Clean Architecture Gute Softwarearchitekturen Das

"Every developer working with the Web needs to read this book." -- David Heinemeier Hansson, creator of the Rails framework "RESTful Web Services finally provides a practical roadmap for constructing services that embrace the Web, instead of trying to route around it." -- Adam Trachtenberg, PHP author and EBay Web Services Evangelist You've built web sites that can be used by humans. But can you also build web sites that are usable by machines? That's where the future lies, and that's what RESTful Web Services shows you how to do. The World Wide Web is the most popular distributed application in history, and Web services and mashups have turned it into a powerful distributed computing platform. But today's web service technologies have lost sight of the simplicity that made the Web successful. They don't work like the Web, and they're missing out on its advantages. This book puts the "Web" back into web services. It shows how you can connect to the programmable web with the technologies you already use every day. The key is REST, the architectural style that drives the Web. This book: Emphasizes the power of basic Web technologies -- the HTTP application protocol, the URI naming standard, and the XML markup language Introduces the Resource-Oriented Architecture (ROA), a common-sense set of rules for designing RESTful web services Shows how a RESTful design is simpler, more versatile, and more scalable than a design based on Remote Procedure Calls (RPC) Includes real-world examples of RESTful web services, like Amazon's Simple Storage Service and the Atom Publishing Protocol Discusses web service clients for popular programming languages Shows how to implement RESTful services in three popular frameworks -- Ruby on Rails, Restlet (for Java), and Django (for Python) Focuses on practical issues: how to design and implement RESTful web services and clients This is the first book that applies the REST design philosophy to real web services. It sets down the best practices you need to make your design a success, and the techniques you need to turn your design into working code. You can harness the power of the Web for programmable applications: you just have to work with the Web instead of against it. This book shows you how.

If you want to speed up the development of your .NET applications, you're ready for C# design patterns -- elegant, accepted and proven ways to tackle common programming problems. This practical guide offers you a clear introduction to the classic object-oriented design patterns, and explains how to use the latest features of C# 3.0 to code them. C# Design Patterns draws on new C# 3.0 language and .NET 3.5 framework features to implement the 23 foundational patterns known to working developers. You get plenty of case studies that reveal how each pattern is used in practice, and an insightful comparison of patterns and where they would be best used or combined. This well-organized and illustrated book includes: An explanation of design patterns and why they're used, with tables and guidelines to help you choose one pattern over another Illustrated coverage of each classic Creational, Structural, and Behavioral design pattern, including its representation in UML and the roles of its various players C# 3.0 features introduced by example and summarized in sidebars for easy reference Examples of each pattern at work in a real .NET 3.5 program available for download from O'Reilly and the author's companion web site Quizzes and exercises to test your understanding of the material. With C# 3.0 Design Patterns, you learn to make code correct, extensible and efficient to save time up front and eliminate problems later. If your business relies on efficient application development and quality code, you need C# Design Patterns.

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.

Summary The Art of Unit Testing, Second Edition guides you step by step from writing your first simple tests to developing robust test sets that are maintainable, readable, and trustworthy. You'll master the foundational ideas and quickly move to high-value subjects like mocks, stubs, and isolation, including frameworks such as Moq, FakeItEasy, and Typemock Isolator. You'll explore test patterns and organization, working with legacy code, and even "untestable" code. Along the way, you'll learn about integration testing and techniques and tools for testing databases and other technologies. About this Book You know you should be unit testing, so why aren't you doing it? If you're new to unit testing, if you find unit testing tedious, or if you're just not getting enough payoff for the effort you put into it, keep reading. The Art of Unit Testing, Second Edition guides you step by step from writing your first simple unit tests to building complete test sets that are maintainable, readable, and trustworthy. You'll move quickly to more complicated subjects like mocks and stubs, while learning to use isolation (mocking) frameworks like Moq, FakeItEasy, and Typemock Isolator. You'll explore test patterns and organization, refactor code applications, and learn how to test "untestable" code. Along the way, you'll learn about integration testing and techniques for testing with databases. The examples in the book use C#, but will benefit anyone using a statically typed language such as Java or C++. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. What's Inside Create readable, maintainable, trustworthy tests Fakes, stubs, mock objects, and isolation (mocking) frameworks Simple dependency injection techniques Refactoring legacy code About the Author Roy Osherove has been coding for over 15 years, and he consults and trains teams worldwide on the gentle art of unit testing and test-driven development. His blog is at ArtOfUnitTesting.com. Table of Contents PART 1 GETTING STARTED The basics of unit testing A first unit test PART 2 CORE TECHNIQUES Using stubs to break dependencies Interaction testing using mock objects Isolation (mocking) frameworks Digging deeper into isolation frameworks PART 3 THE TEST CODE Test hierarchies and organization The pillars of good unit tests PART 4 DESIGN AND PROCESS Integrating unit testing into the organization Working with legacy code Design and testability

Sustainable Software Architecture

Microservices

The Art of Unit Testing

Das Praxishandbuch für gutes Softwaredesign. Regeln und Paradigmen für effiziente Softwarestrukturen

