

Software Architecture Foundations Theory And Practice

If you ally obsession such a referred **Software Architecture Foundations Theory And Practice** book that will meet the expense of you worth, acquire the agreed best seller from us currently from several preferred authors. If you want to witty books, lots of novels, tale, jokes, and more fictions collections are with launched, from best seller to one of the most current released.

You may not be perplexed to enjoy all ebook collections Software Architecture Foundations Theory And Practice that we will completely offer. It is not on the subject of the costs. Its more or less what you obsession currently. This Software Architecture Foundations Theory And Practice , as one of the most involved sellers here will unconditionally be among the best options to review.

Software Architecture Foundation - Gernot Starke 2021-05-25

This book covers everything you need to master the iSAQB® Certified Professional for Software Architecture - Foundation Level (CPSA-F) certification. This internationally renowned education and certification schema defines various learning path for practical software architects. This book concentrates on the foundation level examination. It explains and clarifies all 40+ learning goals of the CPSA-F® curriculum. In addition, you find step-by-step preparation guide for the examination. Please beware: This book is not meant as a replacement for existing software architecture books and courses, but strongly focusses on explaining and clarifying the iSAQB CPSA-F foundation.

Software Architecture - Zheng Qin 2009-02-07

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

Foundations for Architecting Data Solutions - Ted Malaska

2018-08-29

While many companies ponder implementation details such as distributed processing engines and algorithms for data analysis, this practical book takes a much wider view of big data development, starting with initial planning and moving diligently toward execution. Authors Ted Malaska and Jonathan Seidman guide you through the major components necessary to start, architect, and develop successful big data projects. Everyone from CIOs and COOs to lead architects and developers will explore a variety of big data architectures and applications, from massive data pipelines to web-scale applications. Each chapter addresses a piece of the software development life cycle and identifies patterns to maximize long-term success throughout the life of your project. Start the planning process by considering the key data project types Use guidelines to evaluate and select data management solutions Reduce risk related to technology, your team, and vague requirements Explore system interface design using APIs, REST, and pub/sub systems Choose the right distributed storage system for your big data system Plan and implement metadata collections for your data architecture Use data pipelines to ensure data integrity from source to final storage Evaluate the attributes of various engines for processing the data you collect

[The Essence of Software Engineering](#) - Volker Gruhn 2018-06-13

This open access book includes contributions by leading researchers and industry thought leaders on various topics related to the essence of software engineering and their application in industrial projects. It offers a broad overview of research findings dealing with current practical software engineering issues and also pointers to potential future developments. Celebrating the 20th anniversary of adesso AG, adesso gathered some of the pioneers of software engineering including Manfred Broy, Ivar Jacobson and Carlo Ghezzi at a special symposium, where they presented their thoughts about latest software engineering research and which are part of this book. This way it offers readers a concise overview of the essence of software engineering, providing valuable insights into the latest methodological research findings and adesso's experience applying these results in real-world projects.

Use Case Driven Object Modeling with UML Theory and Practice - Don Rosenberg 2008-06-28

Diagramming and process are important topics in today's software development world, as the UML diagramming language has come to be almost universally accepted. Yet process is necessary; by themselves, diagrams are of little use. *Use Case Driven Object Modeling with UML - Theory and Practice* combines the notation of UML with a lightweight but effective process - the ICONIX process - for designing and developing software systems. ICONIX has developed a growing following over the years. Sitting between the free-for-all of Extreme Programming and overly rigid processes such as RUP, ICONIX offers just enough structure to be successful.

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 Engineering for Variability Intensive Systems - Ivan Mistrik 2019-01-15

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.

Balancing Agility and Discipline - Barry W. Boehm 2004

"Being a certified bibliophile and a professional geek, I have more shelf space devoted to books on software methods than any reasonable human should possess. Balancing Agility and Discipline has a prominent place in that section of my library, because it has helped me sort through the noise and smoke of the current method wars." --From the Foreword by Grady Booch "This is an outstanding book on an emotionally complicated topic. I applaud the authors for the care with which they have handled the subject." --From the Foreword by Alistair Cockburn "The authors have done a commendable job of identifying five critical factors-- personnel, criticality, size, culture, and dynamism--for creating the right balance of flexibility and structure. Their thoughtful analysis will help developers who must sort through the agile-disciplined debate, giving them guidance to create the right mix for their projects." --From the Foreword by Arthur Pyster

