Softwarearchitekturen Dokumentieren Und Kommunizi

The primary audience for this book are advanced undergraduate students and graduate students. Computer architecture, as it happened in other fields such as electronics, evolved from the small to the large, that is, it left the realm of low-level hardware constructs, and gained new dimensions, as distributed systems became the keyword for system implementation. As such, the system architect, today, assembles pieces of hardware that are at least as large as a computer or a network router or a LAN hub, and assigns pieces of software that are self-contained, such as client or server programs, Java applets or pro tocol modules, to those hardware components. The freedom she/he now has, is tremendously challenging. The problems alas, have increased too. What was before mastered and tested carefully before a fully-fledged mainframe or a closely-coupled computer cluster came out on the market, is today left to the responsibility of computer engineers and scientists invested in the role of system architects, who fulfil this role on behalf of software vendors and in tegrators, add-value system developers, R&D institutes, and final users. As system complexity, size and diversity grow, so increases the probability of in consistency, unreliability, non responsiveness and insecurity, not to mention the management overhead. What System Architects Need to Know The insight such an architect must have includes but goes well beyond, the functional properties of distributed systems.
Oft werden Absolventen von Informatikstudieng ä ngen in Unternehmen eingestellt, die Software f ü r eingebettete Systeme entwickeln - obwohl sie in ihrer Ausbildung nur wenig Kontakt mit technischen Systemen hatten. Daher ist der Einstieg oft zeit- und kostenintensiv. Dieses Buch erleichtert das Einarbeiten, indem es systematisch und anschaulich die grundlegenden Begriffe, Konzepte und Problemstellungen vermittelt. Entlang eines Softwareentwicklungszyklus wird beschrieben, wie in jedem Prozessschritt die speziellen Anforderungen eines eingebetteten bzw. Echtzeitsystems ber ü cksichtigt werden.
A fast-paced, thorough introduction to modern C++ written for experienced programmers. After reading C++ Crash Course, you'll be proficient in the core language concepts, the C++ Standard Library, and the Boost Libraries. C++ is one of the most widely used languages for real-world software. In the hands of a knowledgeable programmer, C++ can produce small, efficient, and readable code that any programmer would be proud of. Designed for intermediate to advanced programmers, C++ Crash Course cuts through the weeds to get you straight to the core of C++17, the most modern revision of the ISO standard. Part 1 covers the core of the C++ language, where you'll learn about everything from types and functions, to the object life cycle and expressions. Part 2 introduces you to the C++ Standard Library and Boost Libraries, where you'll learn about all of the high-quality, fully-featured facilities available to you. You'll cover special utility classes, data structures, and algorithms, and learn how to manipulate file systems and build high-performance programs that communicate over networks. You'll learn all the major features of modern C++, including: · Fundamental types, reference types, and user-defined types · The object lifecycle including storage duration, memory management, exceptions, call stacks, and the RAII paradigm · Compile-time polymorphism with templates and run-time polymorphism with virtual classes · Advanced expressions, statements, and functions · Smart pointers, data structures, dates and times, numerics, and probability/statistics facilities · Containers, iterators, strings, and algorithms · Streams and files, concurrency, networking, and application development With well over 500 code samples and nearly 100 exercises, C++ Crash Course is sure to help you build a strong C++ foundation.
This book provides a comprehensive introduction into the SPES XT modeling framework. Moreover, it shows the applicability of the framework for the development of embedded systems in different industry domains and reports on the lessons learned. It also describes how the SPES XT modeling framework can be tailored to meet domain and project-specific needs. The book is structured into four parts: Part I “Starting Situation” discusses the status quo of the development of embedded systems with specific focus on model-based engineering and summarizes key challenges emerging from industrial practice. Part II “Modeling Theory” introduces the SPES XT modeling framework and explains the core underlying principles. Part III “Application of the SPES XT Framework” describes the application of the SPES XT modeling framework and how it addresses major industrial challenges. Part IV “Evaluation and Technology Transfer” assess the impact of the SPES XT modeling framework and includes various exemplary applications from automation, automotive, and avionics. Overall, the SPES XT modeling framework offers a seamless model-based engineering approach. It addresses core challenges faced during the engineering of embedded systems. Among others, it offers aligned and integrated techniques for the early validation of engineering artefacts (including requirements and functional and technical designs), the management of product variants and their variability, modular safety assurance and deployment of embedded software.
Groovy in Action
Ein praktischer Leitfaden
Grundlagen der Programmierung eingebetteter Systeme - Eine Einf ü hrung f ü r Anwendungsentwickler
A Study Guide for the Certified Professional for Software Architecture ® - Foundation Level - iSAQB compliant
A Risk-Driven Approach
Eine Konstruktionslehre f ü r administrative Softwaresysteme
Volume 1: Constraint Validation, Enumerations, Special Datatypes

