

## Evaluating Software Architectures Methods And Case Studies

This book reports on recent advances in software engineering research and practice. Divided into 15 chapters, it addresses: languages and tools; development processes; modelling, simulation and verification; and education. In the first category, the book includes chapters on domain-specific languages, software complexity, testing and tools. In the second, it reports on test-driven development, processing of business rules, and software management. In turn, subsequent chapters address modelling, simulation and verification of real-time systems, mobile systems and computer networks, and a scrum-based framework. The book was written by researchers and practitioners, the goal being to achieve a synergistic combination of research results achieved in academia and best practices used in the industry, and to provide a valuable reference guide for both groups.

This book constitutes the refereed proceedings of the 5th European Conference on Software Architecture, ECSA 2011, held in Essen, Germany, in September 2011. The 13 revised full papers presented together with 24 emerging research papers, and 7 research challenge poster papers were carefully reviewed and selected from over 100 submissions. The papers are organized in topical sections on requirements and software architectures; software architecture, components, and compositions; quality attributes and software architectures; software product line architectures; architectural models, patterns and styles; short papers; process and management of architectural decisions; software architecture run-time aspects; ADLs and metamodels; and services and software architectures.

This title provides a forum where expert insights are presented on the subject of linking three current phenomena: software evolution, UML and XML.

This book constitutes the refereed proceedings of the 2nd European Workshop on Software Architecture, EWSA 2004, held in Pisa, Italy in June 2005. The 12 revised full research papers, one revised case study, and four revised position papers presented together with one invited presentation on ongoing European projects on software architectures were carefully reviewed and selected from 41 submissions. All current aspects of software architectures are addressed ranging from foundational and methodological issues to application issues of practical relevance.

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

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.

Enterprise solutions have emerged as promising tools for integrating and extending business processes across business functions. Supplying a clear and comprehensive introduction to the field, this book provides a detailed description of enterprise information integration—from the development of enterprise systems to extended enterprise information integration in supply

chain environments. Enterprise Integration and Information Architecture: A Systems Perspective on Industrial Information Integration explains how to improve industrial information integration through the application of a systems approach. Describing how systems science is impacting current research in industrial information integration, it covers enterprise architecture, information architecture for enterprises, business process/work flow modeling, and enterprise information integration. Covering the emergence, growth, and extension of integrated enterprise systems, the book provides you with various perspectives of modern enterprise solutions. It introduces the critical concepts of ERP, industry-oriented enterprise resource planning, and entire resource planning. It also provides guidance on how to transition from extended enterprise integration in a supply chain environment to systems-based enterprise architecture, enterprise modeling, and enterprise modeling in a supply chain environment. The book proposes a new information architecture for enterprise and supply chain management. It presents modeling and integration information flows for enterprise information integration, together with the Internet of Things (IoT). It also explores the theory and methods of industrial information integration including integration approaches and enterprise application integration. Complete with numerous examples of extended enterprise integration in actual supply chain environments, the book illustrates the critical issues that arise in professional practice and also explores emerging trends in enterprise integration and its information architecture

Embedded systems are ubiquitous. They appear in cell phones, microwave ovens, refrigerators, consumer electronics, cars, and jets. Some of these embedded systems are safety- or security-critical such as in medical equipment, nuclear plants, and X-by-wire control systems in naval, ground and aerospace transportation vehicles. With the continuing shift from hardware to software, embedded systems are increasingly dominated by embedded software. Embedded software is complex. Its engineering inherently involves a multidisciplinary interplay with the physics of the embedding system or environment. Embedded software also comes in ever larger quantity and diversity. The next generation of premium automobiles will carry around one gigabyte of binary code. The proposed US DDX submarine is effectively a floating embedded software system, comprising 30 billion lines of code written in over 100 programming languages. Embedded software is expensive. Cost estimates are quoted at around US\$15– 30 per line (from commencement to shipping). In the defense realm, costs can range up to \$100, while for highly critical applications, such as the Space Shuttle, the cost per line approximates \$1,000. In view of the exponential increase in complexity, the projected costs of future embedded software are staggering.