Architekturstile, Patterns und Best Practices

Clean Agile

Neuronale Netze und Deep Learning kapieren

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. Softwarearchitektur zeitgemäß und pragmatisch geplant Mark Richards und Neal Ford — Praktiker mit Erfahrung aus erster Hand, die seit Jahren das Thema Softwarearchitektur

unterrichten —, betrachten Softwarearchitektur vor dem Hintergrund der Entwicklungen, Innovationen und Herausforderungen des letzten Jahrzehnts. Sie konzentrieren sich auf Architekturprinzipien, die für alle Technologie-Stacks gelten. Angehende und erfahrene Architekten finden in diesem Buch umfassende Informationen zu architektonischen Merkmalen und Architekturstilen, zur Bestimmung von Komponenten, zur Diagrammerstellung und Präsentation, zu evolutionärer Architektur und vielen weiteren Themen. Die Autoren verstehen Softwarearchitektur als Engineering-Disziplin: mit wiederhol- und messbaren Ergebnissen und konkreten Kennzahlen für stabile Softwarearchitekturen.
Today's programmers don't develop software systems from scratch. instead, they spend their time fixing, extending, modifying, and enhancing existing software. Legacy systems often turn into an unwieldy mess that becomes increasingly difficult to modify, and with architecture that continually accumulates technical debt. Carola Lilienthal has analyzed more than 300 software systems written in Java, C#, C++, PHP, ABAP, and TypeScript and, together with her teams, has successfully refactored them. This book condenses her experience with monolithic systems, architectural and design patterns, layered architectures, domain-driven design, and microservices. With more than 200 color images from real-world systems, good and sub-optimal sample solutions are presented in a comprehensible and thorough way, while recommendations and suggestions based on practical projects allow the reader to directly apply the author's knowledge to their daily work. "Throughout the book, Dr. Lilienthal has provided sound advice on diagnosing, understanding, disentangling, and ultimately preventing the issues that make software systems brittle and subject to breakage. In addition to the technical examples that you'd expect in a book on software architecture, she takes the time to dive into the behavioral and human aspects that impact sustainability and, in my experience, are inextricably linked to the health of a codebase. She also expertly zooms out, exploring architecture concepts such as domains and layers, and then zooms in to the class level where your typical developer works day-to-day. This holistic approach is crucial for implementing long-lasting change." From the Foreword of Andrea Goulet CEO, Corgibytes, Founder, Legacy Code Rocks
From lambda expressions and JavaFX 8 to new support for network programming and mobile development, Java 8 brings a wealth of changes. This cookbook helps you get up to speed right away with hundreds of hands-on recipes across a broad range of Java topics. You'll learn useful techniques for everything from debugging and data structures to GUI development and functional programming. Each recipe includes self-contained code solutions that you can freely use, along with a discussion of how and why they work. If you are familiar with Java basics, this cookbook will bolster your knowledge of the language in general and Java 8's main APIs in particular. Recipes include: Methods for compiling, running, and debugging Manipulating, comparing, and rearranging text Regular expressions for string- and pattern-matching Handling numbers, dates, and times Structuring data with collections, arrays, and other types Object-oriented and functional programming techniques Directory and filesystem operations Working with graphics, audio, and video GUI development, including JavaFX and handlers Network programming on both client and server Database access, using JPA, Hibernate, and JDBC Processing JSON and XML for data storage Multithreading and concurrency
Pattern Enterpr Applica Arch
das Praxis-Handbuch für professionelles Softwaredesign : Regeln und Paradigmen für effiziente Softwarestrukturen
Design It!
UML 2.0 in a Nutshell
Lean UX
Wunder Informatik
A Code of Conduct for Professional Programmers

What others in the trenches say about The Pragmatic Programmer... "The cool thing about this book is that it's great for keeping the programming process fresh. The book helps you to continue to grow and clearly comes from people who have been there." —Kent Beck, author of Extreme Programming Explained: Embrace Change "I found this book to be a great mix of solid advice and wonderful analogies!" —Martin Fowler, author of Refactoring and UML Distilled "I would buy a copy, read it twice, then tell all my colleagues to run out and grab a copy. This is a book I would never loan because I would worry about it being lost." —Kevin Ruland, Management Science, MSG-Logistics "The wisdom and practical experience of the authors is obvious. The topics presented are relevant and useful.... By far its greatest strength for me has been the outstanding analogies—tracer bullets, broken windows, and the fabulous helicopter-based explanation of the need for orthogonality, especially in a crisis situation. I have little doubt that this book will eventually become an excellent source of useful information for journeymen programmers and expert mentors alike." —John Lakos, author of Large-Scale C++ Software Design "This is the sort of book I will buy a dozen copies of when it comes out so I can give it to my clients." —Eric Vought, Software Engineer "Most modern books on software development fail to cover the basics of what makes a great software developer, instead spending their time on syntax or technology where in reality the greatest leverage possible for any software team is in having talented developers who really know their craft well. An excellent book." —Pete McBreen, Independent Consultant "Since reading this book, I have implemented many of the practical suggestions and tips it contains. Across the board, they have saved my company time and money while helping me get my job done quicker! This should be a desktop reference for everyone who works with code for a living." —Jared Richardson, Senior Software Developer, iRenaissance, Inc. "I would like to see this issued to every new employee at my company...." —Chris Cleeland, Senior Software Engineer, Object Computing, Inc. "If I'm putting together a project, it's the authors of this book that I want. . . . And failing that I'd settle for people who've read their book." —Ward Cunningham Straight from the programming trenches, The Pragmatic Programmer cuts through the increasing specialization and technicalities of modern software development to examine the core process--taking a requirement and producing working, maintainable code that delights its users. It covers topics ranging from personal responsibility and career development to architectural techniques for keeping your code flexible and easy to adapt and reuse. Read this book, and you'll learn how to Fight software rot; Avoid the trap of duplicating knowledge; Write flexible, dynamic, and adaptable code; Avoid programming by coincidence; Bullet-proof your code with contracts, assertions, and exceptions; Capture real requirements; Test ruthlessly and effectively; Delight your users; Build teams of pragmatic programmers; and Make your developments more precise with automation. Written as a series of self-contained sections and filled with entertaining anecdotes, thoughtful examples, and interesting analogies, The Pragmatic Programmer illustrates the best practices and major pitfalls of many different aspects of software development. Whether you're a new coder, an experienced programmer, or a manager responsible for software projects, use these lessons daily, and you'll quickly see improvements in personal productivity, accuracy, and job satisfaction. You'll learn skills and develop habits and attitudes that form the foundation for long-term success in your career. You'll become a Pragmatic Programmer.