Dokumentation wird oft als lästige Pflicht angesehen und in vielen Softwareprojekten stark vernachlässigt. Dabei ermöglicht sie die Kommunikation von Konzepten im Team und dem Auftraggeber gegenüber oft überhaupt erst. Das gilt besonders für die Softwarearchitektur. Dieses Buch zeigt, was von einer Architektur in jedem Falle festgehalten werden sollte, und warum. Hauptziel ist eine nachvollziehbare Softwarearchitektur. Daher gliedert sich das Buch in einen Anforderungsteil ("Welche Rahmenbedingungen, Qualitätsmerkmale und Risiken beeinflussen die Architektur?") und einen Lösungsteil ("Wie haben wir uns an zentralen Stellen entschieden, um die Anforderungen zu erfüllen?"). Neben dem Festhalten von Architekturentscheidungen geht es vor allem um Sichten auf Softwarearchitekturen, also um graphische Techniken, z.B. UML-Diagramme. Im Idealfall entstehen die vorgeschlagenen Arbeitsergebnisse schon während des Architekturentwurfs. So lassen sich sowohl Entscheidungen als auch die Gründe dafür leicht festhalten. Diese Dokumentation von etwas Entstehenden ist weitaus einfacher als die Dokumentation von Bestehendem. In der realen Welt kommt der zweite Fall leider extrem häufig vor. Das Buch zeigt auf, wie die Arbeitsergebnisse und Methoden so auf bestehende Softwaresysteme angewendet werden können, dass sie schnell positive Wirkung zeigen.
Renowned Excel experts Bill Jelen (MrExcel) and Tracy Syrstad explain how to build more powerful, reliable, and efficient Excel spreadsheets. Use this guide to automate virtually any routine Excel task: save yourself hours, days, maybe even weeks. Make Excel do things you thought were impossible, discover macro techniques you won’t find anywhere else, and create automated reports that are amazingly powerful. Bill Jelen and Tracy Syrstad help you instantly visualize information to make it actionable; capture data from anywhere, and use it anywhere; and automate the best new features in Excel 2019 and Excel in Office 365. You’ll find simple, step-by-step instructions, real-world case studies, and 50 workbooks packed with examples and complete, easy-to-adapt solutions. By reading this book, you will: Quickly master Excel macro development Work more efficiently with ranges, cells, and formulas Generate automated reports and quickly adapt them for new requirements Learn to automate pivot tables to summarize, analyze, explore, and present data Use custom dialog boxes to collect data from others using Excel Improve the reliability and resiliency of your macros Integrate data from the internet, Access databases, and other sources Automatically generate charts, visualizations, sparklines, and Word documents Create powerful solutions with classes, collections, and custom functions Solve sophisticated business analysis problems more rapidly About This Book For everyone who wants to get more done with Microsoft Excel in less time For business and financial professionals, entrepreneurs, students, and others who need to efficiently manage and analyze data
Als erste Monographie im deutschsprachigen Raum vermittelt das vorliegende Buch eine Konstruktionslehre des Software Engineering über den gesamten Lebenszyklus eines Softwareproduktes. Während Software-Technologie üblicherweise in Hochschulen oder Softwarehäusern entsteht, wird hier eine Technologie dargestellt, die bei Anwendern entstanden ist. Nur Anwender sind in großem Stil mit dem gesamten Lebenszyklus von Software konfrontiert, da sie über 50% ihres DV-Personals für die Wartung der investierten Software einsetzen müssen. Es wird ein neues, objekt-orientiertes Vorgehensmodell für die Entwicklung kommerzieller Dialogsoftware vorgestellt. Die dabei besonders wichtige Funktion und die Erscheinungsformen des Prototyping im Software-Entwicklungsprozeß werden konstruktiv geklärt, wobei der Aspekt der Kommunikation der am Entwicklungsprozeß Beteiligten besonders herausgearbeitet wird. Das Buch vermittelt eine durchgehende, produktneutrale Methodik, hinter der 10 Jahre Industrieerfahrung, aber keine Verkaufsinteressen für bestimmte Hardware- oder Softwarewerkzeuge stehen. Trotzdem wird die Benutzung von Softwarewerkzeugen fundiert behandelt, da eine prozeßorientierte Software-Entwicklung nur mit Werkzeugen möglich ist. Neben den frühen Phasen, in denen Methoden zur Datenmodellierung und Prototyping als kommunikationsunterstützende Methode wesentlich sind, wird die "Phase" Wartung vertieft behandelt, in der die wichtigsten Entscheidungen bei der Evolution von Software fallen; hierzu wird auch ein praktisch eingesetztes Werkzeug skizziert. In einer Bibliographie sind sowohl grundlegende Quellen als auch aktuelle weiterführende Literatur zusammengestellt.
Erfahren Sie, wie die Dokumentation der Architektur von der lästigen Pflicht zu einem integralen Kommunikations- und Arbeitsmittel wird. - Lernen Sie architekturrelevante Einflussfaktoren und zentrale Entscheidungen festzuhalten. - Erleben Sie am Beispiel einer Schach-Engine, wie eine nachvollziehbare Architektur entsteht. - Auf www.swadok.de: Vorlagen und weitere Informationen zum Thema und zu den Fallbeispielen Dokumentation wird oft als lästige Pflicht angesehen und in vielen Softwareprojekten stark vernachlässigt. Die Architektur wird manchmal überhaupt nicht beschrieben. Damit das in Ihren Projekten nicht passiert, schlägt dieses Buch praxiserprobte und schlanke Bestandteile für eine wirkungsvolle Architekturdokumentation vor. An einem durchgängigen Beispiel erfahren Sie, wie Sie architekturrelevante Einflussfaktoren erfassen und Ihre Softwarelösung angemessen und ohne Ballast festhalten. Sie lernen nicht nur die Vorgehensweise für das Dokumentieren während des Entwickelns kennen, sondern auch wie Sie bestehende Systeme im Nachhinein beschreiben. Neben der Methodik diskutiert das Buch auch typische Werkzeuge, mit denen Sie Architekturdokumentation erfassen, verwalten und verbreiten können, wie Wikis, UML-Werkzeuge u.a. Checklisten und Übungsaufgaben geben Ihnen die nötige Sicherheit, um Architekturdokumentation zu einem integralen Bestandteil auch in Ihrem Softwarevorhaben zu machen. "Ich wünsche Ihnen Freude mit diesem Buch. Als Reviewer durfte ich ja schon vor längerer Zeit frühe Versionen testlesen. Mehr als einmal haben mir Stefans Ratschläge in konkreten Projektsituationen seitdem geholfen." Gernot Starke.
Microsoft Excel 2019 VBA and Macros
SOFTWAREARCHITEKTUREN DOK. 2.A.
APM - Agiles Projektmanagement
Software Engineering und Prototyping
Basiswissen für Softwarearchitekten
Qualität von Softwaresystemen
Web Applications with Javascript or Java

