A Practical Guide To Software Localization Language International World Directory V 3

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
The amount of software used in safety-critical systems is increasing at a rapid rate. At the same time, software technology is changing, projects are pressed to develop software faster and more cheaply, and the software is being used in more critical ways. Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance equips you with the information you need to effectively and efficiently develop safety-critical, life-critical, and mission-critical software for aviation. The principles also apply to software for automotive, medical, nuclear, and other safety-critical domains. An international authority on safety-critical software, the author helped write DO-178C and the U.S. Federal Aviation Administration's policy and guidance on safety-critical software. In this book, she draws on more than 20 years of experience as a certification authority, an avionics manufacturer, an aircraft integrator, and a software developer to present best practices, real-world examples, and concrete recommendations. The

book includes: An overview of how software fits into the systems and safety processes Detailed examination of DO-178C and how to effectively apply the guidance Insight into the DO-178C-related documents on tool qualification (DO-330), model-based development (DO-331), object-oriented technology (DO-332), and formal methods (DO-333) Practical tips for the successful development of safety-critical software and certification Insightful coverage of some of the more challenging topics in safety-critical software development and verification, including real-time operating systems, partitioning, configuration data, software reuse, previously developed software, reverse engineering, and outsourcing and offshoring An invaluable reference for systems and software managers, developers, and quality assurance personnel, this book provides a wealth of information to help you develop, manage, and approve safety-critical software more confidently.

Short of hauling it from their garages to the curb with their SUVs, most folks do not have a clue about getting their new-born product on the street. Experienced practitioners Richardson and Gwaltney give inside information on the practicalities of managing a development project, whether from the aforesaid garage or from the largest cube farm in th.

Heather Meeker's Open Source for Business is a practical, readable guide to help businesspeople, engineers, and lawyers understand open source software licensing. Based on the author's twenty

years as an attorney working at the crossroads of intellectual property and technology, this guide explains the legal and technical principles behind open source licensing so you can make the right decisions for your business. It offers tips on using open source, contributing to open source projects, and releasing your own open source software. You'll also get access to quick-reference tables on the major open source licenses, plus forms and checklists you can use to promote compliance. In this book, you will learn . . . * Why open source is not a "virus" * What the GPL is and how to handle it * When and how to conduct open source audits * What a user-friendly open source policy looks like * How to avoid and respond to open source enforcement claims * How to use open source to fight patent infringement claims * How to manage trademarks for open source products
Guide to Software Development
A practical guide for translators and translation students
A Practical Guide to Developing and Marketing Your Software Project
Delivering Successful Projects with TSP(SM) and Six Sigma
Performance Solutions
How to Break Software
Software for Use

This guidebook examines the fundamental issues that both licensors and licensees confront in the negotiation of a software license and, where appropriate, relevant

ancillary issues such as software development as well as maintenance and support. A companion CD-ROM is included with customizable agreements and relevant forms.

If you are responsible for designing, implementing, or managing a quality software program, this updated edition of the Practical Guide to Software Quality Management now identifies 10 major components that make up a solid program in line with ISO 9001 quality management precepts. Thoroughly revised and with new chapters on software safety and software risk management, this comprehensive primer provides you with the starting points for a standardized documentation system, and analyzes each individual program component separately, addressing in detail its specific role and overall importance to the system. Translation technology has evolved quickly with a large number of translation tools available. In this revised addition, much content has been added about translating and engineering HTML and XML documents, multilingual web sites, and HTML-based online help systems. Other major changes include the addition of chapters on internationalizatoi, software quailty assurance, descktop publishing and localization supprort. There is a focus on translators who want to learn about localization ad translation technology. A rich case-study analysis of open source software adoption by public organizations in different countries and settings. Government agencies and public organizations often consider adopting open source software (OSS) for reasons of transparency, cost, citizen

