence for performing those activities. With The Process of Software Architecting , Peter Eeles and Peter Cripps provide guidance on these challenges by covering all aspects of architecting a software system, introducing best-practice techniques that apply in every environment, whether based on Java EE, Microsoft .NET, or other technologies. Eeles and Cripps first illuminate concepts related to software architecture, including architecture documentation and reusable assets. Next, they present an accessible, task-focused guided tour through a typical project, focusing on the architect’s role, with common issues illuminated and addressed throughout. Finally, they conclude with a set of best practices that can be applied to today’s most complex systems. You will come away from this book understanding The role of the architect in a typical software development project How to document a software architecture to satisfy the needs of different stakeholders The applicability of reusable assets in the process of architecting The role of the architect with respect to requirements definition The derivation of an architecture based on a set of requirements The relevance of architecting in creating complex systems The Process of Software Architecting will be an indispensable resource for every working and aspiring software architect—and for every project manager and other software professional who needs to understand how architecture influences their work.

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 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.

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.

Architectural design is a crucial first step in developing complex software intensive systems. Early design decisions establish the structures necessary for achieving broad systemic properties. However, today's organizations lack synergy between software their development processes and technological methodologies. Providing a thorough treatment of Documenting Software ArchitecturesViews and BeyondPearson Education

Introduction. Architectural styles. Case studies. Shared information systems. Architectural design guidance. Formal models and specifications. Linguistics issues. Tools for architectural design. Education of software architects.

Abstract: "An important issue for software system development is the documentation of architecture designs. In this report, we describe techniques for the architectural documentation of software-based systems in the context of development processes that use UML for software design. The architectural documentation is organized in four kinds of

views: problem domain view, code view, run-time view and deployment view. We examine JavaPhone[TM] as a case study to illustrate the approach: what kinds of information are provided in each kind of view, what forms of notation should be used, what are their limitations, and what uses can be made of this documentation."

A guide for leveraging SketchUp for any project size, type, or style. New construction or renovation. The revised and updated second edition of The SketchUp Workflow for Architecture offers guidelines for taking SketchUp to the next level in order to incorporate it into every phase of the architectural design process. The text walks through each step of the SketchUp process from the early stages of schematic design and model organization for both renovation and new construction projects to final documentation and shows how to maximize the LayOut toolset for drafting and presentations. Written by a noted expert in the field, the text is filled with tips and techniques to access the power of SketchUp and its related suite of tools. The book presents a flexible workflow method that helps to make common design tasks easier and gives users the information needed to incorporate varying degrees of SketchUp into their design process. Filled with best practices for organizing projects and drafting schematics, this resource also includes suggestions for working with LayOut, an underused but valuable component of SketchUp Pro. In addition, tutorial videos compliment the text and clearly demonstrate more advanced methods. This important text: Presents intermediate and advanced techniques for architects who want to use SketchUp in all stages of the design process Includes in-depth explanations on using the LayOut tool set that contains example plans, details, sections, presentations, and other information Updates the first edition to reflect the changes to SketchUp 2018 and the core functionalities, menus, tools, inferences, arc tools, reporting, and much more Written by a SketchUp authorized trainer who has an active online platform and extensive connections within the SketchUp community Contains accompanying tutorial videos that demonstrate some of the more advanced SketchUp tips and tricks Written for professional architects, as well as professionals in interior design and landscape architecture, The SketchUp Workflow for Architecture offers a revised and updated resource for using SketchUp in all aspects of the architectural design process.