Dieses Lehrbuch des international bekannten Autors und Software-Entwicklers Craig Larman ist ein Standardwerk zur objektorientierten Analyse und Design unter Verwendung von UML 2.0 und Patterns. Das Buch zeichnet sich insbesondere durch die Fahigkeit des Autors aus, komplexe Sachverhalte anschaulich und praxisnah darzustellen. Es vermittelt grundlegende OOA/D-Fertigkeiten und bietet umfassende Erlauterungen zur iterativen Entwicklung und zum Unified Process (UP). Anschliessend werden zwei Fallstudien vorgestellt, anhand derer die einzelnen Analyse- und Designprozesse des UP in Form einer Inception-, Elaboration- und Construction-Phase durchgespielt werden
This is the digital version of the printed book (Copyright © 2004). Who Says Large Teams Can’t Handle Agile Software Development? Agile or "lightweight" processes have revolutionized the software development industry. They're faster and more efficient than traditional software development processes. They enable developers to embrace requirement changes during the project deliver working software in frequent iterations focus on the human factor in software development Unfortunately, most agile processes are designed for small or mid-sized software development projects—bad news for large teams that have to deal with rapid changes to requirements. That means all large teams! With Agile Software Development in the Large, Jutta Eckstein—a leading speaker and consultant in the agile community—shows how to scale agile processes to teams of up to 200. The same techniques are also relevant to teams of as few as 10 developers, especially within large organizations. Topics include the agile value system as used in large teams the impact of a switch to agile processes the agile coordination of several sub-teams the way project size and team size influence the underlying architecture Stop getting frustrated with inflexible processes that cripple your large projects! Use this book to harness the efficiency and adaptability of agile software development. Stop getting frustrated with inflexible processes that cripple your large projects! Use this book to harness the efficiency and adaptability of agile software development.
Software architecture is an important factor for the success of any software project. In the context of systematic design and construction, solid software architecture ensures the fulfilment of quality requirements such as expandability, flexibility, performance, and time-to-market. Software architects reconcile customer requirements with the available technical options and the prevailing conditions and constraints. They ensure the creation of appropriate structures and smooth interaction of all system components. As team players, they work closely with software developers and other parties involved in the project. This book gives you all the basic know-how you need to begin designing scalable system software architectures. It goes into detail on all the most important terms and concepts and how they relate to other IT practices. Following on from the basics, it describes the techniques and methods required for the planning, documentation, and quality management of software architectures. It details the role, the tasks, and the work environment of a software architect, as well as looking at how the job itself is embedded in company and project structures. The book is designed for self-study and covers the curriculum for the Certified Professional for Software Architecture – Foundation Level (CPSA-F) exam as defined by the International Software Architecture Qualification Board (iSAQB).
Generative design, once known only to insiders as a revolutionary method of creating artwork, models, and animations with programmed algorithms, has in recent years become a popular tool for designers. By using simple languages such as JavaScript in p5.js, artists and makers can create everything from interactive typography and textiles to 3D-printed furniture to complex and elegant infographics. This updated volume gives a jump-start on coding strategies, with step-by-step tutorials for creating visual experiments that explore the possibilities of color, form, typography, and images. Generative Design includes a gallery of all-new artwork from a range of international designers—fine art projects as well as commercial ones for Nike, Monotype, Dolby Laboratories, the musician Bjork, and others.
Adrenaline Junkies and Template Zombies
SOA Source Book
Python 3 Object-oriented Programming
Microservices
UML 2 und Patterns angewendet - objektorientierte Softwareentwicklung
Pragmatic Evaluation of Software Architectures
Generative Design

