Architecting Software Intensive Systems A Practitioners Guide

Software Architecture in PracticeSoftware Architect's HandbookSoftware Architecture for Big Data and the CloudAligning Enterprise, System, and Software ArchitecturesThe Art of Systems ArchitectingFundamentals of Software ArchitectureEconomics-Driven Software ArchitectureAgile Software ArchitectureArchitecting Dependable Systems VSoftware Architecture 1Practical Software ArchitectureDesigning Software-Intensive Systems: Methods and PrinciplesDesign and Use of Software ArchitecturesThe Process of Software ArchitectingSoftware Architecture in ActionFowlerApplied Software ArchitectureSoftware ArchitectureArchitecting Software Intensive SystemsJust Enough Software ArchitectureSoftware Engineering for Variability Intensive SystemsSoftware Architecture for Product FamiliesSoftware ArchitectureSoftware Architecture with PythonThe Art of Systems ArchitectingThe Software Architect ElevatorSoftware Systems ArchitectureProject Management of Large Software-Intensive SystemsRelating System Quality and Software ArchitectureSummary of a Workshop on Software-Intensive Systems and Uncertainty at ScaleArchitecting Data-Intensive ApplicationsArchitecting Dependable Systems IIProceedings of the 11th European Conference on Software ArchitectureEffective Methods for Software and Systems IntegrationDocumenting Software ArchitecturesDesigning Data-

Intensive ApplicationsManaging Trade-offs in Adaptable Software ArchitecturesEvaluating Software ArchitecturesDesigning Software-Intensive Systems: Methods and PrinciplesSoftware Architecture Knowledge Management

Software Architecture in Practice

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

Software Architect’s Handbook

Today's architecting must handle systems of types unknown until very recently. New domains, including personal computers, intersatellite networks, health services, and joint service command and control are calling for new architectures-and for architects specializing in those domains. Since the original publication, of this bestselling text, these

Software Architecture for Big Data and the Cloud

System Quality and Software Architecture collects state-of-the-art knowledge on how to intertwine software quality requirements with software architecture and how quality attributes are exhibited by the architecture of the system. Contributions from leading researchers and industry evangelists detail the techniques required to achieve quality management in software architecting, and the best way to apply these techniques effectively in various application domains (especially in cloud, mobile and ultra-large-scale/internet-scale architecture) Taken together, these approaches show how to assess the value of total quality

management in a software development process, with an emphasis on architecture. The book explains how to improve system quality with focus on attributes such as usability, maintainability, flexibility, reliability, reusability, agility, interoperability, performance, and more. It discusses the importance of clear requirements, describes patterns and tradeoffs that can influence quality, and metrics for quality assessment and overall system analysis. The last section of the book leverages practical experience and evidence to look ahead at the challenges faced by organizations in capturing and realizing quality requirements, and explores the basis of future work in this area. Explains how design decisions and method selection influence overall system quality, and lessons learned from theories and frameworks on architectural quality Shows how to align enterprise, system, and software architecture for total quality Includes case studies, experiments, empirical validation, and systematic comparisons with other approaches already in practice.

Aligning Enterprise, System, and Software Architectures

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

The Art of Systems Architecting

Managing Trade-Offs in Adaptable Software Architectures explores the latest research on adapting large complex systems to changing requirements. To be able to adapt a system, engineers must evaluate different quality attributes, including trade-offs to balance functional and quality requirements to maintain a well-functioning system throughout the lifetime of the system. This comprehensive resource brings together research focusing on how to manage trade-offs and architect adaptive systems in different business contexts. It presents state-of-the-art techniques, methodologies, tools, best practices, and guidelines for developing adaptive systems, and offers guidance for future software engineering research and practice. Each contributed chapter considers the practical application of the topic through case studies, experiments, empirical validation, or systematic comparisons with other approaches already in practice. Topics of interest include, but are not limited to, how to architect a system for adaptability, software architecture for self-adaptive systems, understanding and balancing the trade-offs involved, architectural patterns for self-adaptive systems, how quality attributes are exhibited by the architecture of the system, how to connect the quality of a software architecture to system architecture or other system considerations, and more. Explains software architectural processes and metrics supporting highly adaptive and complex engineering Covers validation, verification, security, and quality assurance in system design Discusses domain-specific software

engineering issues for cloud-based, mobile, context-sensitive, cyber-physical, ultra-large-scale/internet-scale systems, mash-up, and autonomic systems Includes practical case studies of complex, adaptive, and context-critical systems

