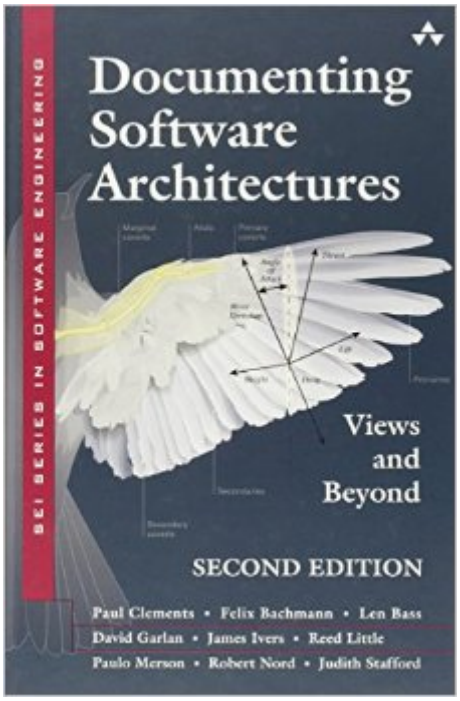
Documenting Software Architectures: Views And Beyond (2nd Edition)



Synopsis

â œThis new edition is brighter, shinier, more complete, more pragmatic, more focused than the previous one, and I wouldnâ ™t have thought it possible to improve on the original. As the field of software architecture has grown over these past decades, there is much more to be said, much more that we know, and much more that we can reflect upon of whatâ ™s worked and what hasnâ ™tâ ”and the authors here do all that, and more.â • â ”From the Foreword by Grady Booch, IBM Fellow Â Software architectureâ ”the conceptual glue that holds every phase of a project together for its many stakeholdersâ ”is widely recognized as a critical element in modern software development. Practitioners have increasingly discovered that close attention to a software systemâ ™s architecture pays valuable dividends. Without an architecture that is appropriate for the problem being solved, a project will stumble along or, most likely, fail. Even with a superb architecture, if that architecture is not well understood or well communicated the project is unlikely to succeed. Â Documenting Software Architectures, Second Edition, provides the most complete and current guidance, independent of language or notation, on how to capture an architecture in a commonly understandable form. Drawing on their extensive experience, the authors first help you decide what information to document, and then, with guidelines and examples (in various notations, including UML), show you how to express an architecture so that others can successfully build, use, and maintain a system from it. The book features rules for sound documentation, the goals and strategies of documentation, architectural views and styles, documentation for software interfaces and software behavior, and templates for capturing and organizing information to generate a coherent package. New and improved in this second edition: Â Coverage of architectural styles such as service-oriented architectures, multi-tier architectures, and data models Guidance for documentation in an Agile development environment Deeper treatment of documentation of rationale, reflecting best industrial practices Improved templates, reflecting years of use and feedback, and more documentation layout options A new, comprehensive example (available online), featuring documentation of a Web-based service-oriented system Reference guides for three important architecture documentation languages: UML, AADL, and SysML

Book Information

Hardcover: 592 pages
Publisher: Addison-Wesley Professional; 2 edition (October 15, 2010)
Language: English
ISBN-10: 0321552687

ISBN-13: 978-0321552686
Product Dimensions: 6.6 x 1.4 x 9.3 inches
Shipping Weight: 2.1 pounds (View shipping rates and policies)
Average Customer Review: 4.5 out of 5 starsÂ Â See all reviewsÂ (20 customer reviews)
Best Sellers Rank: #337,059 in Books (See Top 100 in Books) #163 inÂ Books > Reference > Writing, Research & Publishing Guides > Writing > Technical #397 inÂ Books > Textbooks > Computer Science > Software Design & Engineering #910 inÂ Books > Computers & Technology > Programming > Software Design, Testing & Engineering > Software Development

Customer Reviews

This book is the bible of documenting software architectures. It is a mandatory read for all software architects. It should also be read by the various stakeholders who have invested in a software architect so they have some idea about what their investment should be accomplishing.This version includes a lot of new content. It includes new architectural styles for SOA, database models, and multi-tier architectures.The authors have expanded the epilogue which is the comparison of Views and Beyond to other documenting approaches. It now includes comparisons to ISO/IEC 42010 - ANSI/IEEE Std 1471-2000, the RUPs 4+1, Rozanski and Woods Viewpoint Set, agile projects, and DoDAF.The book now includes 3 appendixes. One on UML, SysML, and AADL. Each is an overview of the language.They have improved templates based on experience gained since the first version of the book.The book's example is now on line. It documents a Web-based service-oriented system.This book shows you how to communicate with your stakeholders and how to address quality attribute requirements like no other book out there. I am not saying it is the best book out there, I am saying it accomplishes what it tries to teach. It creates a view of a great method of documenting architecture. It then provides a cross reference with some other great methods of documenting software architecture in the epilogue I mentioned above.It is a must read for any architect that takes there job serious. It will add a wealth of knowledge to your arsenal of tools.