Die Herausforderungen, denen sich Projektleiter, Führungskräfte und andere Verantwortliche in Softwareprojekten tagtäglich stellen, sind vielfältig. Sozialkompetenz und Soft Skills stellen hier maßgebliche Erfolgsfaktoren dar. In diesem Buch werden Techniken zur Führung und Weiterentwicklung von Mitarbeitern sowie zum Aufbau von Hochleistungsteams aufgezeigt und anhand konkreter Beispiele aus der IT erläutert. Die Autoren geben Antworten auf die Fragen, was Softwareentwickler motiviert, was moderne Führung gerade in den immer agiler werdenden Projekten bedeutet und wie das komplexe Miteinander überhaupt funktionieren kann. Dazu werden die Mechanismen iterativen Vorgehens und des Lernens über Retrospektiven erläutert. Auch auf Techniken zur effektiven und effizienten Gestaltung von Besprechungen wird eingegangen. Das Buch gliedert sich in fünf Teile: • Grundlagen: Soft Skills, Kommunikation und Selbstorganisation • Organisatorische Grundlagen: Besprechungen und Zeitmanagement • Entwickler führen: Agile Teams leiten, Motivation erhalten und Entscheidungen treffen • Mitarbeiter weiterentwickeln: Die Führungskraft als Coach und Mentor • Hochleistungsteams aufbauen und in die Performance führen Im Anhang befinden sich theoretische Grundlagen sowie zwei Übungen zur Selbsterfahrung. Die 3. Auflage wurde in vielen einzelnen Aspekten aktualisiert.
Cutting-edge typography for digital media and examples of how it is applied. It includes QR codes with links to the designers' videos and webpages, with examples of the fonts they use.
Offers tips, techniques, and tools to help readers take advantage of Windows XP, covering such topics as the control panel, file downloads, firewalls, removing XP components, and cookies.
Summary Groovy in Action, Second Edition is a thoroughly revised, comprehensive guide to Groovy programming. It introduces Java developers to the dynamic features that Groovy provides, and shows how to apply Groovy to a range of tasks including building new apps, integration with existing code, and DSL development. Covers Groovy 2.4. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology In the last ten years, Groovy has become an integral part of a Java developer's toolbox. Its comfortable, common-sense design, seamless integration with Java, and rich ecosystem that includes the Grails web framework, the Gradle build system, and Spock testing platform have created a large Groovy community About the Book Groovy in Action, Second Edition is the undisputed definitive reference on the Groovy language. Written by core members of the Groovy language team, this book presents Groovy like no other can—from the inside out. With relevant examples, careful explanations of Groovy's key concepts and features, and insightful coverage of how to use Groovy in-production tasks, including building new applications, integration with existing code, and DSL development, this is the only book you'll need. Updated for Groovy 2.4. Some experience with Java or another programming language is helpful. No Groovy experience is assumed. What's Inside Comprehensive coverage of Groovy 2.4 including language features, libraries, and AST transformations Dynamic, static, and extensible typing Concurrency: actors, data parallelism, and dataflow Applying Groovy: Java integration, XML, SQL, testing, and domain-specific language support Hundreds of reusable examples About the Authors Authors Dierk König, Paul King, Guillaume Laforge, Hamlet D'Arcy, Cédric Champeau, Erik Pragt, and Jon Skeet are intimately involved in the creation and ongoing development of the Groovy language and its ecosystem. Table of Contents PART 1 THE GROOVY LANGUAGE Your way to Groovy Overture: Groovy basics Simple Groovy datatypes Collective Groovy datatypes Working with closures Groovy control structures Object orientation, Groovy style Dynamic programming with Groovy Compile-time metaprogramming and AST transformations Groovy as a static language PART 2 AROUND THE GROOVY LIBRARY Working with builders Working with the GDK Database programming with Groovy Working with XML and JSON Interacting with Web Services Integrating Groovy PART 3 APPLIED GROOVY Unit testing with Groovy Concurrent Groovy with GPars Domain-specific languages The Groovy ecosystem
Entwurfsmuster für effektive Software-Entwicklung
Softwaretechnik
Methodisches Programmieren im Großen
Ein pattern-basiertes Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen
Industrie Management 6/2013
Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten
Grundlagen flexibler Softwarearchitekturen

Dieses Buch vermittelt das grundlegende Wissen, um Softwarearchitekturen zu entwerfen und richtig einzusetzen. Es beantwortet u.a. die Fragen: Was ist Softwarearchitektur? Was ist eine gute Softwarearchitektur? Wie wird sie entwickelt? Ausführlich werden Aufgaben und Werkzeuge des Softwarearchitekten, wie Dokumentation mit UML 2, Architekturstile und -muster, behandelt. Ein weiterer Schwerpunkt ist Software im industriellen Maßstab: Produktlinienansätze, MDA und - neu in der 3. Auflage - domänenspezifische Sprachen. Das Buch richtet sich an Softwareentwickler und Projektleiter.