access, and greater efficiency in communication and delivering services. Adopting Open Source Software offers five richly detailed real-world case studies of OSS adoption by public organizations. The authors analyze the cases and develop an overarching, conceptual framework to clarify the various enablers and inhibitors of OSS adoption in the public sector. The book provides a useful resource for policymakers, practitioners, and academics. The five cases of OSS adoption include a hospital in Ireland; an IT consortium serving all the municipalities of the province of Bozen-Bolzano, Italy; schools and public offices in the Extremadura region of Spain; the Massachusetts state government's open standards policy in the United States; and the ICT department of the Italian Chamber of Deputies. The book provides a comparative analysis of these cases around the issues of motivation, strategies, technologies, economic and social aspects, and the implications for theory and practice.

Object-oriented Software Development
Architecture-centric Software Project Management
A Practical Guide using UML
An Introduction with LabVIEW
A Practical Guide to Enterprise Architecture
A Practical Guide to TPM 2.0
Designing Software Architectures

The non-technical guide to building a booming tech-enabled business Thinking of starting a technology-enabled business? Or maybe you just want to increase your technology mojo so you can do your job better? You do not need to learn programming to participate in the development of today's hottest technologies. But

there are a few easy-to-grasp foundation concepts that will help you engage with a technical team. Starting a Tech Business explains in practical, actionable terms how to formulate and reality test new ideas package what you learn into frameworks that are highly actionable for engineers understand key foundation concepts about modern software and systems participate in an agile/lean development team as the "voice of the customer" Even if you have a desire to learn to program (and I highly recommend doing whatever unlocks your "inner tinkerer"), these foundation concepts will help you target what exactly you want to understand about hands-on technology development. While a decade ago the barriers to creating a technology-enabled business required a pole vault, getting started today only requires a determined step in the right direction. Starting a Tech Business supplies the tools prospective entrepreneurs and business enterprises need to avoid common pitfalls and succeed in the fast-paced world of high-tech business. Successful execution requires thoughtful, evidence-based product formulation, well-articulated design, economic use of systems, adaptive management of technical resources, and empathetic deployment to customers. Starting a Tech Business offers practical checklists and frameworks that business owners, entrepreneurs, and professionals can apply to any tech-based business idea, whether you're developing software and products or beginning a technology-enabled business. You'll learn: 1. How to apply today's leading management frameworks to a tech business 2. How to package your product idea in a way that's highly actionable for your technical team 3. How to ask the right questions about technology selection and product architecture 4. Strategies to leverage what your technology ecosystem has to offer 5. How to carefully define the roles on your team, and then effectively evaluate candidates 6. The most common disconnects between engineers and business people and how to avoid them 7. How you can apply process design to your tech business without stifling creativity 8. The steps to avoid the most common pitfalls tech founders encounter Now is one of

the best times to start a technology-enabled business, and anyone can do it with the right amount and kind of preparation. Starting a Tech Business shows you how to move a product idea to market quickly and inexpensively—and to tap into the stream of wealth that a tech business can provide.

Simplifying the often confusing array of software programs for fitting linear mixed models (LMMs), Linear Mixed Models: A Practical Guide Using Statistical Software provides a basic introduction to primary concepts, notation, software implementation, model interpretation, and visualization of clustered and longitudinal data. This easy-to-navigate reference details the use of procedures for fitting LMMs in five popular statistical software packages: SAS, SPSS, Stata, R/S-plus, and HLM. The authors introduce basic theoretical concepts, present a heuristic approach to fitting LMMs based on both general and hierarchical model specifications, develop the model-building process step-by-step, and demonstrate the estimation, testing, and interpretation of fixed-effect parameters and covariance parameters associated with random effects. These concepts are illustrated through examples using real-world data sets that enable comparisons of model fitting options and results across the software procedures. The book also gives an overview of important options and features available in each procedure. Making popular software procedures for fitting LMMs easy-to-use, this valuable resource shows how to perform LMM analyses and provides a clear explanation of mixed modeling techniques and theories.

