Software Architecture In Practice SEI Series In Software Engineering Hardcover

When people should go to the book stores, search inauguration by shop, shelf by shelf, it is in reality problematic. This is why we give the book compilations in this website. It will agreed ease you to see guide **Software Architecture In Practice SEI Series In Software Engineering Hardcover** as you such as.

By searching the title, publisher, or authors of guide you truly want, you can discover them rapidly. In the house, workplace, or perhaps in your method can be all best place within net connections. If you set sights on to download and install the Software Architecture In Practice SEI Series In Software Engineering Hardcover , it is totally simple then, since currently we extend the belong to to purchase and create bargains to download and install Software Architecture In Practice SEI Series In Software Engineering Hardcover thus simple!

Software Architecture in Practice - Len Bass 2003-04-09 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.

Software Architecture in Practice - Len Bass 2003
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.

Continuous Architecture - Murat Erder 2015-10-21

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

Evaluating Software Architectures - Paul Clements 2002

The foundation of any software system is its architecture. Using this book, you can evaluate every aspect of architecture in advance, at remarkably low cost -- identifying improvements that can dramatically improve any system's performance, security, reliability, and maintainability. As the practice of software architecture has matured, it has become possible to identify causal connections between architectural design decisions and the qualities and properties that result downstream in the systems that follow from them. This book shows how, offering step-by-step guidance, as well as detailed practical examples -- complete with sample artifacts reflective of those that evaluators will encounter. The techniques presented here are applicable not only to software architectures, but also to system architectures encompassing computing hardware, networking equipment, and other elements. For all software architects, software engineers, developers, IT managers, and others responsible for creating, evaluating, or implementing software architectures.

Software Specification and Design - Ph.D., John C. Munson 2005-09-26

The rigors of engineering must soon be applied to the software development process, or the complexities of new systems will initiate the collapse of companies that attempt to produce them. Software Specification and Design: An Engineering Approach offers a foundation for rigorously engineered software. It provides a clear vision of what occurs at e

Fundamentals of Software Architecture - Mark Richards 2020-01-28

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

Essential Software Architecture - Ian Gorton 2011-04-27

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.

The Process of Software Architecting - Peter Eeles 2009-07-14

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.

Quality of Software Architectures - Christine Hofmeister 2006-12-07

This book constitutes the thoroughly refereed post-proceedings of the Second International Conference on the Quality of Software Architectures, QoSA 2006, held in Västerås, Sweden in June 2006, co-located with the 9th International Symposium on Component-Based Software Engineering, CBSE 2006. Coverage includes architecture evaluation, managing and applying architectural knowledge, and processes for supporting architecture quality.

Fowler - Martin Fowler 2012-03-09

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

Building Evolutionary Architectures - Neal Ford 2017-09-18
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.

Practical Software Architecture - Tilak Mitra 2015-11-18
Getting Architecture Just Right: Detailed Practical Guidance for Architecting Any Real-World IT Project To build effective architectures, software architects must tread a fine line between precision and ambiguity (a.k.abig animal pictures). This is difficult but

crucial: Failure to achieve this balance often leads directly to poor systems design and implementation. Now, pioneering IBM Distinguished Engineer and Chief Technology Officer Tilak Mitra offers the first complete guide to developing end-to-end solution architectures that are "just enough"--identifying and capturing the most important artifacts, without over-engineering or excessive documentation, and providing a practical approach to consistent and repeated success in defining software architectures. Practical Software Architecture provides detailed prescriptive and pragmatic guidance for architecting any real-world IT project, regardless of system, methodology, or environment. Mitra specifically identifies the artifacts that require emphasis and shows how to communicate evolving solutions with stakeholders, bridging the gap between architecture and implementation.

Software Architectures, Components, and Applications - Sven Overhage 2008-01-23

Researchers and professionals will find in this text the thoroughly refereed post-proceedings of the Third International Conference on the Quality of Software Architectures, QoSA 2007, held in Medford, MA, USA, in 2007. It was mounted in conjunction with the 10th International ACM SIGSOFT Symposium on Component-Based Software Engineering, CBSE 2007. The 13 revised full papers presented together with one keynote lecture were carefully reviewed and selected from 42 submissions.