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

Software Quality Assurance in Large Scale and Complex Software-intensive Systems presents novel and high-quality research related approaches that relate the quality of software architecture to system requirements, system architecture and enterprise-architecture, or software testing. Modern software has become complex and adaptable due to the emergence of globalization and new software technologies, devices and networks. These changes challenge both traditional software quality assurance techniques and software engineers to ensure

software quality when building today (and tomorrow's) adaptive, context-sensitive, and highly diverse applications. This edited volume presents state of the art techniques, methodologies, tools, best practices and guidelines for software quality assurance 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: quality attributes of system/software architectures; aligning enterprise, system, and software architecture from the point of view of total quality; design decisions and their influence on the quality of system/software architecture; methods and processes for evaluating architecture quality; quality assessment of legacy systems and third party applications; lessons learned and empirical validation of theories and frameworks on architectural quality; empirical validation and testing for assessing architecture quality. Focused on quality assurance at all levels of software design and development Covers domain-specific software quality assurance issues e.g. for cloud, mobile, security, context-sensitive, mash-up and autonomic systems Explains likely trade-offs from design decisions in the context of complex software system engineering and quality assurance Includes practical case studies of software quality assurance for complex, adaptive and context-critical systems

Why have a book about the relation between requirements and software architecture? Understanding the relation between requirements and architecture is important because the requirements, be they explicit or implicit, represent the function, whereas the architecture determines the form. While changes to a set of requirements may impact on the realization of the architecture, choices made for an architectural solution may impact on requirements, e.g., in terms of revising functional or non-functional requirements that cannot actually be met. Although research in both requirements engineering and software architecture is quite active, it is in their combination that understanding is most needed and actively sought. Presenting the current state of the art is the purpose of this book. The editors have divided the contributions into four parts: Part 1 "Theoretical Underpinnings and Reviews" addresses the issue of requirements change management in architectural design through traceability and reasoning. Part 2 "Tools and Techniques" presents approaches, tools, and techniques for bridging the gap between software requirements and architecture. Part 3 "Industrial Case Studies" then reports industrial experiences, while part 4 on "Emerging Issues" details advanced topics such as synthesizing architecture from requirements or the role of middleware in architecting for non-functional requirements. The final chapter is a conclusions chapter identifying key contributions and outstanding areas for future research and improvement of practice. The book is targeted at academic and industrial researchers in requirements engineering or software architecture. Graduate students specializing in these areas as well as advanced

professionals in software development will also benefit from the results and experiences presented in this volume.

The European Conference on Software Architecture (ECSA) is the premier European conference dedicated to the field of software architecture, covering all architectural features of software engineering. It is the follow-up of a successful series of European workshops on software architecture held in the UK in 2004 (Springer LNCS 3047), Italy in 2005 (Springer LNCS 3527), and France in 2006 (Springer LNCS 4344). It evolved into a series of European conferences whose first edition was ECSA 2007, held in Madrid, Spain during September 24–26, 2007 (Springer LNCS 4758). This year's conference was held at the beautiful Coral Beach Hotel and Resort near Paphos in Cyprus. As with the previous versions of the conference, ECSA 2008 (Springer LNCS 5292) provided an international forum for researchers and practitioners from academia and industry to present innovative research and to discuss a wide range of topics in the area of software architecture. It focused on formalisms, technologies, and processes for describing, verifying, validating, transforming, building, and evolving software systems. Covered topics included architecture modelling, architecture description languages, architectural aspects, architecture analysis, transformation and synthesis, architecture evolution, quality attributes, model-driven engineering, built-in testing and architecture-based support for component-based and service-oriented systems. The conference attracted paper submissions from 29 countries (Australia, Belgium, Brazil, Canada, China, Chile, Denmark, Finland, France, Germany, Greece, Ireland, Italy, Lithuania, Luxembourg, Malta, The Netherlands, Norway, Pakistan, Peru, - land, Portugal, Romania, South Africa, Spain, Turkey, the UK, USA, and Venezuela).