Agility and discipline: These apparently opposite attributes are, in fact, complementary values in software development. Plan-driven developers must also be agile; nimble developers must also be disciplined. The key to success is finding the right balance between the two, which will vary from project to project according to the circumstances and risks involved. Developers, pulled toward opposite ends by impassioned arguments, ultimately must learn how to give each value its due in their particular situations. Balancing Agility and Discipline sweeps aside the rhetoric, drills down to the operational core concepts, and presents a constructive approach to defining a balanced software development strategy. The authors expose the bureaucracy and stagnation that mark discipline without agility, and liken agility without discipline to unbridled and fruitless enthusiasm. Using a day in the life of two development teams and ground-breaking case studies, they illustrate the differences and similarities between agile and plan-driven methods, and show that the best development strategies have ways to combine both attributes. Their analysis is both objective and grounded, leading finally to clear and practical guidance for all software professionals--showing how to locate the sweet spot on the agility-discipline continuum for any given project.

0321186125B10212003

SOFTWARE ARCHITECTURE: FOUNDATIONS, THEORY, AND PRACTICE - Richard N. Taylor 2010-09-01

Market_Desc: Software Architects and Systems Engineers Special Features: · Approaches software architecture from a rigorous systems view while also including real-world perspectives· Incorporates the very latest research results as well as some material that is completely new· Focuses on practical principles and insights· Addresses a wide range of topics· Contains a CD-ROM that includes tools and technologies that enable readers to apply many of the key techniques discussed About The Book: Designed for software engineers, Software Architecture is the first With CD book to cover the maturing and important area of software engineering and development. It approaches software architecture from a rigorous systems view while also including real-world perspectives. The chapters incorporate the very latest research results as well as some material that is completely new. Many of the key techniques discussed are supported by tools and technologies included on the book's CD-ROM. Through this package, software engineers will be armed with concise, practical strategies for designing, implementing, and evolving successful systems using software architecture.

Building Software for Simulation - James J. Nutaro 2011-03-23

Fundamentals of Turbulent and Multiphase Combustion Detailed coverage of advanced combustion topics from the author of Principles of Combustion, Second Edition Turbulence, turbulent combustion, and multiphase reacting flows have become major research topics in recent decades due to their application across diverse fields, including energy, environment, propulsion, transportation, industrial safety, and nanotechnology. Most of the knowledge accumulated from this research has never been published in book form—until now. Fundamentals of Turbulent and Multiphase Combustion presents up-to-date, integrated coverage of the fundamentals of turbulence, combustion, and multiphase phenomena along with useful experimental techniques, including non-intrusive, laser-based measurement techniques, providing a firm background in both contemporary and classical approaches. Beginning

with two full chapters on laminar premixed and non-premixed flames, this book takes a multiphase approach, beginning with more common topics and moving on to higher-level applications. In addition, *Fundamentals of Turbulent and Multiphase Combustion: Addresses seven basic topical areas in combustion and multiphase flows, including laminar premixed and non-premixed flames, theory of turbulence, turbulent premixed and non-premixed flames, and multiphase flows* Covers spray atomization and combustion, solid-propellant combustion, homogeneous propellants, nitramines, reacting boundary-layer flows, single energetic particle combustion, and granular bed combustion Provides experimental setups and results whenever appropriate Supported with a large number of examples and problems as well as a solutions manual, *Fundamentals of Turbulent and Multiphase Combustion* is an important resource for professional engineers and researchers as well as graduate students in mechanical, chemical, and aerospace engineering.

System Engineering Analysis, Design, and Development - Charles S. Wasson 2015-11-16

Praise for the first edition: "This excellent text will be useful to every system engineer (SE) regardless of the domain. It covers ALL relevant SE material and does so in a very clear, methodical fashion. The breadth and depth of the author's presentation of SE principles and practices is outstanding." -Philip Allen This textbook presents a comprehensive, step-by-step guide to System Engineering analysis, design, and development via an integrated set of concepts, principles, practices, and methodologies. The methods presented in this text apply to any type of human system -- small, medium, and large organizational systems and system development projects delivering engineered systems or services across multiple business sectors such as medical, transportation, financial, educational, governmental, aerospace and defense, utilities, political, and charity, among others. Provides a common focal point for "bridging the gap" between and unifying System Users, System Acquirers, multi-discipline System Engineering, and Project, Functional, and Executive Management education, knowledge,