Fundamentals of Software Architecture

Today's architecting must handle systems of types unknown until very recently. New domains, including personal computers, intersatellite networks, health services, and joint service command and control are calling for new architectures-and for architects specializing in those domains. Since the original publication, of this bestselling text, these

Economics-Driven Software Architecture

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.

Agile Software Architecture

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.

Architecting Dependable Systems V

This book provides an achievable answer. The author proposes a method for designing software architectures, and product line architectures, which is based on his experience in industry and research. The first part of the book introduces the design method,

Software Architecture 1

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.

Practical Software Architecture

Economics-driven Software Architecture presents a guide for engineers and architects who need to understand the economic impact of architecture design decisions: the long term and strategic viability, cost-effectiveness, and sustainability of applications and systems. Economics-driven software development can increase quality, productivity, and profitability, but comprehensive knowledge is needed to understand the architectural challenges involved in dealing with the development of large, architecturally challenging systems in an economic way. This book covers how to apply economic considerations during the software architecting activities of a project. Architecture-centric approaches to development and systematic evolution, where managing complexity, cost reduction, risk mitigation, evolvability, strategic planning and long-term value creation are among the major drivers for adopting such approaches. It assists the objective assessment of the lifetime costs and benefits of evolving systems, and the identification of legacy situations, where architecture or a component is indispensable but can no longer be evolved to meet changing needs at economic cost. Such consideration will form the scientific foundation for reasoning about the economics of nonfunctional requirements in the context of architectures and architecting. Familiarizes readers with essential considerations in economic-informed and value-driven software design and analysis Introduces techniques for making value-based software architecting decisions Provides readers a better understanding of the methods of economics-driven architecting

Designing Software-Intensive Systems: Methods and Principles

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

Design and Use of Software Architectures

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.

The Process of Software Architecting

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 Architecture in Action

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

Fowler

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.

Applied Software Architecture

This survey contains expanded and peer-reviewed papers based on the selected contributions to the Workshop on Architecting Dependable Systems (WADS 2007), and the Third Workshop on the Role of Software Architecture for Testing and Analysis (ROSATEA 2007).

Software Architecture

The growing scale and complexity of software-intensive systems are introducing fundamental new challenges of uncertainty and scale that are particularly demanding for defense systems. To assist in meeting these challenges, the Department of Defense asked the NRC to assess the nature of U.S. national investment in software research. As part of this study, a workshop was held to examine uncertainty at scale in current and future software-intensive systems. This report presents a summary of the workshop discussions that centered on

process, architecture, and the grand scale; DoD software challenges for future systems; agility at scale; quality and assurance with scale and uncertainty; and enterprise scale and beyond. The report also offers a summary of key themes emerging from the workshop: architectural challenges in large-scale systems; the need for software engineering capability; and open questions and research opportunities.

Architecting Software Intensive Systems

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.

Just Enough Software Architecture

As software systems become ubiquitous, the issues of dependability become more and more critical. Given that solutions to these issues must be taken into account from the very beginning of the design process, it is appropriate that dependability is addressed at the architectural level. This book results from an effort to bring together the research communities of software architectures and dependability. Inspired by the ICSE 2003 Workshop on Software Architectures for Dependable

Systems, the book focuses on topics relevant to improving the state of the art in architecting dependable systems. The 15 thoroughly reviewed papers originate partly from the workshop; others were solicited in order to achieve complete coverage of all relevant aspects. The papers are organized into topical sections on architectures for dependability, fault-tolerance in software architectures, dependability analysis in software architectures, and industrial experience.