Eine Microservices-Architektur unterteilt Software-Systeme in eine Vielzahl kleiner Dienste, die unabhängig voneinander in Produktion gebracht werden können. Jedes Team arbeitet dabei an seinen Microservices und ist weitgehend entkoppelt von anderen Teams, das erlaubt eine einfache Skalierung agiler Prozesse. Die Aufteilung in Microservices schützt gegen den Verfall der Architektur, sodass die Systeme auch langfristig wartbar bleiben. Zudem können Legacy-Systeme durch Microservices ergänzt werden, ohne dabei den alten Code zu ändern. Und auch Continuous Delivery ist einfacher umsetzbar. Eberhard Wolff bietet Ihnen in diesem Buch eine umfangreiche Einführung in das Thema Microservices. Dabei geht es u.a. um: Vor- und Nachteile des Microservice-Ansatzes Microservices vs. SOA Die übergreifende Architektur von Microservice-Systemen Die Architektur einzelner Services Auswirkungen auf Projektorganisation, Betrieb, Testen und Deployment Nanoservices Das Buch erläutert technologieneutrale Konzepte und Architekturen, die mit verschiedenen Technologien umgesetzt werden können. Als Beispiel für einen konkreten Technologie-Stack wird Java mit Spring Boot, dem Netflix-Stack und Spring Cloud gezeigt. Anhand von vielen Beispielen und konkreten Szenarien lernen Sie, wie Microservices möglichst gewinnbringend genutzt werden können. Außerdem erhalten Sie Anregungen, das Gelernte durch eigene Experimente weiter zu vertiefen. In der zweiten Auflage wurde der Abschnitt zu Domain-Driven Design komplett überarbeitet. Erweitert wurde die beispielhafte Beschreibung von Microservices-Technologien: Neben dem Netflix-Stack werden nun auch Alternativen erwähnt. Außerdem wurden die Essays zur Evolution von Microservices und zu Microservices in der Amazon Cloud aktualisiert.
Document the architecture of your software easily with this highly practical, open-source template. Key Features Get to grips with leveraging the features of arc42 to create insightful documents Learn the concepts of software architecture documentation through real-world examples Discover techniques to create compact, helpful, and easy-to-read documentation Book Description When developers document the architecture of their systems, they often invent their own specific ways of articulating structures, designs, concepts, and decisions. What they need is a template that enables simple and efficient software architecture documentation. arc42 by Example shows how it's done through several real-world examples. Each example in the book, whether it is a chess engine, a huge CRM system, or a cool web system, starts with a brief description of the problem domain and the quality requirements. Then, you'll discover the system context with all the external interfaces. You'll dive into an overview of the solution strategy to implement the building blocks and runtime scenarios. The later chapters also explain various cross-cutting concerns and how they affect other aspects of a program. What you will learn Utilize arc42 to document a system's physical infrastructure Learn how to identify a system's scope and boundaries Break a system down into building blocks and illustrate the relationships between them Discover how to describe the runtime behavior of a system Know how to document design decisions and their reasons Explore the risks and technical debt of your system Who this book is for This book is for software developers and solutions architects who are looking for an easy, open-source tool to document their systems. It is a useful reference for those who are already using arc42. If you are new to arc42, this book is a great learning resource. For those of you who want to write better technical documentation will benefit from the general concepts covered in this book.
APM steht für Agiles Projektmanagement und ist eine Methodik für die konsequente und praxisnahe Umsetzung agiler Projekte im Kontext anspruchsvoller Softwareprojekte. Der Leser erfährt in diesem Buch, wie er von der Projektvorbereitung und dem Requirements Engineering bis hin zu einer durchgängigen Softwarearchitektur agil entwickeln kann. Dabei wird auch auf das skalierbare und flexible APM-Rollenmodell eingegangen, um unterschiedlich große Projekte unter verschiedenen Rahmenbedingungen adressieren zu können. Das Buch gliedert sich in fünf Teile: - Teil I erläutert die Konzepte hinter dem Begriff Agilität und gibt einen Überblick über APM. - Teil II behandelt das Aufsetzen eines agilen Projekts. - Teil III legt dar, wie Softwarearchitektur und APM zusammenspielen. - Teil IV beschreibt detailliert die Struktur und Dynamik innerhalb von Iterationen sowie die fortlaufende Backlog-Arbeit hin zu hochwertigen Releases. Dabei wird auch auf Projektcontrolling sowie Kanban und Lean Management eingegangen. - Teil V zeigt, wie Sie APM für große Projekte skalieren und in verteilten Teams anwenden können. Erörtert werden auch die Besonderheiten im regulierten Umfeld und wie Agilität im Unternehmen eingeführt wird. APM stellt somit einen gut gefüllten Werkzeugkasten für viele unterschiedliche Situationen in agilen Projekten dar. Dem Buch liegt das zweiseitige Poster "Product-Owner-Werkzeugkoffer" und "Anforderungen agil zerlegen" bei.
Basiswissen Softwarearchitektur
Modern English for Aeronautics and Space Technology