More C++ Gems picks up where the first book left off, presenting tips, tricks, proven strategies, easy-to-follow techniques, and usable source code.

Von den Grundlagen Neuronaler Netze über Machine Learning bis hin zu Deep-Learning-Algorithmen Anschauliche Diagramme, Anwendungsbeispiele in Python und der Einsatz von NumPy Keine Vorkenntnisse in Machine Learning oder höherer Mathematik erforderlich Deep Learning muss nicht kompliziert sein. Mit diesem Buch lernst du anhand vieler Beispiele alle Grundlagen, die du brauchst, um Deep-Learning-Algorithmen zu verstehen und anzuwenden. Dafür brauchst du nichts weiter als Schulmathematik und Kenntnisse der Programmiersprache Python. Alle Codebeispiele werden ausführlich erläutert und mathematische Hintergründe anhand von Analogien veranschaulicht. Der Autor erklärt leicht verständlich, wie Neuronale Netze lernen und wie sie mit Machine-Learning-Verfahren trainiert werden können. Du erfährst, wie du dein erstes Neuronales Netz erstellst und wie es mit Deep-Learning-Algorithmen Bilder erkennen sowie natürliche Sprache verarbeiten und modellieren kann. Hierbei kommen Netze mit mehreren Schichten wie CNNs und RNNs zum Einsatz. Fokus des Buches ist es, Neuronale Netze zu trainieren, ohne auf vorgefertigte Python-Frameworks zurückzugreifen. So verstehst du Deep Learning von Grund auf und kannst in Zukunft auch komplexe Frameworks erfolgreich für deine Projekte einsetzen. Aus dem Inhalt: Parametrische und nichtparametrische Modelle Überwachtes und unüberwachtes Lernen Vorhersagen mit mehreren Ein- und Ausgaben Fehler messen und verringern Hot und Cold Learning Batch- und stochastischer Gradientenabstieg Überanpassung vermeiden Generalisierung Dropout-Verfahren Backpropagation und Forward Propagation Bilderkennung Verarbeitung natürlicher Sprache (NLP) Sprachmodellierung Aktivierungsfunktionen: Sigmoid-Funktion Tangens hyperbolicus Softmax Convolutional Neural Networks (CNNs) Recurrent Neural Networks (RNNs) Long Short-Term Memory (LSTM) Deep-Learning-Framework erstellen

Domain Driven Design is a vision and approach for dealing with highly complex domains that is based on making the domain itself the main focus of the project, and maintaining a software model that reflects a deep understanding of the domain. This book is a short, quickly-readable summary and introduction to the fundamentals of DDD; it does not introduce any new concepts; it attempts to concisely summarize the essence of what DDD is, drawing mostly Eric Evans' original book, as well other sources since published such as Jimmy Nilsson's Applying Domain Driven Design, and various DDD discussion forums. The main topics covered in the book include: Building Domain Knowledge, The Ubiquitous Language, Model Driven Design, Refactoring Toward Deeper Insight, and Preserving Model Integrity. Also included is an interview with Eric Evans on Domain Driven Design today.

The Pragmatic Programmer

Analyze and Reduce Technical Debt

with examples in C#

Java Cookbook

21. Fachgespräch Karlsruhe, 3./4. Dezember 2009

Domain-Driven Design Quickly

Handbuch moderner Softwarearchitektur

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

Programmers who endure and succeed amidst swirling uncertainty and nonstop pressure share a common attribute: They care deeply about the practice of creating software. They treat it as a craft. They are professionals. In The Clean Coder: A Code of Conduct for Professional Programmers, legendary software expert Robert C. Martin introduces the disciplines, techniques, tools, and practices of true software craftsmanship. This book is packed with practical advice–about everything from estimating and coding to refactoring and testing. It covers much more than technique: It is about attitude. Martin shows how to approach software development with honor, self-respect, and pride; work well and work clean; communicate and estimate faithfully; face difficult decisions with clarity and honesty; and understand that deep knowledge comes with a responsibility to act. Readers will learn What it means to behave as a true software craftsman How to deal with conflict, tight schedules, and unreasonable managers How to get into the flow of coding, and get past writer's block How to handle unrelenting pressure and avoid burnout How to combine enduring attitudes with new development paradigms How to manage your time, and avoid blind alleys, marshes, bogs, and swamps How to foster environments where programmers and teams can thrive When to say "No"–and how to say it When to say "Yes"–and what yes really means Great software is something to marvel at: powerful, elegant, functional, a pleasure to work with as both a developer and as a user. Great software isn't

written by machines. It is written by professionals with an unshakable commitment to craftsmanship. The Clean Coder will help you become one of them–and earn the pride and fulfillment that they alone possess.

Collaboration among individuals – from users to developers – is central to modern software engineering. It takes many forms: joint activity to solve common problems, negotiation to resolve conflicts, creation of shared definitions, and both social and technical perspectives impacting all software development activity. The difficulties of collaboration are also well documented. The grand challenge is not only to ensure that developers in a team deliver effectively as individuals, but that the whole team delivers more than just the sum of its parts. The editors of this book have assembled an impressive selection of authors, who have contributed to an authoritative body of work tackling a wide range of issues in the field of collaborative software engineering. The resulting volume is divided into four parts, preceded by a general editorial chapter providing a more detailed review of the domain of collaborative software engineering. Part 1 is on "Characterizing Collaborative Software Engineering", Part 2 examines various "Tools and Techniques", Part 3 addresses organizational issues, and finally Part 4 contains four examples of "Emerging Issues in Collaborative Software Engineering". As a result, this book delivers a comprehensive state-of-the-art overview and empirical results for researchers in academia and industry in areas like software process management, empirical software engineering, and global software development. Practitioners working in this area will also appreciate the detailed descriptions and reports which can often be used as guidelines to improve their daily work.