Rev. ed of: A practical guide to software licensing for licensees and licensors / H. Ward Classen. 3rd ed. c2008.

The Practical Guide to Software Licensing and Cloud Computing
Software That Sells
A Practitioner's Guide to Software Test Design
A Practical Guide to Testing
A Practical Guide
A Practical Approach

The Practical Guide to Software Licensing and Cloud Computing
A Practical Guide to Building Trust While Delivering Innovation
Mathematics of Computing -- Numerical Analysis.
This title provides systematic performance planning techniques for diverse computing environments and architectures. It seeks to smoothly integrate performance analysis into an existing software development process.
The software industry has undergone rapid development since the beginning of the twenty-first century. These changes have had a profound impact on translators who, due to the evolving nature of digital content, are under increasing pressure to adapt their ways of working. Localizing Apps looks at these challenges by focusing on the localization of software applications, or apps. In each of the five core chapters, Johann Roturier examines: The role of translation and other linguistic activities in adapting software to the needs of different cultures (localization); The procedures required to prepare source content before it gets localized (internationalization); The measures taken by software companies to guarantee the quality and success of a localized app. With practical tasks, suggestions for further reading and concise chapter summaries, Localizing Apps takes a comprehensive look at the transformation processes and tools used by the software industry today. This text is essential reading for students, researchers and translators working in the area of translation and creative digital media.
Written by a leading expert in the field, this unique volume contains current test design approaches and focuses only on software test design. Copeland illustrates each test design through detailed examples and step-by-step instructions.
A Practical Guide for Anyone Creating or Designing Applications or Software
A Practical Guide to Successful Software Projects
A Practical Guide Using Statistical Software

Using the Trusted Platform Module in the New Age of Security

Introduction to Finite and Spectral Element Methods Using MATLAB

A Practical Guide to SysML

"Clear, correct, and deep, this is a welcome addition to discussions of law and computing for anyone -- even lawyers!"-- Lawrence Lessig, Professor of Law at Stanford Law School and founder of the Stanford Center for Internet and Society If you work in information technology, intellectual property is central to your job -- but dealing with the complexities of the legal system can be mind-boggling. This book is for anyone who wants to understand how the legal system deals with intellectual property rights for code and other content. You'll get a clear look at intellectual property issues from a developer's point of view, including practical advice about situations you're likely to encounter. Written by an intellectual property attorney who is also a programmer, Intellectual Property and Open Source helps you understand patents, copyrights, trademarks, trade secrets, and licenses, with special focus on the issues surrounding open source development and the GPL. This book answers questions such as: How do open source and intellectual property work together? What are the most important intellectual property-related issues when starting a business or open source project? How should you handle copyright, licensing and other issues when accepting a patch from another developer? How can you pursue your own ideas while working for someone else? What parts of a patent should be reviewed to see if it applies to your work? When is your idea a trade secret? How can you reverse engineer a product without getting into trouble? What should you think about when choosing an open source license for your project? Most legal sources are too scattered, too arcane, and too hard to read. Intellectual

Property and Open Source is a friendly, easy-to-follow overview of the law that programmers, system administrators, graphic designers, and many others will find essential.
This practical guide brings DevOps principles to Salesforce development. It fits together two major movements within the IT world: the movement to Software/Platform as a Service (SaaS/PaaS), and the DevOps movement. While SaaS and PaaS allow companies to invest in their core competencies rather than maintain their own infrastructure, the goal of DevOps is to optimize the process of delivering software innovation and value. The release of Salesforce DX in late 2017 unlocks the possibility of a true DevOps workflow on Salesforce. But DevOps is new to the Salesforce world and there is not a widespread understanding of its goals and methods, and so adoption of Salesforce DX is still in the early stages. Mastering Salesforce DevOps explains how to build a powerful and comprehensive DevOps workflow for Salesforce—allowing you to finally deploy the world's most innovative platform using the world's most effective and efficient techniques. It addresses the need for a comprehensive guide to DevOps for Salesforce, allowing teams to bring proven practices from the IT world to resolve the hardest problems facing Salesforce developers today. What You Will Learn Improve company performance and software delivery performance using Salesforce DX Translate DevOps concepts into the unique language and practices of Salesforce Understand why and how you can implement Salesforce DX to achieve greater productivity and innovation Enable continuous delivery on Salesforce Build packages and architect code so it can be deployed easilyAllow admins to participate in what has traditionally been a developer workflow Know the techniques for reducing the stress and risk of deploymentApply the full range of automated tests that can be used on Salesforce Who This Book Is for Salesforce developers, release managers, and those managing Salesforce

