Domain Specific Languages

Domain Specific Languages—Martin Fowler 2010-09-23 When carefully selected and used, Domain-Specific Languages (DSLs) may simplify complex code, promote effective communication with customers, improve productivity, and unplug development bottlenecks. In Domain-Specific Languages, noted software development expert Martin Fowler first provides the information software professionals need to decide if and when to utilize DSLs. Then, where DSLs prove suitable, Fowler presents effective techniques for building them, and guides software engineers in choosing the right approaches for their applications. This book’s techniques may be utilized with most modern object-oriented languages; the author provides numerous examples in Java and C#, as well as selected examples in Ruby. Wherever possible, chapters are organized to be self-standing, and most reference topics are presented in a familiar patterns format. Armed with this wide-ranging book, developers will have the knowledge they need to make important decisions about DSLs—and, where appropriate, gain the significant technical and business benefits they offer. The topics covered include: How DSLs compare to frameworks and libraries, and when those alternatives are sufficient Using parsers and parser generators, and parsing external DSLs Understanding, comparing, and choosing DSL language constructs Determining whether to use code generation, and comparing code generation strategies Previewing new language workbench tools for creating DSLs.

Domain-Specific Languages—Martin Fowler 2011 Martin Fowler’s breakthrough practitioner-oriented book on Domain Specific Languages—will do for DSLs what Fowler did for UML. Domain-Specific Languages is a highly anticipated introduction to DSLs, a category-defining book by one of the software world’s most influential authors. Two books in one: a concise narrative that introduces DSLs, and a larger reference that shows how to plan and develop them. *Helps software professionals reduce the cost and complexity of building DSLs—so they can take full advantage of them. Domain Specific Languages (DSLs) offer immense promise for software engineers who need better, faster ways to solve problems of specific types, or in specific areas or industries. DSLs have been around for several years, and have begun to grow in popularity. Now, Martin Fowler—one of the world’s most influential software engineering authors—has written the first practitioner-oriented book about them. Fowler’s legendary book, Refactoring, made software refactoring a crucial tool for software engineers worldwide; this book will do the same for DSLs. Fowler has designed Domain Specific Languages as two books in one. The first—a narrative designed to be read from “cover to cover” offers a concise introduction to DSLs, how they are implemented, and what are useful for. Next, Fowler thoroughly introduces today’s most effective techniques for building DSLs. Fowler covers both the “external” and “internal” DSLs, as well as alternative computational models, code generation, common parser topics, and much more. He provides extensive Java and C# examples for concepts that can best be explained using a dynamic language. Together, both sections enable readers to make well-informed choices about whether to use a DSL in their work, and which techniques to employ in order to build DSLs more quickly and cost-effectively.

Domain-Specific Languages—Martin Fowler 2010 When carefully selected and used, Domain-Specific Languages (DSLs) may simplify complex code, promote effective communication with customers, improve productivity, and unplug development bottlenecks. In Domain-Specific Languages, noted software development expert Martin Fowler first provides the information software professionals need to decide if and when to utilize DSLs. Then, where DSLs prove suitable, Fowler presents effective techniques for building them, and guides software engineers in choosing the right approaches for their applications. This book’s techniques may be utilized with most modern object-oriented languages; the author provides numerous examples in Java and C#, as well as selected examples in Ruby. Wherever possible, chapters are organized to be self-standing, and most reference topics are presented in a familiar patterns format. Armed with this wide-ranging book, developers will have the knowledge they need to make important decisions about DSLs—and, where appropriate, gain the significant technical and business benefits they offer. The topics covered include: How DSLs compare to frameworks and libraries, and when those alternatives are sufficient Using parsers and parser generators, and parsing external DSLs Understanding, comparing, and choosing DSL language constructs Determining whether to use code generation, and comparing code generation strategies Previewing new language workbench tools for creating DSLs.