Extreme Programming is the most exciting revolution to hit the software engineering industry in the last decade. But what exactly is XP? And how do you XP? Simply put, XP is about playing to win. If you are serious about becoming an agile organization, decreasing your time to market, keeping your development team happy, and improving the overall quality of your software, then XP is for you. Extreme Programming in Practice provides a candid, refreshing, insiders view of how an XP project works. The artifacts presented in this book are real, the user stories are real, and the anecdotes are real. The book represents all-access, uncensored XP. The authors have chosen example over explanation, so that you can personalize the tenets of XP and put them into practice on your next development project. The book is supported with sample code and test examples. You can learn how to emphasize planning in your project; deliver multiple iterations of your project (each with increasing business value); gather customer feedback as you build; and test the integrity of your code without halting your development efforts. The authors also provide a handy summary of more than a dozen lessons learned i

Crafting Interpreters
Produktentwicklung und -design mit agilen Teams
WORK EFFECT LEG CODE _p1
Clean Agile. Die Essenz der agilen Softwareentwicklung
Microsoft Excel 2019 VBA and Macros
Extreme Programming in Practice
Clean architecture

The practice of enterprise application development has benefited from the emergence of many new enabling technologies. Multi-tiered object-oriented platforms, such as Java and .NET, have become commonplace. These new tools and technologies are capable of building powerful applications, but they are not easily implemented. Common failures in enterprise applications often occur because their developers do not understand the architectural lessons that experienced object developers have learned. Patterns of Enterprise Application Architecture is written in direct response to the stiff challenges that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology--from Smalltalk to CORBA to Java to .NET--the same basic design ideas can be adapted and applied to solve common problems. With the help of an expert group of contributors, Martin distills over forty recurring solutions into patterns. The result is an indispensable handbook of solutions that are applicable to any enterprise application platform. This book is actually two books in one. The first section is a short tutorial on developing enterprise applications, which you can read from start to finish to understand the scope of the book's lessons. The next section, the bulk of the book, is a detailed reference to the patterns themselves. Each pattern provides usage and implementation information, as well as detailed code examples in Java or C#. The entire book is also richly illustrated with UML diagrams to further explain the concepts. Armed with this book, you will have the knowledge necessary to make important architectural decisions about building an enterprise application and the proven patterns for use when building them. The topics covered include · Dividing an enterprise application into layers · The major approaches to organizing business logic · An in-depth treatment of mapping between objects and relational databases · Using Model-View-Controller to organize a Web presentation · Handling concurrency for data that spans multiple transactions · Designing distributed object interfaces

Die Informatik ist das wichtigste Werkzeug des 21. Jahrhunderts. Die ganze Welt spricht in den Sprachen der Informatik. Das gilt für alle Bereiche der modernen Welt und zunehmend auch für unser privates Leben. Man kann die Welt ohne Informatik nicht mehr verstehen und nicht mehr verändern. Doch es gibt auch viele Missverständnisse über die Informatik. Das liegt daran, dass sie eine junge Wissenschaft ist, die sich permament und schnell verändert: von ihren Anfängen in den 1940er-Jahren als Rechenmaschine für Chemiker und Physiker bis zum Smartphone und der Cloud. Dieses Buch ermöglicht eine intuitive Einführung in die Informatik. Es beschreibt die grundlegenden Konzepte und erläutert Teilbereiche wie Data Science, Big Data oder künstliche Intelligenz. Vor allem aber entmystifiziert es die Welt der Informatik anhand vieler Alltagsbeispiele. Es muss nicht jeder Informatik studieren oder ein Experte sein. Doch es soll jeder von ihren Ideen und Innovationen profitieren. Das Buch richtet sich einerseits an Jugendliche: Sie erhalten einen Einblick, was sie in einem Studium oder auf dem Berufsweg erwartet. Sie sollen weder zufällig Informatik studieren noch zufällig nicht Informatik studieren. Doch es ist auch für Erwachsene relevant, die eine allgemeinverständliche Einführung suchen, in die auch viele persönliche Erfahrungen eingeflossen sind.

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).

Despite using them every day, most software engineers know little about how programming languages are designed and implemented. For many, their only experience with that corner of computer science was a terrifying "compilers" class that they suffered through in undergrad and tried to blot from their memory as soon as they had scribbled their last NFA to DFA conversion on the final exam. That fearsome reputation belies a field that is rich with useful techniques and not so difficult as some of its practitioners might have you believe. A better understanding of how programming languages are built will make you a stronger software engineer and teach you concepts and data structures you'll use the rest of your coding days. You might even have fun. This book teaches you everything you need to know to implement a full-featured, efficient scripting language. You'll learn both high-level concepts around parsing and semantics and gritty details like bytecode representation and garbage collection. Your brain will light up with new ideas, and your hands will get dirty and calloused. Starting from main(), you will build a language that features rich syntax, dynamic typing, garbage collection, lexical scope, first-class functions, closures, classes, and inheritance. All packed into a few thousand lines of clean, fast code that you thoroughly understand because you wrote each one yourself.

APM - Agiles Projektmanagement

A Handbook of Agile Software Craftsmanship

Technische Schulden analysieren, begrenzen und abbauen

At Home, But Not Alone.