development teams who need a guide to DevOps, and DevOps specialists who need to apply familiar concepts to Salesforce Ship It! is a collection of tips that show the tools andtechniques a successful project team has to use, and how to use themwell. You'll get quick, easy-to-follow advice on modernpractices: which to use, and when they should be applied. This bookavoids current fashion trends and marketing hype; instead, readersfind page after page of solid advice, all tried and tested in thereal world. Aimed at beginning to intermediate programmers, Ship It! will show you: Which tools help, and which don't How to keep a project moving Approaches to scheduling that work How to build developers as well as product What's normal on a project, and what's not How to manage managers, end-users and sponsors Danger signs and how to fix them Few of the ideas presented here are controversial or extreme; most experiencedprogrammers will agree that this stuff works. Yet 50 to 70 percent of allproject teams in the U.S. aren't able to use even these simple, well-acceptedpractices effectively. This book will help you get started. Ship It! begins by introducing the common technicalinfrastructure that every project needs to get the job done. Readerscan choose from a variety of recommended technologies according totheir skills and budgets. The next sections outline the necessarysteps to get software out the door reliably, using well-accepted,easy-to-adopt, best-of-breed practices that really work. Finally, and most importantly, Ship It! presents commonproblems that teams face, then offers real-world advice on how tosolve them.

Software development consultant Wiegers describes various formal and informal methods for conducting a peer review program, such as pair programming, team reviews, the "walkthrough," and the ad hoc review. The main part of the text is devoted to the various stages of the technique of inspection. Coverage extends to the social issues involved in

critiquing the work of others and overcoming resistance to reviews. c. Book News Inc.

A Practical Guide to the Models and Methods of Usage-Centered Design

Starting a Tech Business

Large-Scale Software Architecture

A Practical Guide to Software Licensing for Licensees and Licensors

Developing Safety-Critical Software

Peer Reviews in Software

A Practical Guide to Localization

This book is an essential resource for lawyers who work with clients and business transactions in the software licensing arena. It is written in practical, easy-to-understand language, and written from the perspective of both the licensor and the licensee. A CD-ROM is included and filled with numerous customizable and easy-to-use forms and agreements.

In the quest for quality, software developers have long focused on improving the internal architecture of their products. Larry L. Constantine--who originally created structured design to effect such improvement--now joins with well-known consultant Lucy A. D. Lockwood to turn the focus of software development to the external architecture. In this book, they present the models and methods of a revolutionary approach to software that will help programmers deliver more usable software--software that will enable users to accomplish their tasks with greater ease and efficiency. Recognizing usability as the key to successful software, Constantine and Lockwood provide concrete tools and techniques that programmers can employ to meet that end. Much more than just another set of rules for good user-interface design, this book guides readers through a systematic software development

process. This process, called usage-centered design, weaves together two major threads in software development methods: use cases (also used with UML) and essential modeling. With numerous examples and case studies of both conventional and specialized software applications, the authors illustrate what has been shown in practice to work and what has proved to be of greatest practical value. Highlights Presents a streamlined process for developing highly usable software Describes practical methods and models successfully implemented in industry Complements modern development practices, including the Unified Process and other object-oriented software engineering approaches

This updated 7th edition examines fundamental issues that both vendors and customers may confront in the negotiation of a software license and a cloud computing agreement, including related issues such as software development, professional services, maintenance and support. Included are four new chapters and 41 new forms, checklists, and policies. The cloud computing chapter has been expanded.