Visualize, Program, and Create with JavaScript in p5.js
Agile Software Development in the Large
arc42 by Example
Typography for Screen

Software-Architekten müssen komplexe fachliche und technische Anforderungen an IT-Systeme umsetzen und diese Systeme durch nachvollziehbare Strukturen flexibel und erweiterbar gestalten. Dieser Praxisleitfaden zeigt Ihnen, wie Sie Software-Architekturen effektiv und systematisch entwickeln können. Der bekannte Software-Architekt Gernot Starke unterstützt Sie mit praktischen Tipps, Architekturmustern und seinen Erfahrungen. Er gibt Antworten auf zentrale Fragen: - Welche Aufgaben haben Software-Architekten? - Wie gehen Software-Architekten beim Entwurf vor? - Wie kommunizieren und dokumentieren Sie Software-Architekturen? - Wie helfen Architekturmuster und Architekturbausteine? - Wie bewerten Sie Software-Architekturen? - Wie behandeln Sie Persistenz, grafische Benutzeroberflächen, Geschäftsregeln, Integration, Verteilung, Sicherheit, Fehlerbehandlung, Workflow-Management und sonstige technische Konzepte? - Was müssen Software-Architekten über MDA/MDSD, UML 2 und arc42 wissen? - Welche Aufgaben nehmen Enterprise-IT-Architekten wahr? - Sie erfahren, wie die Dokumentation der Architektur von einer lästigen Pflicht zum integralen Kommunikations- und Arbeitsmittel wird. - Sie lernen, architekturrelevante Einflussfaktoren und zentrale Entscheidungen festzuhalten. - Sie erleben am Beispiel einer Schach-Engine, wie eine nachvollziehbare Architektur entsteht. - Auf der Buchwebsite finden Sie Vorlagen und weitere Informationen zum Thema und zu den Fallbeispielen. - Ihr exklusiver Vorteil: E-Book inside beim Kauf des gedruckten Buches Dokumentation wird oft als lästige Pflicht angesehen und in vielen Softwareprojekten stark vernachlässigt, die Architektur wird manchmal überhaupt nicht beschrieben. Damit das in Ihren Projekten nicht passiert, schlägt dieses Buch praxiserprobte und schlanke Bestandteile für eine wirkungsvolle Architekturdokumentation vor. Anhand eines durchgängigen Beispiels erfahren Sie, wie Sie architekturrelevante Einflussfaktoren erfassen und Ihre Softwarelösung angemessen und ohne Ballast festhalten. Sie lernen nicht nur die Vorgehensweise für das Dokumentieren während des Entwickelns kennen, sondern auch, wie Sie bestehende Systeme im Nachhinein beschreiben. Neben der Methodik diskutiert das Buch auch typische Formate und Werkzeuge wie Wikis, UML-Werkzeuge u.a., mit denen Sie Architekturdokumentation erfassen, verwalten und verbreiten können. Checklisten und Übungsaufgaben geben Ihnen die nötige Sicherheit, um die Architekturdokumentation zu einem integralen Bestandteil in Ihrem Softwarevorhaben zu machen. // Mein Fazit: Es gibt viele Bücher über Softwarearchitektur. Und dieses gehört zu denen, die man gelesen haben sollte, wenn man Softwareprojekte macht. // Phillip Ghadir zur ersten Auflage
This is the digital version of the printed book (Copyright © 2008). Adrenaline junkies, dead fish, project sluts, true believers, Lewis and Clark, template zombies . . . Most developers, testers, and managers on IT projects are pretty good at recognizing patterns of behavior and gut-level hunches, as in, “I sense that this project is headed for disaster.” But it has always been more difficult to transform these patterns and hunches into a usable form, something a team can debate, refine, and use. Until now. In Adrenaline Junkies and Template Zombies, the six principal consultants of The Atlantic Systems Guild present the patterns of behavior they most often observe at the dozens of IT firms they transform each year, around the world. The result is a quick-read guide to identifying nearly ninety typical scenarios, drawing on a combined one-hundred-and-fifty years of project management experience. Project by project, you’ll improve the accuracy of your hunches and your ability to act on them. The patterns are presented in an easy-reference format, with names designed to ease communication with your teammates. In just a few words, you can describe what’s happening on your project. Citing the patterns of behavior can help you quickly move those above and below you to the next step on your project. You’ll find classic patterns such as these: News Improvement Management by Mood Ring Piling On Rattle Yer Dags Natural Authority Food++ Fridge Door and more than eighty more! Not every pattern will be evident in your organization, and not every pattern is necessarily good or bad. However, you’ll find many patterns that will apply to your current and future assignments, even in the most ambiguous circumstances. When you assess your situation and follow your next hunch, you'll have the collective wisdom of six world-class consultants at your side.
Dieses Trainingsbuch vermittelt Ingenieurstudenten der Luft- und Raumfahrttechnik sowie Technikern, Ingenieuren und Managern in der Luft- und Raumfahrtindustrie praxisnah und effektiv einen fundierten Grundwortschatz. Die englischsprachigen Übungstexte sind übersichtlich strukturiert, schwierige Begriffe werden in deutscher Sprache erklärt. Das Lehrbuch ist als Lehrwerk für Hochschulen und Betriebe gedacht, in dem die Verbesserung der sprachlichen Kompetenzen in fachlicher/akademischer Ausrichtung im Vordergrund steht und nicht die Fachtexte an sich. Durch die Bearbeitung anspruchsvoller, authentischer und relevanter Texte können die Studierenden bzw. Kursteilnehmer/-innen ihre rezeptiven und produktiven Ausdrucksfähigkeiten in der englischen Sprache fürs Studium und für den Beruf verbessern. Zahlreiche, fachlich zugehörige Bilder verdeutlichen die Aussagen und verbessern das Verständnis. In der 2. Auflage gibt es folgende Änderungen: - neues Kapitel zu nachhaltigen Technologien in der Flugzeugbranche - komplette Erneuerung der Raumfahrttechnik, weil mittlerweile neue Technologien im Einsatz sind (Space Shuttle eingestellt)
Technica
Extensions of the SPES 2020 Methodology
Von starren Strukturen zu agilen Kulturen
Softwareentwickler führen und coachen, Hochleistungsteams aufbauen
The Hitchhiker's Guide to the Galaxy: The Illustrated Edition
C++ Crash Course
Diving Into the Deep