and decision-making for developing systems, products, or services Each chapter provides definitions of key terms, guiding principles, examples, author's notes, real-world examples, and exercises, which highlight and reinforce key SE&D concepts and practices Addresses concepts employed in Model-Based Systems Engineering (MBSE), Model-Driven Design (MDD), Unified Modeling Language (UMLTM) / Systems Modeling Language (SysMLTM), and Agile/Spiral/V-Model Development such as user needs, stories, and use cases analysis; specification development; system architecture development; User-Centric System Design (UCSD); interface definition & control; system integration & test; and Verification & Validation (V&V) Highlights/introduces a new 21st Century Systems Engineering & Development (SE&D) paradigm that is easy to understand and implement. Provides practices that are critical staging points for technical decision making such as Technical Strategy Development; Life Cycle requirements; Phases, Modes, & States; SE Process; Requirements Derivation; System Architecture Development, User-Centric System Design (UCSD); Engineering Standards, Coordinate Systems, and Conventions; et al. Thoroughly illustrated, with end-of-chapter exercises and numerous case studies and examples, *Systems Engineering Analysis, Design, and Development, Second Edition* is a primary textbook for multi-discipline, engineering, system analysis, and project management undergraduate/graduate level students and a valuable reference for professionals.

Feedback Control for Computer Systems - Philipp K. Janert 2013-09-19

How can you take advantage of feedback control for enterprise programming? With this book, author Philipp K. Janert demonstrates how the same principles that govern cruise control in your car also apply to data center management and other enterprise systems. Through case studies and hands-on simulations, you'll learn methods to solve several control issues, including mechanisms to spin up more servers automatically when web traffic spikes. Feedback is ideal for controlling large, complex systems, but its use in software engineering raises unique

issues. This book provides basic theory and lots of practical advice for programmers with no previous background in feedback control. Learn feedback concepts and controller design Get practical techniques for implementing and tuning controllers Use feedback “design patterns” for common control scenarios Maintain a cache’s “hit rate” by automatically adjusting its size Respond to web traffic by scaling server instances automatically Explore ways to use feedback principles with queueing systems Learn how to control memory consumption in a game engine Take a deep dive into feedback control theory

Understanding Context - Andrew Hinton 2014-12-02

To make sense of the world, we’re always trying to place things in context, whether our environment is physical, cultural, or something else altogether. Now that we live among digital, always-networked products, apps, and places, context is more complicated than ever—starting with “where” and “who” we are. This practical, insightful book provides a powerful toolset to help information architects, UX professionals, and web and app designers understand and solve the many challenges of contextual ambiguity in the products and services they create. You’ll discover not only how to design for a given context, but also how design participates in making context. Learn how people perceive context when touching and navigating digital environments See how labels, relationships, and rules work as building blocks for context Find out how to make better sense of cross-channel, multi-device products or services Discover how language creates infrastructure in organizations, software, and the Internet of Things Learn models for figuring out the contextual angles of any user experience

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.

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

Microservice Architecture - Irakli Nadareishvili 2016-07-18

Have you heard about the tremendous success Amazon and Netflix have had by switching to a microservice architecture? Are you wondering how this can benefit your company? Or are you skeptical about how it might work? If you've answered yes to any of these questions, this practical book will benefit you. You'll learn how to take advantage of the microservice architectural style for building systems, and learn from the experiences of others to adopt and execute this approach most successfully.

Foundations of Software Engineering - Ashfaque Ahmed 2016-08-25

The best way to learn software engineering is by understanding its core and peripheral areas. Foundations of Software Engineering provides in-depth coverage of the areas of software engineering that are essential for becoming proficient in the field. The book devotes a complete chapter to each of the core areas. Several peripheral areas are also explained by assigning a separate chapter to each of them. Rather than using UML or other formal notations, the content in this book is explained in easy-to-understand language. Basic programming knowledge using an object-oriented language is helpful to understand the material in this book. The knowledge gained from this book can be readily used in other relevant courses or in real-world software development environments. This textbook educates students in software engineering principles. It covers almost all facets of software engineering, including requirement

engineering, system specifications, system modeling, system architecture, system implementation, and system testing. Emphasizing practical issues, such as feasibility studies, this book explains how to add and develop software requirements to evolve software systems. This book was written after receiving feedback from several professors and software engineers. What resulted is a textbook on software engineering that not only covers the theory of software engineering but also presents real-world insights to aid students in proper implementation. Students learn key concepts through carefully explained and illustrated theories, as well as concrete examples and a complete case study using Java. Source code is also available on the book's website. The examples and case studies increase in complexity as the book progresses to help students build a practical understanding of the required theories and applications.