Using Continuous Delivery, you can bring software into production more rapidly, with greater reliability. A Practical Guide to Continuous Delivery is a 100% practical guide to building Continuous Delivery pipelines that automate rollouts, improve reproducibility, and dramatically reduce risk. Eberhard Wolff introduces a proven Continuous Delivery technology stack, including Docker, Chef, Vagrant, Jenkins, Graphite, the ELK stack, JBehave, and Gatling. He guides you through applying these technologies throughout build, continuous integration, load testing, acceptance testing, and monitoring. Wolff's start-to-finish example projects offer the basis for your own experimentation, pilot programs, and full-fledged deployments. A Practical Guide

to Continuous Delivery is for everyone who wants to introduce Continuous Delivery, with or without DevOps. For managers, it introduces core processes, requirements, benefits, and technical consequences. Developers, administrators, and architects will gain essential skills for implementing and managing pipelines, and for integrating Continuous Delivery smoothly into software architectures and IT organizations. Understand the problems that Continuous Delivery solves, and how it solves them Establish an infrastructure for maximum software automation Leverage virtualization and Platform as a Service (PAAS) cloud solutions Implement build automation and continuous integration with Gradle, Maven, and Jenkins Perform static code reviews with SonarQube and repositories to store build artifacts Establish automated GUI and textual acceptance testing with behavior-driven design Ensure appropriate performance via capacity testing Check new features and problems with exploratory testing Minimize risk throughout automated production software rollouts Gather and analyze metrics and logs with Elasticsearch, Logstash, Kibana (ELK), and Graphite Manage the introduction of Continuous Delivery into your enterprise Architect software to facilitate Continuous Delivery of new capabilities

A Practical Guide to Brain–Computer Interfacing with BCI2000

Practical Guide to Machine Vision Software

Designing and Managing the Life Cycle

Ship It!

Mastering Salesforce DevOps

General-Purpose Software for Brain-Computer Interface Research, Data Acquisition, Stimulus Presentation, and Brain Monitoring

Java Software Development with Event B

Incorporating new topics and original material, Introduction to Finite and Spectral Element Methods Using MATLAB, Second Edition enables readers to quickly understand the theoretical foundation and practical implementation of the finite element method and its companion spectral element method. Readers gain hands-on computational experience by using
The cost of fixing software design flaws after the completion of a software product is so high that it is vital to come up with ways to detect software design flaws in the early stages of software development, for instance, during the software requirements, the analysis activity, or during software design, before coding starts. It is not uncommon that software requirements are ambiguous or contradict each other. Ambiguity is exacerbated by the fact that software requirements are typically written in a natural language, which is not tied to any formal semantics. A palliative to the ambiguity of software requirements is to restrict their syntax to boilerplates, textual templates with placeholders. However, as informal requirements do not enjoy any particular semantics, no essential properties about them (or about the system they attempt to describe) can be proven easily. Formal methods are an alternative to address this problem. They offer a range of mathematical techniques and mathematical tools to validate software requirements in the early

3333333

333333333333

333

stages of software development. This book is a living proof of the use of formal methods to develop software. The particular formalisms that we use are EVENT B and refinement calculus. In short: (i) software requirements as written as User Stories; (ii) they are ported to formal specifications; (iii) they are refined as desired; (iv) they are implemented in the form of a prototype; and finally (v) they are tested for inconsistencies. If some unit-test fails, then informal as well as formal specifications of the software system are revisited and evolved. This book presents a case study of software development of a chat system with EVENT B and a case study of formal proof of properties of a social network.