“This is an incredibly wise and useful book. The authors have considerable real-world experience in delivering quality systems that matter, and their expertise shines through in these pages. Here you will learn what technical debt is, what is it not, how to manage it, and how to pay it down in responsible ways. This is a book I wish I had when I was just beginning my career. The authors present a myriad of case studies, born from years of experience, and offer a multitude of actionable insights for how to apply it to your project.” –Grady Booch, IBM Fellow Master Best Practices for Managing Technical Debt to Promote Software Quality and Productivity As software systems mature, earlier design or code decisions made in the context of budget or schedule constraints increasingly impede evolution and innovation. This

phenomenon is called technical debt, and practical solutions exist. In Managing Technical Debt, three leading experts introduce integrated, empirically developed principles and practices that any software professional can use to gain control of technical debt in any software system. Using real-life examples, the authors explain the forms of technical debt that afflict software-intensive systems, their root causes, and their impacts. They introduce proven approaches for identifying and assessing specific sources of technical debt, limiting new debt, and “paying off” debt over time. They describe how to establish managing technical debt as a core software engineering practice in your organization. Discover how technical debt damages manageability, quality, productivity, and morale–and what you can do about it Clarify root causes of debt, including the linked roles of business goals, source code, architecture, testing, and infrastructure Identify technical debt items, and analyze their costs so you can prioritize action Choose the right solution for each technical debt item: eliminate, reduce, or mitigate Integrate software engineering practices that minimize new debt Managing Technical Debt will be a valuable resource for every software professional who wants to accelerate innovation in existing systems, or build new systems that will be easier to maintain and evolve.

This book introduces the concept of software architecture as one of the cornerstones of software in modern cars. Following a historical overview of the evolution of software in modern cars and a discussion of the main challenges driving that evolution, Chapter 2 describes the main architectural styles of automotive software and their use in cars' software. Chapter 3 details this further by presenting two modern architectural styles, i.e. centralized and federated software architectures. In Chapter 4, readers will find a description of the software development processes used to develop software on the car manufacturers' side. Chapter 5 then introduces AUTOSAR - an important standard in automotive software. Chapter 6 goes beyond simple architecture and describes the detailed design process for automotive software using Simulink, helping readers to understand how detailed design links to high-level design.^The new chapter 7 reports on how machine learning is exploited in automotive software e.g. for image recognition and how both on-board and off-board learning are applied. Next, Chapter 8 presents a method for assessing the quality of the architecture - ATAM (Architecture Trade-off Analysis Method) - and provides a sample assessment, while Chapter 9 presents an alternative way of assessing the architecture, namely by using quantitative measures and indicators. Subsequently Chapter 10 dives deeper into one of the specific properties discussed in Chapter 8 - safety - and details an important standard in that area, the ISO/IEC 26262 norm. Lastly, Chapter 11 presents a set of future trends that are currently emerging and have the potential to shape automotive software engineering in the coming years. This book explores the concept of software architecture for modern cars and is intended for both beginning and advanced software designers.^It mainly aims at two different groups of audience - professionals working with automotive software who need

to understand concepts related to automotive architectures, and students of software engineering or related fields who need to understand the specifics of automotive software to be able to construct cars or their components. Accordingly, the book also contains a wealth of real-world examples illustrating the concepts discussed and requires no prior background in the automotive domain. Compared to the first edition, besides the two new chapters 3 and 7 there are considerable updates in chapters 5 and 8 especially.

"... a curriculum geared toward helping students gain skills in consciously regulating their actions, which in turn leads to increased control and problem solving abilities. Using a cognitive behavior approach, the curriculum's learning activities are designed to help students recognize when they are in different states called "zones," with each of four zones represented by a different color. In the activities, students also learn how to use strategies or tools to stay in a zone or move from one to another. Students explore calming techniques, cognitive strategies, and sensory supports so they will have a toolbox of methods to use to move between zones. To deepen students' understanding of how to self-regulate, the lessons set out to teach students these skills: how to read others' facial expressions and recognize a broader range of emotions, perspective about how others see and react to their behavior, insight into events that trigger their less regulated states, and when and how to use tools and problem solving skills. The curriculum's learning activities are presented in 18 lessons. To reinforce the concepts being taught, each lesson includes probing questions to discuss and instructions for one or more learning activities. Many lessons offer extension activities and ways to adapt the activity for individual student needs. The curriculum also includes worksheets, other handouts, and visuals to display and share. These can be photocopied from this book or printed from the accompanying CD."--Publisher's website.