Software Engineering for Variability Intensive Systems

The book describes how to manage and successfully deliver large, complex, and expensive systems that can be composed of millions of line of software code, being developed by numerous groups throughout the globe, that interface with many hardware items being developed by geographically dispersed companies, where the system also includes people, policies, constraints, regulations, and a myriad of other factors. It focuses on how to seamlessly integrate systems, satisfy the customer’s requirements, and deliver within the budget and on time. The guide is essentially a “shopping list” of all the activities that could be conducted with tailoring guidelines to meet the needs of each project.

Software Architecture for Product Families

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

Software Architecture

"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

Software Architecture with Python

The Art of Systems Architecting

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.

The Software Architect Elevator

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

Software Systems Architecture

Data is at the center of many challenges in system design today. Difficult issues need to be figured out, such as scalability, consistency, reliability, efficiency, and maintainability. In addition, we have an overwhelming variety of tools, including relational databases, NoSQL datastores, stream or batch processors, and message brokers. What are the right choices for your application? How do you make sense

of all these buzzwords? In this practical and comprehensive guide, author Martin Kleppmann helps you navigate this diverse landscape by examining the pros and cons of various technologies for processing and storing data. Software keeps changing, but the fundamental principles remain the same. With this book, software engineers and architects will learn how to apply those ideas in practice, and how to make full use of data in modern applications. Peer under the hood of the systems you already use, and learn how to use and operate them more effectively Make informed decisions by identifying the strengths and weaknesses of different tools Navigate the trade-offs around consistency, scalability, fault tolerance, and complexity Understand the distributed systems research upon which modern databases are built Peek behind the scenes of major online services, and learn from their architectures

Project Management of Large Software-Intensive Systems

Before software engineering builds and installations can be implemented into software and/or systems integrations in military and aerospace programs, a comprehensive understanding of the software development life cycle is required. Covering all the development life cycle disciplines, Effective Methods for Software and Systems Integration explains h

Relating System Quality and Software Architecture

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.

Summary of a Workshop on Software-Intensive Systems and Uncertainty at Scale

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

Architecting Data-Intensive Applications

"This book addresses the complex issues associated with software engineering environment capabilities for designing real-time embedded software systems"--Provided by publisher.

Architecting Dependable Systems II

"This book addresses the complex issues associated with software engineering environment capabilities for designing real-time embedded software systems"--Provided by publisher.

Proceedings of the 11th European Conference on Software Architecture

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.

Effective Methods for Software and Systems Integration

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.

Documenting Software Architectures

This book addresses the challenges in the software engineering of variability-intensive systems. Variability-intensive systems can support different usage scenarios by accommodating different and unforeseen features and qualities. The book features academic and industrial contributions that discuss the challenges in developing, maintaining and evolving systems, cloud and mobile services for variability-intensive software systems and the scalability requirements they imply. The book explores software engineering approaches that can efficiently deal with variability-intensive systems as well as applications and use cases benefiting from variability-intensive systems.

Designing Data-Intensive Applications

Software architecture (SA) is one of the most significant areas of research and practice in software engineering. It has been shown that getting architecture of large-scale complex systems right is not only extremely important but hugely challenging. The increasing popularity and adoption of Agile Software Development (ASD) methods have brought architecture-centric methods and practices into question as agile followers tend to perceive architecture in the context of plan-driven software development. It is widely recognized that SA needs sufficient attention for successful development and evolution of software-intensive systems and services irrespective of the software development paradigm. Given the nature of the discipline, SA methods and approaches tend to be effort-intensive and heavyweight for certain kinds of projects. There is an increasing interest in finding ways to apply architecture-centric principles and practices in an Agile fashion—Agile architecting. A good understanding of architectural principles and approaches is a prerequisite to agile architecting. The aim of this chapter is to briefly describe the fundamental concepts, principles, and practices of architecture-centric approaches. These concepts, principles, and practices are expected to provide a reader with sufficient understanding of different aspects of SA and its related methods to combine them with ASD methods. We start with a brief discussion of the points that make architecture and agile approaches seemingly incompatible. Then we present and discuss some of the key aspects of architecture-centric approaches and techniques that need to be considered for use in ASD projects. We also provide an overview of some of the key practices that have been

recommended for successfully integrating architecture-centric approaches in ASD for developing large-scale, software-intensive systems.