Delivering successful projects means the ability to produce high quality software within budget and on time—consistently, but when one mentions quality to software engineers or project managers, they talk about how impossible it is to eliminate defects from software. This assumption is passed on and on until it becomes accepted wisdom, with the power of a self-fulfilling prophecy. And when a project fails to arrive on time or up to standards, team members will turn on each other. The project got delayed because the engineers did a poor job in development or too much was promised upfront for this short of a timeline. In Delivering Successful Projects with TSPSM and

Six Sigma: A Practical Guide to Implementing Team Software ProcessSM, you will learn how to effectively manage the development of a software project and deliver it in line with customer expectations. This refreshing volume — Offers real-world case studies about the author's experience at Microsoft successfully implementing TSP to achieve higher quality software Empowers software developers to take responsibility for project management Explains how Six Sigma and TSP combined can dramatically reduce software defects By applying these principles put forth by one of the most respected names in software development, your software team will learn how to function as a team and turn out products where zero defects and on-time delivery are the norm.

A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security is a straight-forward primer for developers. It shows security and TPM concepts, demonstrating their use in real applications that the reader can try out. Simply put, this book is designed to empower and excite the programming community to go out and do cool things with the TPM. The approach is to ramp the reader up quickly and keep their interest.A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security explains security concepts, describes the TPM 2.0 architecture, and provides code and pseudo-code examples in

parallel, from very simple concepts and code to highly complex concepts and pseudo-code. The book includes instructions for the available execution environments and real code examples to get readers up and talking to the TPM quickly. The authors then help the users expand on that with pseudo-code descriptions of useful applications using the TPM.

Practical Guide to Software Quality Management Open Source for Business

A Practical Guide to Creating Responsive, Scalable Software

The Systems Modeling Language

A New Theory and Practical Guide for Modern Architects

A Practical Guide to Testing Object-oriented Software

A Practical Guide to Implementing Team Software Process(SM)

What Is BCI2000? BCI2000 is a general-purpose software platform for brain–computer interface (BCI) research. It can also be used for a wide variety of data acquisition, stimulus p-sentation, and brain monitoring applications. BCI2000 has been in development since 2000 in a project led by the Brain–Computer Interface R&D Program at the Wadsworth Center of the New York State Department of Health in Albany, New York, USA, with substantial contributions by the Institute of Medical Psychology and Behavioral Neurobiology at the University of Tübingen, Germany. In addition, many laboratories around the world, most notably the BrainLab at Georgia State University in Atlanta, Georgia, and Fondazione Santa Lucia in Rome, Italy, have also played an important

role in the project's development. Mission The mission of the BCI2000 project is to facilitate research and the development of applications in all areas that depend on real-time acquisition, processing, and feedback of biosignals. Vision Our vision is that BCI2000 will become a widely used software tool for diverse areas of research and development. bull; Written by expert practitioners who have hands-on experience solving real-world problems for large corporations bull; Helps enterprise architects make sense of data, systems, software, services, product lines, methodologies, and much more bull; Provides explanation of theory and implementation with real-world business examples to support key points

This book presents a guide to navigating the complicated issues of quality and process improvement in enterprise software implementation, and the effect these have on the software development life cycle (SDLC). Offering an integrated approach that includes important management and decision practices, the text explains how to create successful automated solutions that fit user and customer needs, by mixing different SDLC methodologies. With an emphasis on the realities of practice, the book offers essential advice on defining business requirements, and managing change. This revised and expanded second edition includes new content on such areas as cybersecurity, big data, and digital transformation. Features: presents examples, case studies, and chapter-ending problems and exercises; concentrates on the skills needed to distinguish successful software implementations; considers the political and cultural realities in organizations; suggests many alternatives for how to manage and model a system.