Game Engine Architecture, Third Edition - Jason Gregory 2018-07-20

In this new and improved third edition of the highly popular Game Engine Architecture, Jason Gregory draws on his nearly two decades of experience at Midway, Electronic Arts and Naughty Dog to present both the theory and practice of game engine software development. In this book, the broad range of technologies and techniques used by AAA game studios are each explained in detail, and their roles within a real industrial-strength game engine are illustrated. New to the Third Edition This third edition offers the same comprehensive coverage of game engine architecture provided by previous editions, along with updated coverage of: computer and CPU hardware and memory caches, compiler optimizations, C++ language standardization, the IEEE-754 floating-point representation, 2D user interfaces, plus an entirely new chapter on hardware parallelism and concurrent programming. This book is intended to serve as an introductory text, but it also offers the experienced game programmer a useful perspective on aspects of game development technology with which they may not have deep experience. As always, copious references and citations are provided in this edition, making it an excellent jumping off point for those who wish to dig deeper into any particular aspect of the game development process. Key

Features Covers both the theory and practice of game engine software development Examples are grounded in specific technologies, but discussion extends beyond any particular engine or API. Includes all mathematical background needed. Comprehensive text for beginners and also has content for senior engineers.

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.

Software Engineering Design - Carlos Otero 2012-08-23

Taking a learn-by-doing approach, *Software Engineering Design: Theory and Practice* uses examples, review questions, chapter exercises, and case study assignments to provide students and practitioners with the understanding required to design complex software systems. Explaining the concepts that are immediately relevant to software designers, it begins with a review of software design fundamentals. The text presents a formal top-down design process that consists of several design activities with varied levels of detail, including the macro-, micro-, and construction-design levels. As part of the top-down approach, it provides in-depth coverage of applied architectural, creational, structural, and behavioral design patterns. For each design issue covered, it includes a step-by-step breakdown of the execution of the design solution, along with an evaluation, discussion, and justification for using that particular solution. The book outlines industry-proven software design practices for leading large-scale software design efforts, developing reusable and high-quality software systems, and producing technical and customer-driven design documentation. It also: Offers one-stop guidance for mastering the Software Design & Construction sections of the official Software Engineering Body of Knowledge (SWEBOK®) Details a collection of standards and guidelines for structuring high-quality code Describes techniques for analyzing and evaluating the quality of software designs Collectively, the text supplies comprehensive coverage of the software design concepts students will need to succeed as professional design leaders. The section on engineering leadership for software

designers covers the necessary ethical and leadership skills required of software developers in the public domain. The section on creating software design documents (SDD) familiarizes students with the software design notations, structural descriptions, and behavioral models required for SDDs. Course notes, exercises with answers, online resources, and an instructor's manual are available upon qualified course adoption.

Instructors can contact the author about these resources via the author's website: <http://softwareengineeringdesign.com/>

Software Architecture - Richard N. Taylor 2008

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 focu.

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.

Lean Architecture - James O. Coplien 2011-01-06

More and more Agile projects are seeking architectural roots as they struggle with complexity and scale - and they're seeking lightweight ways to do it Still seeking? In this book the authors help you to find your own path Taking cues from Lean development, they can help steer your project toward practices with longstanding track records Up-front architecture? Sure. You can deliver an architecture as code that compiles and that concretely guides development without bogging it down in a mass of documents and guesses about the implementation

Documentation? Even a whiteboard diagram, or a CRC card, is documentation: the goal isn't to avoid documentation, but to document just the right things in just the right amount Process? This all works within the frameworks of Scrum, XP, and other Agile approaches