Managing Trade-offs in Adaptable Software Architectures

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.

Evaluating Software Architectures

This book constitutes the refereed proceedings of the 12th European Conference on Software Architecture, ECSA 2018, held in Madrid, Spain, in September 2018. The 17 full papers presented together with 7 short papers were carefully reviewed and selected from 96 submissions. They are organized in topical sections as follows: Self-Adaptive Architectures, IoT Architectures, Embedded and Cyber-Physical Systems, Microservices Architectures, Service-Oriented Architectures, Architectural Design Decisions, Software Architecture in Practice.

Designing Software-Intensive Systems: Methods and Principles

Software Architecture for Big Data and the Cloud is designed to be a single resource that brings together research on how software architectures can solve the challenges imposed by building big data software systems. The challenges of big data on the software architecture can relate to scale, security, integrity, performance, concurrency, parallelism, and dependability, amongst others. Big data handling requires rethinking architectural solutions to meet functional and non-functional requirements related to volume, variety and velocity. The book's editors have varied and complementary backgrounds in requirements and architecture, specifically in software architectures for cloud and big data, as well as expertise in software engineering for cloud and big data. This book brings together work across different disciplines in software engineering, including work expanded from conference tracks and workshops led by the editors. Discusses systematic

and disciplined approaches to building software architectures for cloud and big data with state-of-the-art methods and techniques Presents case studies involving enterprise, business, and government service deployment of big data applications Shares guidance on theory, frameworks, methodologies, and architecture for cloud and big data

Software Architecture Knowledge Management

Architect and design data-intensive applications and, in the process, learn how to collect, process, store, govern, and expose data for a variety of use cases Key Features Integrate the data-intensive approach into your application architecture Create a robust application layout with effective messaging and data querying architecture Enable smooth data flow and make the data of your application intensive and fast Book Description Are you an architect or a developer who looks at your own applications gingerly while browsing through Facebook and applauding it silently for its data-intensive, yet fluent and efficient, behaviour? This book is your gateway to build smart data-intensive systems by incorporating the core data-intensive architectural principles, patterns, and techniques directly into your application architecture. This book starts by taking you through the primary design challenges involved with architecting data-intensive applications. You will learn how to implement data curation and data dissemination, depending on the volume of your data. You will then implement your application architecture one step at a

time. You will get to grips with implementing the correct message delivery protocols and creating a data layer that doesn't fail when running high traffic. This book will show you how you can divide your application into layers, each of which adheres to the single responsibility principle. By the end of this book, you will learn to streamline your thoughts and make the right choice in terms of technologies and architectural principles based on the problem at hand. What you will learn Understand how to envision a data-intensive system Identify and compare the non-functional requirements of a data collection component Understand patterns involving data processing, as well as technologies that help to speed up the development of data processing systems Understand how to implement Data Governance policies at design time using various Open Source Tools Recognize the anti-patterns to avoid while designing a data store for applications Understand the different data dissemination technologies available to query the data in an efficient manner Implement a simple data governance policy that can be extended using Apache Falcon Who this book is for This book is for developers and data architects who have to code, test, deploy, and/or maintain large-scale, high data volume applications. It is also useful for system architects who need to understand various non-functional aspects revolving around Data Intensive Systems.

ROMANCE ACTION & ADVENTURE MYSTERY & THRILLER BIOGRAPHIES & HISTORY CHILDREN'S YOUNG ADULT FANTASY HISTORICAL FICTION HORROR LITERARY FICTION NON-FICTION SCIENCE FICTION