Large-Scale Software Architecture - Jeff Garland 2003-07-25

The purpose of large-scale software architecture is to capture and describe practical representations to make development teams more effective. In this book the authors show how to utilise software architecture as a tool to guide the development instead of capturing the architectural details after all the design decisions have been made. * Offers a concise description of UML usage for large-scale architecture * Discusses software architecture and design principles * Technology and vendor independent

Component-Based Software Engineering - George Heineman 2005-04-28

On behalf of the Organizing Committee I am pleased to present the proceedings of the 2005 Symposium on Component-Based Software Engineering (CBSE). CBSE is concerned with the development of software-intensive systems from reusable parts (components), the development of reusable parts, and system maintenance and improvement by means of component replacement and c- tomization. CBSE 2005, "Software Components at Work," was the eighth in a series of events that promote a science and technology foundation for achieving predictable quality in software systems through the use of software component technology and its associated software engineering practices. We were fortunate to have a dedicated Program Committee comprised of 30 internationally recognized researchers and industrial practitioners. We received 91 submissions andeach paper wasreviewedby at least three ProgramComm- tee members (four for papers with an author on the Program Committee). The entirereviewingprocesswassup portedbyCyberChairPro,theWe b-basedpaper submissionandreviewsystemde velopedandsupportedbyRichard vandeStadt of Borbala Online Conference Services. After a two-day virtual Program C- mittee meeting, 21 submissions were accepted as long papers and 2 submissions were accepted as short papers.

A Software Architecture Primer - John Reekie 2006

The authors present a fresh, pragmatic approach to the study of software architecture. This edition contains a series of chapters that introduce and develop an understanding of software architecture by means

of careful explanation and elaboration of a range of key concepts. (Computer Books)
Software Architecture - Mary Shaw 1996
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.
Model-based Engineering with AADL - Peter H. Feiler 2013
The first complete guide to SAE AADL: written by the standard's author, completely authoritative, and promoted by both SAE and SEI * *Thoroughly explains the new SAE AADL architecture notation for model-based analysis and validation of mission/safety-critical software-reliant systems. *Presents many real-world examples: ideal for self-learning, instruction, and as a working reference. *Addresses a key standard pioneered by Boeing, Lockheed Martin, Rockwell Collins, DOD, FAA, NASA, ESA, JAXA, and many top universities. Embedded, software-reliant systems are increasingly critical in many industries. In response, 30+ organizations have joined SAE (formerly, the Society of Automobile Engineers) to define the Architecture Analysis and Design Language (AADL). This international industry standard will help streamline and improve systems development through state-of-the-art architecture modeling, analysis, and validation. Ideal for both self-learning and classroom instruction, and an excellent reference for implementers, Model-Based Engineering with AADL is the first book on this crucial new standard. It introduces the reader to all aspects of AADL notation as part of an architecture-centric, model-based engineering approach to discover embedded software systems problems earlier in the lifecycle, and thereby solve them more cost-effectively. Co-authored by Peter Feiler, the

standard's author and technical lead, this introductory reference and tutorial is packed with real-world examples. Throughout, the authors compare AADL to other modeling notations and approaches, while presenting the language via a complete case study: the development and analysis of a realistic example system through repeated refinement and analysis.

Software Systems Architecture - Rozanski 2005-09

Software Architect’s Handbook - Joseph Ingeno 2018-08-30
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.