This is the lead book in a series of books from the Software Quality Institute (SQI). This series will bring together some of the key individuals in the Software Engineering community, and through their knowledge and experience, develop a library of books that set the standards for best practices in achieving high-quality software. This title presents a set of fundamental engineering strategies for achieving a successful software solution, with practical advice to ensure that the development project is moving in the right direction. Software designers and development managers can improve the development speed and quality of their software, and improve the processes used in development.

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.

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

Algorithms play an important role in both the science and practice of computing. To optimally use algorithms, a deeper understanding of their logic and mathematics is essential. Beyond traditional computing, the ability to apply these algorithms to solve real-world problems is a necessary skill, and this is what this book focuses on.

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

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

"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 First Complete Guide to DevOps for Software Architects DevOps promises to accelerate the release of new software features and improve monitoring of systems in production, but its crucial implications for software architects and architecture are often ignored. In DevOps: A Software Architect's Perspective, three leading architects address these issues head-on. The authors review decisions software architects must make in order to achieve DevOps' goals and clarify how other DevOps participants are

likely to impact the architect's work. They also provide the organizational, technical, and operational context needed to deploy DevOps more efficiently, and review DevOps' impact on each development phase. The authors address cross-cutting concerns that link multiple functions, offering practical insights into compliance, performance, reliability, repeatability, and security. This guide demonstrates the authors' ideas in action with three real-world case studies: datacenter replication for business continuity, management of a continuous deployment pipeline, and migration to a microservice architecture. Comprehensive coverage includes • Why DevOps can require major changes in both system architecture and IT roles • How virtualization and the cloud can enable DevOps practices • Integrating operations and its service lifecycle into DevOps • Designing new systems to work well with DevOps practices • Integrating DevOps with agile methods and TDD • Handling failure detection, upgrade planning, and other key issues • Managing consistency issues arising from DevOps' independent deployment models • Integrating security controls, roles, and audits into DevOps • Preparing a business plan for DevOps adoption, rollout, and measurement

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.

"This book covers both theoretical approaches and practical solutions in the processes for aligning enterprise, systems, and software architectures"--Provided by publisher.

Abstract: "Documenting software architecture (DSA) is a crucial facet in the development of a software system, yet often it is carried out in a haphazard fashion, if at all. Lack of attention to the documentation results from insufficient guidance about what should be documented and when and how to capture the information so that system stakeholders find it useful. The book Documenting Software Architectures: Views and Beyond provides such guidance in the DSA approach, and this report describes the conceptual design for a documentation system based on that approach. A system is envisioned that enables the architect to capture architectural decisions and related artifacts as a living repository that can communicate information to stakeholders who might be both geographically and temporally distributed. The system must communicate in a way that allows each stakeholder quick and easy access to information relevant to the person's role in the software development process. This report describes a design prototype that demonstrates a Web-based approach to creating, communicating, and using software architecture throughout the life of the system."

Abstract: "This report represents a milestone of a work in progress. That work is a comprehensive handbook on how to produce high-quality documentation for software architectures. The handbook, tentatively entitled Documenting Software Architectures, will be published in early 2002 by Addison Wesley Longman as part of the SEI Series on Software Engineering. Since this report is a snapshot of current work, the material described here may change before the handbook is published. The theme of the report is that documenting an architecture entails documenting the set of relevant views of that architecture, and then completing the picture by documenting information that transcends any single view. The audience for Documenting Software Architectures is the community of practicing architects, apprentice architects, and developers who receive architectural documentation."

Software Systems Architecture, Second Edition is a highly regarded, 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. With this book you will learn how to Design and communicate an architecture that reflects and balances the different needs of its stakeholders 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 Reflecting new standards and developments in the field, this new edition extends and updates much of the content, and Adds a “system context viewpoint” that documents the system's interactions with its environment Expands the discussion of architectural principles, showing how they can be used to provide traceability and rationale for architectural decisions Explains how agile development and architecture can work together Positions requirements and architecture activities in the project context Presents a new lightweight method for architectural validation 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.