The Art of Readable Code

Zurück zu den Ursprüngen: Die agilen Werte und Prinzipien effektiv in der Praxis umsetzen

The Clean Coder

Praktische Losungen fur den Aufbau von Softwarearchitekturen von dem legendaren Softwareentwickler Robert C. MArtin ("Uncle Bob") Allgemeingultige Regeln fur die Verbesserung der Produktivitat in der Softwareentwicklung uber den gesamten Lebenszyklus Wie Softwareentwickler wesentliche Prinzipien des Softwaredesigns meistern, warum Softwarearchitekturen haufig scheitern und wie man solche Fehlschlage verhindern kann Wirklich gute Software zu entwickeln, ist ein schwieriges Unterfangen und eine groSSe Herausforderung. ABer wenn Software in der richtigen Art und Weise entwickelt wird, erfordert die Erstellung und Instandhaltung nur wenige Ressourcen, Modifikationen und Anpassungen lassen sich schnell und einfach umsetzen und Mangel und Fehler treten nur hin und wieder in Erscheinung. DEr Entwicklungsaufwand ist minimal, und das bei maximaler Funktionalitat und Flexibilitat. Was hier utopisch klingt, hat Robert C. MArtin schon selbst erlebt und weiSS deshalb, dass es so funktionieren kann. Als Entwickler konnen Sie Ihre Produktivitat uber die Lebenszeit eines jeden Softwaresystems dramatisch verbessern, indem Sie allgemeingultige Grundsatze fur die Entwicklung professioneller Softwarearchitektur anwenden. IN diesem Buch verrat Ihnen der legendare Softwareentwickler diese maSSgeblichen Prinzipien und zeigt Ihnen, wie Sie diese erfolgreich und effektiv anwenden. Basierend auf seiner mehr als 50-jahrigen Berufserfahrung mit Softwareumgebungen jeder erdenklichen Art demonstriert Robert C. MArtin in diesem Buch auf eindrucksvolle Weise, welche Entscheidungen Sie im Entwicklungsprozess treffen sollten und warum diese fur Ihren Erfolg ausschlaggebend sind. WIe man es von "Uncle Bob" kennt, enthalt dieses Buch zahlreiche unmittelbar anwendbare und in sich schlussige Losungen fur die Herausforderungen, mit denen Sie im Berufsleben konfrontiert sein werden - jenen, die uber Gedeih und Verderb Ihrer Projekte entscheiden. In diesem Buch lernen Sie: Architektonische Zielsetzungen der Softwareentwicklung richtig abstecken und die dafur notwendigen Kerndisziplinen und -praktiken planvoll einsetzen Die grundlegenden Prinzipien des Softwaredesigns fur den Umgang mit Funktionalitat, Komponententrennung und Datenmanagement meistern Den Entwicklungsprozess optimieren durch die zielgerichtete Anwendung von Programmierparadigmen und die klare Definition der Handlungsspielraume der Softwareentwickler Wichtige systemrelevante Programmbestandteile von bloSSen "Details" unterscheiden Optimale, hochschichtige Strukturen fur Web, Datenbank, Fat Client, Konsole und eingebettete Anwendungen implementieren Angemessene Grenzen und Layer definieren und die Komponenten und Services in Ihrem System organisieren Faktoren fur das Scheitern von Softwaredesigns und -architekturen erkennen und diese Fehler vermeiden Clean Architecture ist fur jeden gegenwartigen oder angehenden Softwarearchitekten, Systemanalysten, Systemdesigner und Softwaremanager eine Pflichtlekture - ebenso wie fur jeden Programmierer, der die Softwaredesigns anderer Entwickler ausfuhren muss.

This comprehensive guide has been fully revised to cover UML 2.0, today's standard method for modelling software systems. Filled with concise information, it's been crafted to help IT professionals read, create, and understand system artefacts expressed using UML. Includes an example-rich tutorial for those who need familiarizing with the system.

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.

Even bad code can function. But if code isn't clean, it can bring a development organization to its knees. Every year, countless hours and significant resources are lost because of poorly written code. But it doesn't have to be that way. Noted software expert Robert C. Martin presents a revolutionary paradigm with Clean Code: A Handbook of Agile Software Craftsmanship . Martin has teamed up with his colleagues from Object Mentor to distill their best agile practice of cleaning code "on the fly" into a book that will instill within you the values of a software craftsman and make you a better programmer-but only if you work at it. What kind of work will you be doing? You'll be reading code-lots of code. And you will be challenged to think about what's right about that code, and what's wrong with it. More importantly, you will be challenged to reassess your professional values and your commitment to your craft. Clean Code is divided into three parts. The first describes the principles, patterns, and practices of writing clean code. The second part consists of several case studies of increasing complexity. Each case study is an exercise in cleaning up code-of transforming a code base that has some problems into one that is sound and efficient. The third part is the payoff: a single chapter containing a list of heuristics and "smells" gathered while creating the case studies. The result is a knowledge base that describes the way we think when we write, read, and clean code. Readers will come away from this book understanding How to tell the difference between good and bad code How to write good code and how to transform bad code into good code How to create good names, good functions, good objects, and good classes How to format code for maximum readability How to implement complete error handling without obscuring code logic How to unit test and practice test-driven development This book is a must for any developer, software engineer, project manager, team lead, or systems analyst with an interest in producing better code.