Grundlagenwissen nicht nur für Softwarearchitekten ... Softwarearchitektur bildet einen wesentlichen Erfolgsfaktor für Softwareprojekte. Sie stellt im Sinne einer systematischen Konstruktion sicher, dass Qualitätsanforderungen wie beispielsweise Erweiterbarkeit, Flexibilität, Performance oder Time-to-Market erfüllt werden können. "Basiswissen für Softwarearchitekten" vermittelt das notwendige Wissen und Fähigkeiten, um eine dem Problem angemessene Softwarearchitektur für Systeme zu entwerfen. Es behandelt die wichtigen Begriffe und Konzepte der Softwarearchitektur sowie deren Bezug zu anderen Disziplinen. Darauf aufbauend werden die grundlegenden Techniken und Methoden für den Entwurf, die Dokumentation und die Qualitätssicherung von Softwarearchitekturen beschrieben. Ausführlich behandelt werden zudem die Rolle, die Aufgaben, das Umfeld und die Arbeitsumgebung des Softwarearchitekten, ebenso dessen Einbettung in die umfassende Organisations- und Projektstruktur. Das Buch orientiert sich am Lehrplan zum "Certified Professional for Software Architecture – Foundation Level" (CPSA-F) des International Software Architecture Qualification Board (iSAQB). Die 4. Auflage bietet eine Aktualisierung auf Basis des CPSA-F-Lehrplans in der Version 5.1.
Agile Entwicklungsmethoden für Software, allen voran Scrum, sind auf dem Vormarsch und werden mittlerweile von der Mehrheit der Unternehmen eingesetzt. Leider werden bei den meisten Scrumeinführungen fundamentale Fehler gemacht, die dazu führen, dass die Prozesse versagen. Insbesondere die Grundregeln zur Organisationsentwicklung werden missachtet. Dieses Buch zeigt anhand einer theoretischen Einführung, Praxisbeispielen und einer Fallstudie, wie Scrum richtig eingeführt wird. Dabei wird sowohl auf unterschiedliche Zielzustände (ist ein einzelnes Team oder die ganze Organisation betroffen?), als auch auf verschiedene Herangehensweisen (Bottom-Up / Top-Down) eingegangen.
Thorough and continuous architecting is the key to overall success in software engineering, and architecture evaluation is a crucial part of it. This book presents a pragmatic architecture evaluation approach and insights gained from its application in more than 75 projects with industrial customers in the past decade. It presents context factors, empirical data, and example cases, as well as lessons learned on mitigating the risk of change through architecture evaluation. By providing comprehensive answers to more than 100 typical questions and discussing more than 60 frequent mistakes and lessons learned, the book allows readers to not only learn how to conduct architecture evaluations and interpret its results, but also to become aware of risks such as false conclusions, manipulating data, and unsound lines of argument. It equips readers to become confident in assessing quantitative measurement results and recognize when it is better to rely on qualitative expertise. The target readership includes both practitioners and researchers. By demonstrating its impact and providing clear guidelines, data, and examples, it encourages practitioners to conduct architecture evaluations. At the same time, it offers researchers insights into industrial architecture evaluations, which serve as the basis for guiding research in this area and will inspire future research directions.
Patterns kompakt fasst die wichtigsten Entwurfsmuster zusammen, die Sie für Software-Entwicklung benötigen. Software-Entwickler, -Architekten und -Designer finden darin effektiv anwendbare Lösungen für tägliche Entwurfsprobleme. Die vierte Auflage wurde um aktuelle Patterns erweitert und komplett überarbeitet. Das Buch gliedert Patterns anhand typischer Aspekte des Software-Entwurfs: Basismuster für mehr Flexibilität und Wartbarkeit, Präsentation, Kommunikation und Verteilung, Integration und Persistenz. Patterns kompakt richtet sich an Praktiker: Software-Entwickler, -Designer, -Architekten und alle, die einen praxisorientierten Überblick zu Entwurfsmustern benötigen.
Enabling Test-Driven Development, Domain-Driven Design, and Event-Driven Microservices
A Fast-Paced Introduction
Agile Software Architecture
Berufe der Informatik
Software architecture documentation in practice
Softwarearchitekturen dokumentieren und kommunizieren
Software Architecture Fundamentals

This beautifully illustrated edition of the New York Times bestselling classic celebrates the 42nd anniversary of the original publication—with all-new art by award-winning illustrator Chris Riddell. SOON TO BE A HULU SERIES • “An astonishing comic writer.”—Neil Gaiman Nominated as one of America’s best-loved novels by PBS’s The Great American Read It’s an ordinary Thursday morning for Arthur Dent . . . until his house gets demolished. The Earth follows shortly after to make way for a new hyperspace express route, and Arthur’s best friend has just announced that he’s an alien. After that, things get much, much worse. With just a towel, a small yellow fish, and a book, Arthur has to navigate through a very hostile universe in the company of a gang of unreliable aliens. Luckily the fish is quite good at languages. And the book is The Hitchhiker’s Guide to the Galaxy . . . which helpfully has the words DON’T PANIC inscribed in large, friendly letters on its cover. Douglas Adams’s mega-selling pop-culture classic sends logic into orbit, plays havoc with both time and physics, offers up pithy commentary on such things as ballpoint pens, potted plants, and digital watches . . . and, most important, reveals the ultimate answer to life, the universe, and everything. Now, if you could only figure out the question. . . .
Software services are established as a programming concept, but their impact on the overall architecture of enterprise IT and business operations is not well-understood. This has led to problems in deploying SOA, and some disillusionment. The SOA Source Book adds to this a collection of reference material for SOA. It is an invaluable resource for enterprise architects working with SOA.The SOA Source Book will help enterprise architects to use SOA effectively. It explains: What SOA is How to evaluate SOA features in business terms How to model SOA How to use The Open Group Architecture Framework (TOGAF™) for SOA SOA governance This book explains how TOGAF can help to make an Enterprise Architecture. Enterprise Architecture is an approach that can help management to understand this growing complexity.
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 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.
Understanding Patterns of Project Behavior
Soft Skills für IT-Führungskräfte und Projektleiter

Bio-Manufacturing
Windows XP Hacks
Software-Architekturen dokumentieren und kommunizieren
Scrum - Einführung in der Unternehmenspraxis
Architecture Patterns with Python

Unleash the power of Python 3 objects About This Book Stop writing scripts and start architecting programs Learn the latest Python syntax and libraries A practical, hands-on tutorial that teaches you all about abstract design patterns and how to implement them in Python 3 Who This Book Is For If you're new to object-oriented programming techniques, or if you have basic Python skills and wish to learn in depth how and when to correctly apply object-oriented programming in Python to design software, this is the book for you. What You Will Learn Implement objects in Python by creating classes and defining methods Separate related objects into a taxonomy of classes and describe the properties and behaviors of those objects via the class interface Extend class functionality using inheritance Understand when to use object-oriented features, and more importantly when not to use them Discover what design patterns are and why they are different in Python Uncover the simplicity of unit testing and why it's so important in Python Grasp common concurrency techniques and pitfalls in Python 3 Exploit object-oriented programming in key Python technologies such as Kivy and Django. Object-oriented programming concurrently with asyncio In Detail Python 3 is more versatile and easier to use than ever. It runs on all major platforms in a huge array of use cases. Coding in Python minimizes development time and increases productivity in comparison to other languages. Clean, maintainable code is easy to both read and write using Python's clear, concise syntax. Object-oriented programming is a popular design paradigm in which data and behaviors are encapsulated in such a way that they can be manipulated together. Many modern programming languages utilize the powerful concepts behind object-oriented programming and Python is no exception. Starting with a detailed analysis of object-oriented analysis and design, you will use the Python programming language to clearly grasp key concepts from the object-oriented paradigm. This book fully explains classes, data encapsulation, inheritance, polymorphism, abstraction, and exceptions with an emphasis on when you can use each principle to develop well-designed software. You'll get an in-depth analysis of many common object-oriented design patterns that are more suitable to Python's unique style. This book will not just teach Python syntax, but will also build your confidence in how to program. You will also learn how to create maintainable applications by studying higher level design patterns. Following this, you'll learn the complexities of string and file manipulation, and how Python distinguishes between binary and textual data. Not one, but two very powerful automated testing systems will be introduced in the book. After you discover the joy of unit testing and just how easy it can be, you'll study higher level libraries such as database connectors and GUI toolkits and learn how they uniquely apply object-oriented principles. You'll learn how these principles will allow you to make greater use of key members of the Python eco-system such as Django and Kivy. This new edition includes all the topics that made Python 3 Object-oriented Programming an instant Packt classic. It's also packed with updated content to reflect recent changes in the core Python library and covers modern third-party packages that were not available on the Python 3 platform when the book was first published. Style and approach Throughout the book you will learn key object-oriented programming techniques demonstrated by comprehensive case studies in the context of a larger project.