Some of us have been practicing software modeling since the early days of Rational Rose (about 20 years ago!) And still we find this business of describing a system's architecture to be vexing. Just give us some rules! Ah, but it's not so simple. Our job as architects and designers is not to just fill out a template. It's to wrap our brains around requirements, ponder solutions, innovate, and sort it all out for ourselves and others. This book feels like modeling 5.0 or something like that. It's an evolutionary next step in the art and craft of communicating software architectures. Paul Clements

and team have done a good job of proposing a framework for exploring and communicating designs. It’s kind of easy to remember (three views). Yet they recognize that this problem doesn’t have a formulaic solution. I might complain that the book doesn’t actually work through a complete example--rather, it is rich with tiny examples of each view and style. The comprehensive example is online. Based on personal experience, I might add that this approach needs the support of modern presentation techniques. Audiences--even highly technical ones--may not warm up to a purely views-based presentation. All in all, I highly recommend this book to all software practitioners.

As a software tester, I rely heavily on system documentation. Unfortunately, documentation is often missing, obsolete, or never created in the first place.Also, I have a great appreciation for software architects and the work they produce. As a former developer, I used to struggle with the best ways to express system architecture diagrams. After all, there are so many methods available to document systems - UML being a major one, but there are others.When I started reading this book, I was struck by its practicality, beautiful simplicity, and integration between authors. Everything I read in this book is written in a clear and understandable way. The authors understand that different audiences will read this book, so they give graphical (of course) guidance in which chapters are most applicable to architects, stakeholders and novices.This book covers the basics, such as module views and module styles, component and connector views, allocation views and styles. Part two of the book goes beyond the basics and gets into issues regarding levels of detail, deciding among alternatives, documenting interfaces and documenting behavior. Part three is devoted to building the architecture documentation.There are appendices devoted to UML, SysML and AADL to show how architectural documentation is shown in each of these.It would be tempting to say that this book is needed for new technologies, such as SOA and the cloud, which is true, but too narrow. Actually, this book can be applied to any technology or approach - traditional, agile, iterative, or anything. That’s because the one thing people ask for and struggle with is documentation. This is especially true for architectural documentation.Read this book and apply the things in it and you will stand out on projects - in a good way. And, of course, that’s a good thing!

This is one of those books about which it is difficult to say enough good things. I found the first edition very influential on my on thinking to say nothing of enlightening. Impressively, the authors have improved this second edition, for example moving the comprehensive sample documentation online, thus making room for more of the lucid and rational explanation of documentation practices that made the first edition so useful.I usually include a discussion of who would benefit from a

reading in my reviews. Anybody who thinks they might be producers or consumers of software architecture documentation owes it to themselves to get a copy of this book now. Others might think they have no need for this book, but still should read it. This includes Business Analysts, who, this book makes clear, are stakeholders of software architectures. BAs will also their thinking sharpened on how to develop useful documentation and use graphical notation in a precise and meaningful fashion.This book is on my short list of classics of software engineering. It is a book I return to often. Reading this second edition, even after being fully familiar with the first, was still a richly satisfying experience. I cannot recommend this book enough.

Documenting Software Architectures: Views and Beyond (2nd Edition) Internet Routing Architectures (2nd Edition) Documenting the Documentary: Close Readings of Documentary Film and Video Make Your Own History: Documenting Feminist and Queer Activism in the 21st Century Kidwatching: Documenting Children's Literacy Development American Higher Education Transformed, 1940-2005: Documenting the National Discourse Assessing Liaison Librarians: Documenting Impact for Positive Change (PIL #67) (Publications in Librarianship) DSP Processor Fundamentals: Architectures and Features Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide (The Savvy Manager's Guides) UNIX Systems for Modern Architectures: Symmetric Multiprocessing and Caching for Kernel Programmers Parallel Programming with Microsoft Visual C++: Design Patterns for Decomposition and Coordination on Multicore Architectures (Patterns & Practices) VLSI Test Principles and Architectures: Design for Testability (The Morgan Kaufmann Series in Systems on Silicon) Fundamentals of Neural Networks: Architectures, Algorithms And Applications Business Process Management: Concepts, Languages, Architectures Optimizing Compilers for Modern Architectures: A Dependence-based Approach Enterprise Software Procurement: Tools and Techniques for Successful Software Procurement and Business Process Reengineering for Municipal Executives and Managers Code/Space: Software and Everyday Life (Software Studies) The Software Paradox: The Rise and Fall of the Commercial Software Market More Joel on Software: Further Thoughts on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, ... or Ill Luck, Work with Them in Some Capacity Swift: Programming, Master's Handbook: A TRUE Beginner's Guide! Problem Solving, Code, Data Science, Data Structures & Algorithms (Code like a PRO in ... mining, software, software engineering,)