Working Effectively with Legacy Code

A Practical Guide to Continuous Delivery

arc42 by Example

Obey the Testing Goat: Using Django, Selenium, and JavaScript

Clean Architecture

From Journeyman to Master

Langlebige Software-Architekturen

Summary Functional and Reactive Domain Modeling teaches you how to think of the domain model in terms of pure functions and how to compose them to build larger abstractions. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Traditional distributed applications won't cut it in the reactive world of microservices, fast data, and sensor networks. To capture their dynamic relationships and dependencies, these systems require a different approach to domain modeling. A domain model composed of pure functions is a more natural way of representing a process in a reactive system, and it maps directly onto technologies and patterns like Akka, CQRS, and event sourcing. About the Book Functional and Reactive Domain Modeling teaches you consistent, repeatable techniques for building domain models in reactive systems. This book reviews the relevant concepts of FP and reactive architectures and then methodically introduces this new approach to domain modeling. As you read, you'll learn where and how to apply it, even if your systems aren't purely reactive or functional. An expert blend of theory and practice, this book presents strong examples you'll return to again and again as you apply these principles to your own projects. What's Inside Real-world libraries and frameworks Establish meaningful reliability guarantees Isolate domain logic from side effects Introduction to reactive design patterns About the Reader Readers should be comfortable with functional programming and traditional domain modeling. Examples use the Scala language. About the Author Software architect Debasish Ghosh was an early adopter of reactive design using Scala and Akka. He's the author of DSLs in Action, published by Manning in 2010. Table of Contents Functional domain modeling: an introduction Scala for functional domain models Designing functional domain models Functional patterns for domain models Modularization of domain models Being reactive Modeling with reactive streams Reactive persistence and event sourcing Testing your domain model Summary - core thoughts and principles

• Lernen Sie aus Uncle Bobs jahrzehntelanger Erfahrung, worauf es bei der agilen Softwareentwicklung wirklich ankommt • Die ursprünglichen agilen Werte und Prinzipien kurz und prägnant für den Praxiseinsatz erläutert • Von den unternehmerischen Aspekten über die Kommunikation im Team bis zu den technischen Praktiken wie Test-Driven Development (TDD), einfaches Design und Pair Programming Fast 20 Jahre nach der Veröffentlichung des agilen Manifests ruft der legendäre Softwareentwickler Robert C. Martin (»Uncle Bob«) dazu auf, sich wieder auf die ursprünglichen Werte und Prinzipien zurückzubesinnen, die den eigentlichen Kern der agilen Softwareentwicklung ausmachen und die für die Praxis von zentraler Bedeutung sind. Mit Clean Agile lässt er alle an seiner jahrzehntelangen Erfahrung teilhaben und räumt mit Missverständnissen und Fehlinterpretationen auf, die im Laufe der Jahre entstanden sind. Dabei wendet er sich gleichermaßen an Programmierer und Nicht-Programmierer. Uncle Bob macht deutlich, was agile Softwareentwicklung eigentlich ist, war und immer sein sollte: ein einfaches Konzept, das kleinen Softwareteams hilft, kleine Projekte zu managen – denn daraus setzen sich letztendlich alle großen Projekte zusammen. Dabei konzentriert er sich insbesondere auf die Praktiken des Extreme Programmings (XP), ohne sich in technischen Details zu verlieren. Egal, ob Sie Entwickler, Tester, Projektmanager oder Auftraggeber sind – dieses Buch zeigt Ihnen, worauf es bei der Umsetzung agiler Methoden wirklich ankommt.

By taking you through the development of a real web application from beginning to end, the second edition of this hands-on guide demonstrates the practical advantages of test-driven development (TDD) with Python. You'll learn how to write and run tests before building each part of your app, and then develop the minimum amount of code required to pass those tests. The result? Clean code

that works. In the process, you'll learn the basics of Django, Selenium, Git, jQuery, and Mock, along with current web development techniques. If you're ready to take your Python skills to the next level, this book—updated for Python 3.6—clearly demonstrates how TDD encourages simple designs and inspires confidence. Dive into the TDD workflow, including the unit test/code cycle and refactoring Use unit tests for classes and functions, and functional tests for user interactions within the browser Learn when and how to use mock objects, and the pros and cons of isolated vs. integrated tests Test and automate your deployments with a staging server Apply tests to the third-party plugins you integrate into your site Run tests automatically by using a Continuous Integration environment Use TDD to build a REST API with a front-end Ajax interface

As programmers, we've all seen source code that's so ugly and buggy it makes our brain ache. Over the past five years, authors Dustin Boswell and Trevor Foucher have analyzed hundreds of examples of "bad code" (much of it their own) to determine why they're bad and how they could be improved. Their conclusion? You need to write code that minimizes the time it would take someone else to understand it—even if that someone else is you. This book focuses on basic principles and practical techniques you can apply every time you write code. Using easy-to-digest code examples from different languages, each chapter dives into a different aspect of coding, and demonstrates how you can make your code easy to understand. Simplify naming, commenting, and formatting with tips that apply to every line of code Refine your program's loops, logic, and variables to reduce complexity and confusion Attack problems at the function level, such as reorganizing blocks of code to do one task at a time Write effective test code that is thorough and concise—as well as readable "Being aware of how the code you create affects those who look at it later is an important part of developing software. The authors did a great job in taking you through the different aspects of this challenge, explaining the details with instructive examples." —Michael Hunger, passionate Software Developer

Anspruchsvolle Softwareprojekte erfolgreich steuern

C# 3.0 Design Patterns

Das Praxis-Handbuch für professionelles Softwaredesign.Regeln und Paradigmen für effiziente Softwarestrukturierung.