Clean Architecture - Robert C. Martin 2017-09-12
Practical Software Architecture Solutions from the Legendary Robert C. Martin ("Uncle Bob") By applying universal rules of software architecture, you can dramatically improve developer productivity throughout the life of any software system. Now, building upon the success of his best-selling books Clean Code and The Clean Coder, legendary software craftsman Robert C. Martin ("Uncle Bob") reveals those rules and helps you apply them. Martin's Clean Architecture doesn't merely present options. Drawing on over a half-century of experience in software environments of every imaginable type, Martin tells you what choices to make and why they are critical to your success. As you've come to expect from Uncle Bob, this book is packed with direct, no-nonsense solutions for the real challenges you'll face–the ones that will make or break your projects. Learn what software

architects need to achieve–and core disciplines and practices for achieving it Master essential software design principles for addressing function, component separation, and data management See how programming paradigms impose discipline by restricting what developers can do Understand what’s critically important and what’s merely a “detail” Implement optimal, high-level structures for web, database, thick-client, console, and embedded applications Define appropriate boundaries and layers, and organize components and services See why designs and architectures go wrong, and how to prevent (or fix) these failures Clean Architecture is essential reading for every current or aspiring software architect, systems analyst, system designer, and software manager–and for every programmer who must execute someone else’s designs. Register your product for convenient access to downloads, updates, and/or corrections as they become available.

Just Enough Software Architecture - George Fairbanks 2010-08-30 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.

Continuous Architecture in Practice - Murat Erder 2021-04 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

Evaluating Software Architectures - Clements 2002-09
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 - Richard N. Taylor 2009-01-09
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.

Documenting Software Architectures : Views and Beyond - 2010

Managing Technical Debt - Philippe Kruchten 2019-04-15
“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.

The CIO’s Guide to Risk - Jessica Keyes 2017-11-22

In an age of globalization, widely distributed systems, and rapidly advancing technological change, IT professionals and their managers must understand that risk is ever present. The key to project success is to identify risk and subsequently deal with it. The CIO’s Guide to Risk addresses the many faces of risk, whether it be in systems development, adoption of bleeding edge tech, the push for innovation, and even the march toward all things social media. Risk management planning, risk identification, qualitative and quantitative risk analysis, contingency planning, and risk monitoring and control are all addressed on a macro as well as micro level. The book begins with a big-picture view of analyzing technology trends to evaluate risk. It shows how to conceptualize trends, analyze their effect on infrastructure, develop metrics to measure success, and assess risk in adapting new technology. The book takes an in-depth look at project-related risks. It explains the fundamentals of project management and how project management relates to systems development and technology implementation. Techniques for analyzing project risk include brainstorming, the Delphi technique, assumption analysis, and decision analysis. Metrics to track and control project risks include the Balance Scorecard, project monitoring and reporting, and business and technology metrics. The book also takes an in-depth look at the role of knowledge management and innovation management in identifying, assessing, and managing risk. The book concludes with an executive’s guide to the legal and privacy

issues related to risk management, as well overviews of risks associated with social media and mobile environments. With its checklists, templates, and worksheets, the book is an indispensable reference on risk and information technology.
Software Architecture 2 - Mourad Chabane Oussalah 2014-06-02
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.
The Rational Unified Process - Philippe Kruchten 2004
bull; Reflects all of the changes that were integrated into RUP v2003-the latest version of the very popular produc t bull; Learn the key concepts, fundamentals of structure, integral content, and motivation behind the RUP bull; Covers all phases of the software development lifecycle -from concept, to delivery, to revision
Beautiful Architecture - Diomidis Spinellis 2009-01-15
What are the ingredients of robust, elegant, flexible, and maintainable software architecture? Beautiful Architecture answers this question through a collection of intriguing essays from more than a dozen of today's leading software designers and architects. In each essay, contributors present a notable software architecture, and

analyze what makes it innovative and ideal for its purpose. Some of the engineers in this book reveal how they developed a specific project, including decisions they faced and tradeoffs they made. Others take a step back to investigate how certain architectural aspects have influenced computing as a whole. With this book, you'll discover: How Facebook's architecture is the basis for a data-centric application ecosystem The effect of Xen's well-designed architecture on the way operating systems evolve How community processes within the KDE project help software architectures evolve from rough sketches to beautiful systems How creeping featurism has helped GNU Emacs gain unanticipated functionality The magic behind the Jikes RVM self-optimizable, self-hosting runtime Design choices and building blocks that made Tandem the choice platform in high-availability environments for over two decades Differences and similarities between object-oriented and functional architectural views How architectures can affect the software's evolution and the developers' engagement Go behind the scenes to learn what it takes to design elegant software architecture, and how it can shape the way you approach your own projects, with Beautiful Architecture.