With SPLC 2005 we celebrated the formation of a new conference series, the International Software Product Line Conference (SPLC) which results from the "unification" of the former series of three SPLC (Software Product Line) Conferences launched in 2000 in the USA, and the former series of 7ve PFE (Product Family Engineering) Workshops started in 1996 in Europe. SPLC is now the premier forum for the growing community of software product line practitioners, researchers, and educators. SPLC offers a unique opportunity to present and discuss the most recent experiences, ideas, innovations, trends, and concerns in the area of software product line engineering and to build an international network of product line champions. An international SPLC Steering Committee has been established and it is the wish of this committee that from 2005 on, the SPLC conference will be held yearly in Europe, America, or Asia. The technical program of SPLC 2005 included: – two keynotes from David Weiss (Avaya, USA) and Jan Bosch (Nokia, Finland), both leading experts with academic and industrial insights; – 17 full and 3 short research papers organized around the following themes: feature modeling, re-engineering, strategies, validation, scoping and architecture, and product derivation; – eight experience reports describing commercial application of product line practices; – two panels

focused on special topics in product line practice and product line research; – tool demonstrations; –

aHallofFamesessionthatcontinuedtheSPLCtraditioninaslightlyrevised format. In addition, the technical program was preceded by a tutorial and workshop day that included ten half-day tutorials presented by well-recognized experts and ?ve workshops on speci?c areas of product line research.

This innovative book uncovers all the steps readers should follow in order to build successful software and systems With the help of numerous examples, Albin clearly shows how to incorporate Java, XML, SOAP, ebXML, and BizTalk when designing true distributed business systems Teaches how to easily integrate design patterns into software design Documents all architectures in UML and presents code in either Java or C++

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.

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

This book systematically identifies the lack of methodological support for development of requirements and software architecture in the state-of-the-art. To overcome this deficiency, the QuaDRA framework is proposed as a problem-oriented approach. It provides an instantiation of the Twin Peaks model for supporting the intertwining relationship of requirements and software architecture. QuaDRA includes several structured methods which guide software engineers in quality- and pattern-based co-development of requirements and early design alternatives in an iterative and concurrent manner.

This book illustrates the role of software architecture and its application in business. The author describes enterprise architecture along with business architecture to show the role of software architecture in both areas. The place of software architecture in business is outlined from many perspectives in this context. The book outlines quality attributes and how managers can use software architecture to build high quality products. Topics include business software architecture, dealing with qualities, achieving quality attributes, managing business qualities, software product line, Internet of Things (IOT), and Service Oriented Business Architecture. The book is intended to benefit students, researchers, software architects, and business architects. Provides quick and easy access to all the important aspects of software architecture in business; Highlights a wide variety of concepts of software architecture in a straightforward manner, for students, practitioners, or architects; Presents different applications of software architecture in business.

Welcome to the European Conference on Software Architecture (ECSA), which is the premier European software engineering conference. ECSA provides researchers and practitioners with a platform to present and discuss the most recent, innovative, and significant findings and experiences in the field of software architecture research and practice. The fourth edition of ECSA was built upon a history of a successful series of European workshops on software architecture held from 2004 through 2006 and a series of European software architecture conferences from 2007 through 2009. The last

## Bookmark File PDF Evaluating Software Architectures Methods And Case Studies

ECSA was merged with the 8th Working IEEE/IFIP Conference on Software Architecture (WICSA). Apart from the traditional technical program consisting of keynote talks, a main - search track, and a poster session, the scope of the ECSA 2010 was broadened to incorporate other tracks such as an industry track, doctoral symposium track, and a tool demonstration track. In addition, we also offered several workshops and tutorials on diverse topics related to software architecture. We received more than 100 submissions in the three main categories: full research and experience papers, emerging research papers, and research challenges papers. The conference attracted papers (co-)authored by researchers, practitioners, and academics from 30 countries (Algeria, Australia, Austria, Belgium, Brazil, Canada, Chile, China, Colombia, Czech Republic, Denmark, Finland, France, Germany, Hong Kong, I- land, India, Ireland, Israel, Italy, The Netherlands, Poland, Portugal, Romania, Spain, Sweden, Switzerland, Tunisia, United Kingdom, United States).