Der einfache Praxiseinstieg mit Beispielen in Python

Collaborative Software Engineering

Solutions and Examples for Java Developers

A Study Guide for the Certified Professional for Software Architecture® – Foundation Level – iSAQB compliant

The Most Complete, Practical, and Actionable Guide to Microservices Going beyond mere theory and marketing hype, Eberhard Wolff presents all the knowledge you need to capture the full benefits of this emerging paradigm. He illuminates microservice concepts, architectures, and scenarios from a technology-neutral standpoint, and demonstrates how to implement them with today's leading technologies such as Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud. The author fully explains the benefits and tradeoffs associated with microservices, and guides you through the entire project lifecycle: development, testing, deployment, operations, and more. You'll find best practices for architecting microservice-based systems, individual microservices, and nanoservices, each illuminated with pragmatic examples. The author supplements opinions based on his experience with concise essays from other experts, enriching your understanding and illuminating areas where experts disagree. Readers are challenged to experiment on their own the concepts explained in the book to gain hands-on experience. Discover what microservices are, and how they differ from other forms of modularization Modernize legacy applications and efficiently build new systems Drive more value from continuous delivery with microservices Learn how microservices differ from SOA Optimize the microservices project lifecycle Plan, visualize, manage, and evolve architecture Integrate and communicate among microservices Apply advanced architectural techniques, including CQRS and Event Sourcing Maximize resilience and stability Operate and monitor microservices in production Build a full implementation with Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud Explore nanoservices with Amazon Lambda, OSGi, Java EE, Vert.x, Erlang, and Seneca Understand microservices' impact on teams, technical leaders, product owners, and stakeholders Managers will discover better ways to support microservices, and learn how adopting the method affects the entire organization. Developers will master the technical skills and concepts they need to be effective. Architects will gain a deep understanding of key issues in creating or migrating toward microservices, and exactly what it will take to transform their plans into reality.

Don't engineer by coincidence-design it like you mean it! Filled with practical techniques, Design It! is the perfect introduction to software architecture for programmers who are ready to grow their design skills. Lead your team as a software architect, ask the right stakeholders the right questions, explore design options, and help your team implement a system that promotes the right -ilities. Share your design decisions, facilitate collaborative design workshops that are fast, effective, and fun-and develop more awesome software! With dozens of design methods, examples, and practical know-how, Design It! shows you how to become a software architect. Walk through the core concepts every architect must know, discover how to apply them, and learn a variety of skills that will make you a better programmer, leader, and designer. Uncover the big ideas behind software architecture and gain confidence working on projects big and small. Plan, design, implement, and evaluate software architectures and collaborate with your team, stakeholders, and other architects. Identify the right stakeholders and understand their needs, dig for architecturally significant requirements, write amazing quality

attribute scenarios, and make confident decisions. Choose technologies based on their architectural impact, facilitate architecture-centric design workshops, and evaluate architectures using lightweight, effective methods. Write lean architecture descriptions people love to read. Run an architecture design studio, implement the architecture you've designed, and grow your team's architectural knowledge. Good design requires good communication. Talk about your software architecture with stakeholders using whiteboards, documents, and code, and apply architecture-focused design methods in your day-to-day practice. Hands-on exercises, real-world scenarios, and practical team-based decision-making tools will get everyone on board and give you the experience you need to become a confident software architect.

Remote Mob Programming combines two ways of working: Mob Programming and working as a distributed team. Jochen Christ, Simon Harrer and Martin Huber share their experience with their working model - and why they don't want to work differently anymore.

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

Autonome Mobile Systeme 2009

Software Architecture Fundamentals

Just Enough Software Architecture

More C++ Gems

Flexible Software Architecture

Clean Code

A Risk-Driven Approach

Agile Values and Principles for a New Generation “In the journey to all things Agile, Uncle Bob has been there, done that, and has the both the t-shirt and the scars to show for it. This delightful book is part history, part personal stories, and all wisdom. If you want to understand what Agile is and how it came to be, this is the book for you.” -Grady Booch “Bob’s frustration colors every sentence of Clean Agile, but it’s a justified frustration. What is in the world of Agile development is nothing compared to what could be. This book is Bob’s perspective on what to focus on to get to that ‘what could be.’ And he’s been there, so it’s worth listening.” -Kent Beck “It’s good to read Uncle Bob’s take on Agile. Whether just beginning, or a seasoned Agilista, you would do well to read this book. I agree with almost all of it. It’s just some of the parts make me realize my own shortcomings, dammit. It made me double-check our code coverage (85.09%).” -Jon Kern Nearly twenty years after the Agile Manifesto was first presented, the legendary Robert C. Martin (“Uncle Bob”) reintroduces Agile values and principles for a new generation-programmers and nonprogrammers alike. Martin, author of Clean Code and other highly influential software development guides, was there at Agile’s founding. Now, in Clean Agile: Back to Basics, he strips away misunderstandings and distractions that over the years have made it harder to use Agile than was originally intended. Martin describes what Agile is in no uncertain terms: a small discipline that helps small teams manage small projects . . . with huge implications because every big project is comprised of many small projects. Drawing on his fifty years’ experience with projects of every conceivable type, he shows how Agile can help you bring

true professionalism to software development. Get back to the basics–what Agile is, was, and should always be Understand the origins, and proper practice, of SCRUM Master essential business-facing Agile practices, from small releases and acceptance tests to whole-team communication Explore Agile team members' relationships with each other, and with their product Rediscover indispensable Agile technical practices: TDD, refactoring, simple design, and pair programming Understand the central roles values and craftsmanship play in your Agile team's success If you want Agile's true benefits, there are no shortcuts: You need to do Agile right. Clean Agile: Back to Basics will show you how, whether you're a developer, tester, manager, project manager, or customer. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