Architectures for Adaptive Software Systems - Raffaella Mirandola 2009-06-08

Much of a software architect's life is spent designing software systems to meet a set of quality requirements. General software quality attributes include scalability, security, performance or reliability. Quality attribute requirements are part of an application's non-functional requirements, which capture the many facets of how the functional - requirements of an application are achieved. Understanding, modeling and continually evaluating quality attributes throughout a project lifecycle are all complex engineering tasks whichcontinuetochallenge the softwareengineeringsscience community. While we search for improved approaches, methods, formalisms and tools that are usable in practice and can scale to large systems, the complexity of the applications that the so-ware industry is challenged to build is ever increasing. Thus, as a research community, there is little opportunity for us to rest on our laurels, as our innovations that address new aspects of system complexity must be deployed and validated. To this end the 5th

International Conference on the Quality of Software Architectures (QoSA) 2009 focused on architectures for adaptive software systems. Modern software systems must often recon guretheir structure and behavior to respond to c- tinuous changes in requirements and in their execution environment. In these settings, quality models are helpful at an architectural level to guide systematic model-driven software development strategies by evaluating the impact of competing architectural choices.

The Art of Systems Architecting, Third Edition - Mark W. Maier 2009-01-06

If engineering is the art and science of technical problem solving, systems architecting happens when you don't yet know what the problem is. The third edition of a highly respected bestseller, The Art of Systems Architecting provides in-depth coverage of the least understood part of systems design: moving from a vague concept and limited resources to a satisfactory and feasible system concept and an executable program. The book provides a practical, heuristic approach to the "art" of systems architecting. It provides methods for embracing, and then taming, the growing complexity of modern systems. New in the Third Edition: Five major case studies illustrating successful and unsuccessful practices Information on architecture frameworks as standards for architecture descriptions New methods for integrating business strategy and architecture and the role of architecture as the technical embodiment of strategy Integration of process guidance for organizing and managing architecture projects Updates to the rapidly changing fields of software and systems-of-systems architecture Organization of heuristics around a simple and practical process model A Practical Heuristic Approach to the Art of Systems Architecting Extensively rewritten to reflect the latest developments, the text explains how to create a system from scratch, presenting invention/design rules together with clear explanations of how to use them. The author supplies practical guidelines for avoiding common systematic failures while implementing new mandates. He uses a heuristics-based approach that provides an organized attack on very ill-structured engineering problems. Examining architecture as more than a

set of diagrams and documents, but as a set of decisions that either drive a system to success or doom it to failure, the book provide methods for integrating business strategy with technical architectural decision making.

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

Ernst Denert Award for Software Engineering 2019 - Michael Felderer 2020-12-03

This open access book provides an overview of the dissertations of the five nominees for the Ernst Denert Award for Software Engineering in

2019. The prize, kindly sponsored by the Gerlind & Ernst Denert Stiftung, is awarded for excellent work within the discipline of Software Engineering, which includes methods, tools and procedures for better and efficient development of high quality software. An essential requirement for the nominated work is its applicability and usability in industrial practice. The book contains five papers describing the works by Sebastian Baltes (U Trier) on Software Developers'Work Habits and Expertise, Timo Greifenberg's thesis on Artefaktbasierte Analyse modellgetriebener Softwareentwicklungsprojekte, Marco Konersmann's (U Duisburg-Essen) work on Explicitly Integrated Architecture, Marija Selakovic's (TU Darmstadt) research about Actionable Program Analyses for Improving Software Performance, and Johannes Späth's (Paderborn U) thesis on Synchronized Pushdown Systems for Pointer and Data-Flow Analysis - which actually won the award. The chapters describe key findings of the respective works, show their relevance and applicability to practice and industrial software engineering projects, and provide additional information and findings that have only been discovered afterwards, e.g. when applying the results in industry. This way, the book is not only interesting to other researchers, but also to industrial software professionals who would like to learn about the application of state-of-the-art methods in their daily work.

Agile Software Architecture - Muhammad Ali Babar 2013-11-27
Agile software development approaches have had significant impact on industrial software development practices. Today, agile software development has penetrated to most IT companies across the globe, with an intention to increase quality, productivity, and profitability. Comprehensive knowledge is needed to understand the architectural challenges involved in adopting and using agile approaches and industrial practices to deal with the development of large, architecturally challenging systems in an agile way. Agile Software Architecture focuses on gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox. Readers will learn how agile and architectural cultures can co-exist and support each other according to the context.