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 Methodology explores the theory of software architecture, with particular emphasis on general design principles rather than specific methods. This book provides in depth coverage of large scale software systems and the handling of their design problems. It will help students gain an understanding of the general theory of design methodology, and especially in analysing and evaluating software architectural designs, through the use of case studies and examples, whilst broadening their knowledge of large-scale software systems. This book shows how important factors, such as globalisation, modelling, coding, testing and maintenance, need to be addressed when creating a modern information system. Each chapter contains expected learning outcomes, a summary of key points and exercise questions to test knowledge and skills. Topics range from the basic concepts of design to software design quality; design strategies and processes; and software architectural styles. Theory and practice are reinforced with many worked examples and exercises, plus case studies on extraction of keyword vector from text; design space for user interface architecture; and document editor. Software Design Methodology is intended for IT industry professionals as well as software engineering and computer science undergraduates and graduates on Msc conversion courses. \* In depth coverage of large scale software systems and the handling of their design problems \* Many worked examples, exercises and case studies to reinforce theory and practice \* Gain an

## Bookmark File PDF Evaluating Software Architectures Methods And Case Studies

understanding of the general theory of design methodology

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

"This book covers both theoretical approaches and practical solutions in the processes for aligning enterprise, systems, and software architectures"--Provided by publisher. This book constitutes the refereed proceedings of the 14th International Conference on Software Architecture, ECSA 2020, held in A'quila, Italy, in September 2020. In the Research Track, 12 full papers presented together with 5 short papers were carefully reviewed and selected from 103 submissions. They are organized in topical sections as follows: microservices; uncertainty, self-adaptive, and open systems; model-based approaches; performance and security engineering; architectural smells and source code analysis; education and training; experiences and learnings from industrial case studies; and architecting contemporary distributed systems. In the Industrial Track, 11 submissions were received and 6 were accepted to form part of these proceedings. In addition the book contains 3 keynote talks. Due to the Corona pandemic ECSA 2020 was held as an virtual event.

This book constitutes the proceedings of the 8th European Conference on Software Architecture, ECSA 2014, held in Vienna, Austria, in August 2014. The 16 full papers and 18 short papers presented in this volume were carefully reviewed and selected from 91 submissions. They are organized in topical sections named: architecture decisions and knowledge; architecture patterns and anti-patterns; reference architectures and metamodels; architecture description languages; enterprise architecture, SOA and cloud computing; components and connectors; quality attributes; and architecture analysis and verification.

Evaluating Software Architectures Methods and Case Studies Addison-Wesley Professional

This book constitutes the refereed proceedings of the 14th International Conference on Software Reuse for Dynamic Systems in the Cloud and Beyond, ICSR 2015, held in Miami, FL, USA, in January 2015. The 21 revised full papers presented together with 3 revised short papers were carefully reviewed and selected from 60 submissions. The papers cover several software engineering areas where software reuse is important, such as software product lines, domain analysis, open source, components, cloud, quality.

This book constitutes the joint refereed proceedings of two colocated events: the First International Conference on the Quality of Software Architectures (QoSA 2005) and the Second International Workshop on Software Quality (SOQUA 2005) held in Erfurt, Germany, in September 2005. The 18 revised full papers presented were carefully reviewed and selected from 48 submissions. For QoSA 2005 only 12 papers - of the 31 submitted - were accepted for presentation; they are concerned with research and experiences that investigate the influence a

specific software architecture has on software quality aspects. The papers are organized in topical sections on software architecture evaluation, formal approaches to model-driven QoS-handling, modelling QoS in software architectures, software architectures applied, architectural design for QoS, and model-driven software reliability estimation. The 6 papers accepted for SOQUA 2005 - from 17 submissions - mainly focus on quality assurance and on software testing. They are organized in topical sections on test case selection, model-based testing, unit testing, and performance testing.

Presents three methods for evaluating the structure of large software systems during the design phase. The three techniques separately test for whether quality goals are met and how they interact; for modifiability and functionality; and for the feasibility and suitability of a set of services provided by a portion of the system. The authors, who are members of Carnegie Mellon's Software Engineering Institute, illustrate how to apply each step of the methods through case studies. c. Book News Inc.

This book contains a collection of thoroughly refereed papers presented at the 5th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2010, held in Athens, Greece, in July 2010. The 19 revised and extended full papers were carefully selected from 70 submissions. They cover a wide range of topics, such as quality and metrics; service and Web engineering; process engineering; patterns, reuse and open source; process improvement; aspect-oriented engineering; and requirements engineering. Models are used in all kinds of engineering disciplines to abstract from the various details of the modelled entity in order to focus on a specific aspect. Like a blueprint in civil engineering, a software architecture provides an abstraction from the full software system's complexity. It allows software designers to get an overview on the system

underdevelopment and to analyze its properties. In this sense, models are the foundation needed for software development to become a true engineering discipline.