Standardwerk zur SoftwarearchitekturSchließt die Lücke zwischen Softwarearchitektur und Implementierung der Codebasis Einfache und übersichtliche Strukturierung aller wichtigen Grundkonzepte im Bereich der Softwarearchitektur, die die typischen Fehler in der Softwarearchitektur von großen Softwaresysteme aufzeigen und sinnvolle Lösungen vermitteln. Mit über 200 farbigen Bildern aus real existierenden Softwaresystemen und etlichen Fallbeispielen Zu Beginn eines Projekts erarbeiten die Architekten und das Entwicklungsteam eine zugeschnittene Architekturblaupause für die anstehende Entwicklung. Aber während der Implementierung weicht das Team häufig ungewollt von dieser Vorgabe ab. Die Folge davon: Die Architektur des Systems erodiert, die Komplexität nimmt zu, und es werden technische Schulden aufgebaut. Wartung und Erweiterung der Software werden immer aufwendiger. In diesem Buch zeigt die Autorin, welche Fehler in Softwareprojekten bei der Umsetzung der Architektur vermieden werden sollten und welche Prinzipien eingehalten werden müssen, um langlebige Architekturen zu entwerfen oder bei bestehenden Systemen zu langlebigen Architekturen zu gelangen. Sie geht auf Muster in Softwarearchitekturen und Mustersprachen ein, erläutert verschiedene Architekturstile und zeigt, welche Vorgaben letztlich zu Architekturen führen, die für Entwickler noch gut durchschaubar sind. Mit über 200 farbigen Bildern aus real existierenden Softwaresystemen und etlichen Fallbeispielen werden schlechte und gute Lösungen verständlich und nachvollziehbar dargestellt. Empfehlungen und vielfältige Hinweise aus Praxisprojekten erlauben dem Leser einen direkten Transfer zu seiner täglichen Arbeit. Die 3. Auflage wurde in einzelnen Aspekten überarbeitet und insbesondere bei den Themen Domain-Driven Design und Microservices ergänzt. Neu aufgenommen wurden die Analyse von TypeScript-Systemen sowie Clean-, Onion- und hexagonale Architekturen.

Get more out of your legacy systems: more performance, functionality, reliability, and manageability Is your code easy to change? Can you get nearly instantaneous feedback when you do change it? Do you understand it? If the answer to any of these questions is no, you have legacy code, and it is draining time and money away from your development efforts. In this book, Michael Feathers offers start-to-finish strategies for working more effectively with large, untested legacy code bases. This book draws on material Michael created for his renowned Object Mentor seminars: techniques Michael has used in mentoring to help hundreds of developers, technical managers, and testers bring their legacy systems under control. The topics covered include Understanding the mechanics of software change: adding features, fixing bugs, improving design, optimizing performance Getting legacy code into a test harness Writing tests that protect you against introducing new problems Techniques that can be used with any language or platform—with examples in Java, C++, C, and C# Accurately identifying where code changes need to be made Coping with legacy systems that aren't object-oriented Handling applications that don't seem to have any structure This book also includes a catalog of twenty-four dependency-breaking techniques that help you work with program elements in isolation and make safer changes.

Der Band dokumentiert das 21. Fachgespräch Autonome Mobile Systeme (AMS 2009). Die Veranstaltung bietet Wissenschaftlern aus Forschung und Industrie ein Forum für den Gedankenaustausch und eine Basis, um Kooperationen auf diesem Forschungsgebiet zu initiieren. Die Beiträge befassen sich mit Themen wie humanoide Roboter und Flugmaschinen, Perzeption und Sensorik, Kartierung und Lokalisation, Regelung, Navigation, Lernverfahren, Systemarchitekturen sowie mit der Anwendung von autonomen mobilen Systemen.

From Programmer to Software Architect

RESTful Web Services

Use the Power of C# 3.0 to Solve Real-World Problems

Software architecture documentation in practice

Simple and Practical Techniques for Writing Better Code
Test-Driven Development with Python
A Philosophy of Software Design

Lean UX effektiv im Unternehmen implementieren Vorhandene Strukturen anpassen und interdisziplinäre Teams bilden Mit Lean UX schlanke und schnell lieferbare Produktversionen erstellen Lean UX hat sich zum populärsten Ansatz für das Interaction Design entwickelt, es passt genau zu den Anforderungen agiler Teams von heute. Jeff Gothelf und Josh Seiden, Pioniere und führende Experten für Lean UX, erläutern in diesem Buch umfassend die zentralen Prinzipien, Taktiken und Techniken dieser Entwicklungsmethode und zeigen, wie Produktteams ganz einfach Design, Experimente, Iteration und kontinuierliches Lernen echter User in ihren agilen Prozess integrieren können. Lean UX ist inspiriert von den Konzepten des Lean Developments sowie anderer agiler Entwicklungsmethoden und hat den Vorteil, dass Sie sich vor allem auf das Designen der eigentlichen User Experience statt auf die Deliverables konzentrieren können. Dieses Buch zeigt Ihnen, wie Sie eng mit anderen Mitgliedern des Produktteams zusammenarbeiten und Feedback von Usern häufig und frühzeitig erfassen und berücksichtigen können. Außerdem erfahren Sie, wie sich der Designprozess in kurzen iterativen Zyklen vorantreiben lässt, um herauszufinden, was sowohl in geschäftlicher Hinsicht als auch aus Sicht der User am besten funktioniert. »Lean UX« weist Ihnen den Weg, wie Sie dieses Umdenken in Ihrem Unternehmen herbeiführen können – eine Wendung zum Besseren. - Visualisieren Sie das Problem, das Sie zu lösen versuchen, und fokussieren Sie Ihr Team auf die »richtigen« Ergebnisse - Vermitteln Sie dem gesamten Produktteam das Designer Toolkit - Lassen Sie Ihr Team sehr viel früher als üblich an Ihren Erkenntnissen teilhaben - Erstellen Sie MVPs (Minimum Viable Products), um in Erfahrung zu bringen, welche Ideen und Konzepte funktionieren - Beziehen Sie die »Stimme des Kunden« in den gesamten Projektzyklus mit ein - Kombinieren Sie Lean UX mit dem agilen Scrum-Framework und steigern Sie so die Produktivität Ihres Teams - Setzen Sie sich mit den organisatorischen Veränderungen auseinander, die zur Anwendung und Integration der Lean-UX-Methode erforderlich sind

Fowler
Functional and Reactive Domain Modeling
Remote Mob Programming
Back to Basics