Moreover, this book will also provide useful leads for future research in architecture and agile to bridge such gaps by developing appropriate approaches that incorporate architecturally sound practices in agile methods. Presents a consolidated view of the state-of-art and state-of-practice as well as the newest research findings Identifies gaps in the requirements of applying architecture-centric approaches and principles of agile software development and demystifies the agile architecture paradox Explains whether or not and how agile and architectural cultures can co-exist and support each other depending upon the context Provides useful leads for future research in both architecture and agile to bridge such gaps by developing appropriate approaches, which incorporate architecturally sound practices in agile methods

Managing Systems and IT Projects - Frank Tsui 2011-08-24

This book is designed for software engineering students and project management professional in the IT and software industry. It focuses on the four phases of management -- planning, organizing, monitoring, and adjusting (POMA) -- and tailors to systems and applications on software projects. The tasks and techniques utilized in each of the POMA management phases are discussed with specific software engineering and IT related examples. Drawing from years of experience in the industry, the author presents material within a framework of real-world examples and exercises that help readers apply new concepts to everyday situations.

Foundations of Software and System Performance Engineering - André B. Bondi 2014-08-08

"If this book had been available to Healthcare.gov's contractors, and they read and followed its life cycle performance processes, there would not have been the enormous problems apparent in that application. In my 40+ years of experience in building leading-edge products, poor performance is the single most frequent cause of the failure or cancellation of software-intensive projects. This book provides techniques and skills necessary to implement performance engineering at the beginning of a project and manage it throughout the product's life cycle. I cannot recommend it highly enough." - Don Shafer, CSDP,

Technical Fellow, Athens Group, LLC Poor performance is a frequent cause of software project failure. Performance engineering can be extremely challenging. In *Foundations of Software and System Performance Engineering*, leading software performance expert Dr. André Bondi helps you create effective performance requirements up front, and then architect, develop, test, and deliver systems that meet them. Drawing on many years of experience at Siemens, AT&T Labs, Bell Laboratories, and two startups, Bondi offers practical guidance for every software stakeholder and development team participant. He shows you how to define and use metrics; plan for diverse workloads; evaluate scalability, capacity, and responsiveness; and test both individual components and entire systems. Throughout, Bondi helps you link performance engineering with everything else you do in the software life cycle, so you can achieve the right performance-now and in the future-at lower cost and with less pain. This guide will help you

- Mitigate the business and engineering risk associated with poor system performance
- Specify system performance requirements in business and engineering terms
- Identify metrics for comparing performance requirements with actual performance
- Verify the accuracy of measurements
- Use simple mathematical models to make predictions, plan performance tests, and anticipate the impact of changes to the system or the load placed upon it
- Avoid common performance and scalability mistakes
- Clarify business and engineering needs to be satisfied by given levels of throughput and response time
- Incorporate performance engineering into agile processes
- Help stakeholders of a system make better performance-related decisions
- Manage stakeholders' expectations about system performance throughout the software life cycle, and deliver a software product with quality performance

André B. Bondi is a senior staff engineer at Siemens Corp., Corporate Technologies in Princeton, New Jersey. His specialties include performance requirements, performance analysis, modeling, simulation, and testing. Bondi has applied his industrial and academic experience to the solution of performance issues in many problem domains. In addition to holding a doctorate in computer science and a master's in statistics, he is a Certified Scrum Master.

The Art of Failure - Jesper Juul 2013-02-22

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

Exercises in Programming Style - Cristina Videira Lopes 2020-08-30
Using a simple computational task (term frequency) to illustrate different programming styles, *Exercises in Programming Style* helps readers understand the various ways of writing programs and designing systems. It is designed to be used in conjunction with code provided on an online repository. The book complements and explains the raw code in a way that is accessible to anyone who regularly practices the art of programming. The first edition was honored as an ACM Notable Book

and praised as "The best programming book of the decade." This new edition will retain the same presentation, but the entire book will be upgraded to Python 3, and a new section will be added on neural network styles. The book contains 33 different styles for writing the term frequency task. The styles are grouped into nine categories: historical, basic, function composition, objects and object interactions, reflection and metaprogramming, adversity, data-centric, concurrency, and interactivity. The author verbalizes the constraints in each style and explains the example programs. Each chapter first presents the constraints of the style, next shows an example program, and then gives a detailed explanation of the code. Most chapters also have sections focusing on the use of the style in systems design as well as sections describing the historical context in which the programming style emerged.