Architecture-centric Software Project Management - Daniel J. Paulish 2002

To fully leverage the value of software architecture in enterprise development projects, you need to expressly and consciously link architecture with project management. This book shows how, drawing on powerful lessons learned at Siemens, one of the world's leading software development organizations. The authors offer insight into project management for software architects, insight into software architecture for project managers, and above all, insight into integrating the

two disciplines to maximize the effectiveness of both of them. Learn how to develop cost and schedule estimates for development projects, based on software architecture; how to clarify architecture so projects can be more effectively planned and managed; and then how to use architecture to organize, implement, and measure the project iteratively as work progresses.
Documenting Software Architectures - Paul Clements 2010-10-05
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
Designing Software Architectures - Humberto Cervantes 2016-04-29
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

Head First Software Development - Dan Pilone 2008-12-26

Provides information on successful software development, covering such topics as customer requirements, task estimates, principles of good design, dealing with source code, system testing, and handling bugs.

Designing Software Architectures - Rick Kazman 2016-05-04

Designing Software Architectures is the first step-by-step guide to making the crucial design decisions that can make or break your software architecture. SEI expert Rick Kazman and Dr. Humberto Cervantes provide comprehensive guidance for ensuring that your architectural design decisions are consistently rational and evidence-based. Drawing on their own extensive experience, they demonstrate how to craft designs that are practical and effective, and support all phases of architectural development, from requirements to documentation. You'll learn how to successfully integrate the design process in an organizational context, including designing systems that will be built with agile methods. The authors begin with a general review of software architecture concepts and the software architecture lifecycle. Next, they explain what architecture design really means, introduce key design

concepts and principles, and walk through both conventional and alternative design processes. Building on this foundation, they introduce the new Attribute-Driven Design (ADD) 3.0 process, walk the reader through two extended ADD 3.0 case studies, and demonstrate how ADD 3.0 can lead to more successful designs. You'll learn how to scale design and analysis up and down - for example, to design for pre-sales processes and lightweight architecture reviews. Kazman and Cervantes illuminate the relationships between analysis and design, introduce a set of reusable design primitives, and identify issues and solutions for new domains, including cloud, mobile, and big data. Design is the core activity for software designers and architects, but for most practitioners, it's been a black art. This book offers the systematic guidance you need to consistently do it rationally, and do it right.
The Software Architect Elevator - Gregor Hohpe
2020-04-08
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
Applied Software Architecture - Christine Hofmeister 2000
"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

Design It! - Michael Keeling
2017-10-18
Don't engineer by coincidence-design it like you mean it! Filled with practical techniques, Design It! is the perfect introduction to software architecture for programmers who are ready to grow their design skills. Lead your team as a software architect, ask the right stakeholders the right questions, explore design options, and help your team implement a system that promotes the right -ilities. Share your design decisions, facilitate collaborative design workshops that are fast, effective, and fun-and develop more awesome software! With dozens of design methods, examples, and practical know-how, Design It! shows you how to become a software architect. Walk through the core concepts every architect must know, discover how to apply them, and learn a variety of skills that will make you a better programmer, leader, and designer. Uncover the big ideas behind software architecture and gain confidence working on projects big and small. Plan, design, implement, and evaluate software architectures and collaborate with your team, stakeholders, and other architects. Identify the right stakeholders and understand their needs, dig for architecturally significant requirements, write amazing quality attribute scenarios, and make confident decisions. Choose technologies based on their architectural impact, facilitate architecture-centric design workshops, and evaluate architectures using lightweight, effective methods. Write lean architecture

descriptions people love to read. Run an architecture design studio, implement the architecture you've designed, and grow your team's architectural knowledge. Good design requires good communication. Talk about your software architecture with stakeholders using whiteboards, documents, and code, and apply architecture-focused design methods in your day-to-day practice. Hands-on exercises, real-world scenarios, and practical team-based decision-making tools will get everyone on board and give you the experience you need to become a confident software architect.

Software Architecture: The Hard Parts - Neal Ford 2021-09-23

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