Especially when reasoning on a software system's extra-functional properties, its software architecture carries the necessary information for early, design-time analyses. These analyses take the software architecture as input and can be used to direct the design process by allowing a systematic evaluation of different design alternatives. For example, they can be used to cancel out decisions which would lead to architecture - signs whose implementation would not comply with extra-functional requirements like performance or reliability constraints. Besides such quality attributes directly visible to the end user, internal quality attributes, e.g., maintainability, also highly depend on the system's architecture. In addition to the above-mentioned technical aspects of software architecture models, non-technical aspects, especially project management-related activities, require an explicit software architecture model. The models are used as input for cost estimations, time-, deadline-, and resource planning for the development teams. They serve the project management activities of planning, executing, and controlling,

which are necessary to deliver high-quality software systems in time and within the budget.

Applying methodologies of Software Process Improvement (SPI) is an effective way for businesses to remain competitive in the software industry. However, many organizations find implementing software process initiatives challenging. Agile Estimation Techniques and Innovative Approaches to Software Process Improvement reviews current SPI techniques and applications through discussions on current and future trends as well as the presentation of case studies on SPI implementation. Ideal for use by academics, students, and policy-makers, as well as industry professionals and managers, this publication provides a complete overview of current tools and methodologies regarding Software Process Improvement.

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.

This book constitutes the proceedings of the 7th European Conference on Software Architecture, ECSA 2013, held in Montpellier, France, in July 2013. The 25 full papers and 11 poster papers presented in this volume were carefully reviewed and selected from a total of 82 submissions. The contributions are organized in topical sections named: architectural and design patterns and models; ADLs and architectural MetaModels; architectural design decision-making; software architecture conformance and quality; and architectural repair and adaptation.

This book constitutes the refereed proceedings of the First European Conference on Software Architecture, ECSA 2007, held in Aranjuez, Spain. The 12 revised long papers presented together with four short papers cover description languages and metamodels, architecture-based code generation, run-time monitoring, requirements engineering, service-oriented architectures, aspect-oriented software architectures, ontology-based approaches, autonomic systems, middleware and web services.

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.

A new, quantitative architecture simulation approach to software design that circumvents costly testing cycles by modeling quality of service in early design states. Too often, software designers lack an understanding of the effect of design decisions on such quality attributes as performance and reliability. This necessitates costly trial-and-error testing cycles, delaying or complicating rollout. This book presents a new, quantitative architecture simulation approach to software design, which allows software engineers to model quality of service in early design stages. It presents the first simulator for software architectures, Palladio, and shows students and professionals how to model reusable, parametrized components and configured, deployed systems in order to analyze service attributes. The text details the key concepts of Palladio's domain-specific modeling language for software architecture quality and presents the corresponding development stage. It describes how quality information can be used to calibrate architecture models from which detailed simulation models are automatically derived for quality predictions. Readers will learn how to approach systematically questions about scalability, hardware resources, and efficiency. The text features a running example to illustrate tasks and methods as well as three case studies from industry. Each chapter ends with exercises, suggestions for further reading, and "takeaways" that summarize the key points of the chapter. The simulator can be downloaded from a companion website, which offers additional material. The book can be used in graduate courses on software architecture, quality engineering, or performance engineering. It will also be an essential resource for software architects and software engineers and for practitioners who want to apply Palladio in industrial settings.

Job titles like "Technical Architect" and "Chief Architect" nowadays abound in the 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 helps resolve this predicament. 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 software structure and quality attributes, through technical issues like middleware components and documentation techniques, to emerging technologies like model-driven architecture, software product lines, aspect-oriented design, service-oriented architectures, and the Semantic Web, all of which will influence future software system architectures. 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 source of knowledge. "Ian's book helps us to head in the right direction through the various techniques and approaches... An essential guide to computer science students as well as developers and IT professionals who aspire to become an IT architect". (Anna Liu, Architect Advisor, Microsoft Australia)

# Bookmark File PDF Evaluating Software Architectures Methods And Case Studies

[Copyright: c799e491da6890ac0847708d2328d41c](#)