Design and Use of Software Architectures - Jan Bosch 2000
A practical guide to designing and implementing software architectures.
Software Testing and Quality Assurance - Kshirasagar Naik
2011-09-23

A superior primer on software testing and quality assurance, from integration to execution and automation This important new work fills the pressing need for a user-friendly text that aims to provide software engineers, software quality professionals, software developers, and students with the fundamental developments in testing theory and common testing practices. *Software Testing and Quality Assurance: Theory and Practice* equips readers with a solid understanding of: Practices that support the production of quality software Software testing techniques Life-cycle models for requirements, defects, test cases, and test results Process models for units, integration, system, and acceptance testing How to build test teams, including recruiting and retaining test engineers Quality Models, Capability Maturity Model, Testing Maturity Model, and Test Process Improvement Model Expertly balancing theory with practice, and complemented with an abundance of pedagogical tools, including test questions, examples, teaching suggestions, and chapter summaries, this book is a valuable, self-

contained tool for professionals and an ideal introductory text for courses in software testing, quality assurance, and software engineering.

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.

Foundations of Deep Reinforcement Learning - Laura Graesser
2019-11-20

The Contemporary Introduction to Deep Reinforcement Learning that Combines Theory and Practice Deep reinforcement learning (deep RL) combines deep learning and reinforcement learning, in which artificial

agents learn to solve sequential decision-making problems. In the past decade deep RL has achieved remarkable results on a range of problems, from single and multiplayer games—such as Go, Atari games, and DotA 2—to robotics. Foundations of Deep Reinforcement Learning is an introduction to deep RL that uniquely combines both theory and implementation. It starts with intuition, then carefully explains the theory of deep RL algorithms, discusses implementations in its companion software library SLM Lab, and finishes with the practical details of getting deep RL to work. This guide is ideal for both computer science students and software engineers who are familiar with basic machine learning concepts and have a working understanding of Python.

Understand each key aspect of a deep RL problem Explore policy- and value-based algorithms, including REINFORCE, SARSA, DQN, Double DQN, and Prioritized Experience Replay (PER) Delve into combined algorithms, including Actor-Critic and Proximal Policy Optimization (PPO) Understand how algorithms can be parallelized synchronously and asynchronously Run algorithms in SLM Lab and learn the practical implementation details for getting deep RL to work Explore algorithm benchmark results with tuned hyperparameters Understand how deep RL environments are designed Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

[Guide to the Software Engineering Body of Knowledge \(Swebok\(r\)\)](#) - IEEE Computer Society 2014

In the Guide to the Software Engineering Body of Knowledge (SWEBOK(R) Guide), the IEEE Computer Society establishes a baseline for the body of knowledge for the field of software engineering, and the work supports the Society's responsibility to promote the advancement of both theory and practice in this field. It should be noted that the Guide does not purport to define the body of knowledge but rather to serve as a compendium and guide to the knowledge that has been developing and evolving over the past four decades. Now in Version 3.0, the Guide's 15 knowledge areas summarize generally accepted topics and list references for detailed information. The editors for Version 3.0 of the

SWEBOK(R) Guide are Pierre Bourque (Ecole de technologie superieure (ETS), Universite du Quebec) and Richard E. (Dick) Fairley (Software and Systems Engineering Associates (S2EA)).

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.

SOFTWARE DESIGN, ARCHITECTURE AND ENGINEERING - BHATT, PRAMOD CHANDRA P. 2021-07-01

This textbook aims to prepare students, as well as, practitioners for software design and production. Keeping in mind theory and practice, the book keeps a balance between theoretical foundations and practical considerations. The book by and large meets the requirements of students at all levels of computer science and engineering/information technology for their Software design and Software engineering courses.

The book begins with concepts of data and object. This helps in exploring the rationale that guide high level programming language (HLL) design and object oriented frameworks. Once past this post, the book moves on to expand on software design concerns. The book emphasizes the centrality of Parnas's separation of concerns in evolving software designs and architecture. The book extensively explores modelling frameworks such as Unified Modelling Language (UML) and Petri net based methods. Next, the book covers architectural principles and software engineering practices such as Agile - emphasizing software testing during development. It winds up with case studies demonstrating how systems evolve from basic concepts to final products for quality software designs.

TARGET AUDIENCE • Undergraduate/postgraduate students of Computer Science and Engineering, and Information Technology • Postgraduate students of Software Engineering/Software Systems

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