For both students and engineers in R&D, this book explains machine vision in a concise, hands-on way, using the Vision Development Module of the LabView software by National

Instruments. Following a short introduction to the basics of machine vision and the technical procedures of image acquisition, the book goes on to guide readers in the use of the various software functions of LabView's machine vision module. It covers typical machine vision tasks, including particle analysis, edge detection, pattern and shape matching, dimension measurements as well as optical character recognition, enabling readers to quickly and efficiently use these functions for their own machine vision applications. A discussion of the concepts involved in programming the Vision Development Module rounds off the book, while example problems and exercises are included for training purposes as well as to further explain the concept of machine vision. With its step-by-step guide and clear structure, this is an essential reference for beginners and experienced researchers alike.

A Practical Guide to Protecting Code
A Practical Guide to Open Source Software Licensing
Intellectual Property and Open Source
Localizing Apps
Analyses and Model Forms
Ship it!
Linear Mixed Models

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

architecture; how to clarify architecture so projects can be more effectively planned and managed; and then how to use architecture to organize, implement, and measure the project iteratively as work progresses.

Updated and expanded, this sixth edition is an essential resource for lawyers who work with clients and conduct business transactions in this area. The guide includes new chapters on types of licenses, software development agreements, and software development kits and application program interfaces. New and updated model forms are available on an accompanying website that you can access allowing you to customize for your own practice needs. David A. Sykes is a member of Wofford College's faculty. With this practical book, architects, CTOs, and CIOs will learn a set of patterns for the practice of architecture, including analysis, documentation, and communication. Author Eben Hewitt shows you how to create holistic and thoughtful technology plans, communicate them clearly, lead people toward the vision, and become a great architect or Chief Architect. This book covers each key aspect of architecture comprehensively, including how to incorporate business architecture, information architecture, data architecture, application (software) architecture together to have the best chance for the system's success. Get a practical set of proven architecture practices focused on shipping great products using architecture Learn how architecture works effectively with development teams, management, and product management teams through the value chain Find updated special coverage on machine learning architecture Get usable templates to start incorporating into your teams immediately Incorporate business architecture, information architecture, data architecture, and application (software) architecture together
Semantic Software Design

A Practical Guide to Continuous Delivery
Adopting Open Source Software
A Practical Guide for Aviation Software and DO-178C Compliance
Templates for the Solution of Algebraic Eigenvalue Problems

This book provides an iteractive development process and an object-oriented (O-O) development methodology including techniques on scheduling, milestone completion and other requirements for tools to support O-O development. It provides a process and methodology that can be followed to accomplish an analysis, design, implementation, and test of model objects for an application being developed.

CD-ROM contains: Canned HEAT v.2.0 -- Holodeck Lite v. 1.0.

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

description of UML usage for large-scale architecture * Discusses software architecture and design principles * Technology and vendor independent A Practical Guide to SysML: The Systems Modeling Language is a comprehensive guide to SysML for systems and software engineers. It provides an advanced and practical resource for modeling systems with SysML. The source describes the modeling language and offers information about employing SysML in transitioning an organization or project to model-based systems engineering. The book also presents various examples to help readers understand the OMG Systems Modeling Professional (OCSMP) Certification Program. The text is organized into four parts. The first part provides an overview of systems engineering. It explains the model-based approach by comparing it with the document-based approach and providing the modeling principles. The overview of SYsML is also discussed. The second part of the book covers a comprehensive description of the language. It discusses the main concepts of model organization,

parametrics, blocks, use cases, interactions, requirements, allocations, and profiles. The third part presents examples that illustrate how SysML supports different model-based procedures. The last part discusses how to transition and deploy SysML into an organization or project. It explains the integration of SysML into a systems development environment. Furthermore, it describes the category of data that are exchanged between a SysML tool and other types of tools, and the types of exchange mechanisms that can be used. It also covers the criteria that must be considered when selecting a SysML. Software and systems engineers, programmers, IT practitioners, experts, and non-experts will find this book useful. *The authoritative guide for understanding and applying SysML *Authored by the foremost experts on the language *Language description, examples, and quick reference guide included