Stefan Malich entwickelt ein pattern-basiertes Wissensmodell, mit dem die Beziehungen zwischen den Entwurfsentscheidungen bezüglich der Softwarearchitektur und den daraus entstehenden Auswirkungen auf die Qualität des Systems ermittelt und dokumentiert werden können.

Today, web applications are the most important type of software applications. This textbook shows how to design and implement them, using a model-based engineering approach that covers general information management concepts and techniques and the two most relevant technology platforms: JavaScript and Java. The book provides an in-depth tutorial for theory-underpinned and example-based learning by doing it yourself, supported by quiz questions and practice projects. Volume 1 provides an introduction to web technologies and model-based web application engineering, discussing the information management concepts of constraint-based data validation, enumerations and special datatypes. Volume 2 discusses the advanced information management concepts of associations and inheritance in class hierarchies. Web apps are designed using UML class diagrams and implemented with two technologies: JavaScript for front-end (and distributed NodeJS) apps, and Java (with JPA and JSF) for back-end apps. The six example apps discussed in the book can be run, and their source code downloaded, from the book’s website. Gerd Wagner is Professor of Internet Technology at Brandenburg University of Technology, Germany, and Adjunct Associate Professor at Old Dominion University, Norfolk, VA, USA. He works in the areas of web engineering and modeling and simulation. Mircea Diaconescu is a Software Architect and Technical Team Leader at Entri GmbH, Berlin. He enjoys to work with the newest web technologies and to build Web of Things projects. Java, JavaScript/NodeJS and C# are his favorite programming languages.

As Python continues to grow in popularity, projects are becoming larger and more complex. Many Python developers are now taking an interest in high-level software design patterns such as hexagonal/clean architecture, event-driven architecture, and the strategic patterns prescribed by domain-driven design (DDD). But translating those patterns into Python isn’t always straightforward. With this hands-on guide, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most value out of their test suites. Each pattern is illustrated with concrete examples in beautiful, idiomatic Python, avoiding some of the verbosity of Java and C# syntax. Patterns include: Dependency inversion and its links to ports and adapters (hexagonal/clean architecture) Domain-driven design’s distinction between entities, value objects, and aggregates Repository and Unit of Work patterns for persistent storage Events, commands, and the message bus Command-query responsibility segregation (CQRS) Event-driven architecture and reactive microservices

Just Enough Software Architecture
Distributed Systems for System Architects
Advanced Model-Based Engineering of Embedded Systems
Effektive Softwarearchitekturen
Anspruchsvolle Softwareprojekte erfolgreich steuern
Embedded-Software entwickeln
Aligning Agile Processes and Software Architectures

Das Buch behandelt den zentralen Bereich "Entwurf" der Softwaretechnik, nämlich der Erstellung und Modifikation von Softwarearchitekturen, um so die Struktur eines Softwaresystems vor ihrer Detailrealisierung festzulegen. Die Softwarearchitektur enthält alle Grundlagen für Qualitätseigenschaften des später entstehenden Softwaresystems, insbesondere für Wartbarkeit. Auf dieser Betrachtungsebene finden ferner alle substantiellen Überlegungen zur Wiederverwendbarkeit statt. Hierzu werden geeignete Konzepte auf Architekturebene eingeführt, eine geeignete Architekturbeschreibungssprache vorgestellt, und es wird der methodische Umgang mit dieser Sprache geübt. Zur Übersicht werden graphische Architekturdiagramme für das Gesamtsystem zur Detailbeschreibung der Module textueller Beschreibungen verwendet. Die vorgestellten Konzepte bzw. Sprachen haben ihren Ursprung in Programmiersprachen. Trotzdem ist das Buch programmiersprachenunabhängig. Um dies zu zeigen, wird die Übertragung der Konzepte in FORTRAN, C und Pascal vorgeführt. Das Buch ist für den professionellen Software-Praktiker geschrieben. Es betont die softwaretechnische Bedeutung der objektbasierten Architekturmodellierung und ist sowohl kursbegleitend einsetzbar als auch zum Selbststudium geeignet.

Patterns kompakt
Verstehen, entwerfen, wiederverwenden
Aus- und Weiterbildung nach iSAQB-Standard zum Certified Professional for Software Architecture – Foundation Level
mit Funktionen in den Bereichen Wirtschaftsinformatik, technische Informatik, Organisation