DSL Engineering—Markus Voelter 2013 The definitive resource on domain-specific languages: based on years of real-world experience, relying on modern language workbenches and full of examples. Domain-Specific Languages are programming languages specialized for a particular application domain. By incorporating knowledge about that domain, DSLs can lead to more concise and more analyzable programs, better code quality and increased development speed. This book provides a thorough introduction to DSL, relying on today’s state of the art language workbenches. The book has four parts: introduction and discussion of their advantages and drawbacks. It also defines important terms and concepts and draws the case studies used in the most of the remainder of the book. Part II DSL Design: This part discusses the design of DSLs—dependent of implementation techniques. It reviews seven design dimensions, explains a number of reusable language paradigms and points out a number of process-related issues. Part III DSL Implementation: This part provides details about the implementation of DSLs with lots of code. It uses three state-of-the-art but quite different language workbenches: JetBrains MPS, Eclipse Xtext and TU Delft’s Spool. Part IV DSLs and Software Engineering: This part discusses the use of DSLs for requirements, architecture, implementation and product line engineering, as well as their roles as a developer utility and for implementing business logic. The book is available as a printed version (the one you are looking at) and as a PDF. For details see the book’s companion website at http://dslbook.org.

Language Implementation Patterns—Terence Parr 2009-12-31 Learn to build configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. You don’t need a background in computer science—ANTLR creator Terence Parr demystifies language implementation by breaking it down into the most common design patterns. Pattern by pattern, you’ll learn the key skills you need to implement your own computer languages. Knowing how to create domain-specific languages (DSLs) can give you a huge productivity boost. Instead of writing code in a general-purpose programming language, you can first build a custom language tailored to make you efficient in a particular domain. The key is understanding the common patterns found across language implementations. Language Design Patterns identifies and condenses the most common design patterns, providing sample implementations of each. The implementation patterns use Java, but the patterns themselves are completely general. Some of the implementations use well-known ANTLR parser generator, so readers will find this book an excellent source of ANTLR examples as well. But this book will benefit anyone interested in implementing languages, regardless of their tool of choice. Other language implementation books focus on compilers, which you rarely need in your daily life. Instead, Language Design Patterns shows you patterns you can use for all kinds of language applications. You’ll learn to create configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. Each chapter groups related design patterns and, in each pattern, you’ll get hands-on experience by building a complete sample implementation. By the time you finish the book, you’ll know how to solve some common language implementation problems.

to be on the business problem rather than the details of the programming platform. Domain Specific Languages—"little languages" implemented on top of conventional programming languages—give you a way to do this because they model the domain of your business problem. DSLs in Action introduces the concept of developer need-to-know foundational knowledge into the usage as well as implementation aspects of a DSL, focusing on the necessity of applications speaking the language of the domain. After reading this book, a programmer will be able to design APIs that make better domain models. For experienced developers, the book addresses the intricacies of domain language design without the pain of writing parsers by hand. The book discusses DSL usage and implementations in the real world based on a suite of JVM languages like Java, Ruby, Scala, and Groovy. It contains code snippets that implement high-level DSLs and discusses the pros and cons of each implementation. Purchase of the print book comes with an offer of a free PDF, ePub, and Kindle eBook from Manning. Also available is all code from the book. What's Inside Tested, real-world examples and exercises. Some exercises use the Eclipse platform to implement a DSL (or a programming language) together with Eclipse IDE tooling. It assumes that the user is familiar with Eclipse and its functionality. Existing basic knowledge of a compiler implementation would be useful, though not strictly required, since the book will explain all the stages of the development of a DSL.

NoSQL Distilled-Pramod J. Sadalage 2013 The need to handle increasingly larger data volumes is one factor driving the adoption of a new class of nonrelational "NoSQL" databases. Advocates of NoSQL databases claim they can be used to build systems that are more performant, scale better, and are easier to program. NoSQL Distilled is a concise but thorough introduction to this rapidly emerging technology. Pramod J. Sadalage and Martin Fowler explain how NoSQL databases work and the ways that they may be a superior alternative to a traditional RDBMS. The authors provide a fast-paced guide to the concepts you need to know in order to evaluate whether NoSQL databases are right for your needs and, if so, which technologies you should explore further. The first part of the book concentrates on core concepts, including schemaless data models, aggregates, new distribution models, the CAP theorem, and map-reduce. In the second part, the authors explore architectural and design issues associated with implementing NoSQL. They also present realistic use cases that demonstrate NoSQL databases at work and feature representative examples using Riak, MongoDB, Cassandra, and Neo4j. In addition, by drawing on Pramod Sadalage’s pioneering work, NoSQL Distilled shows how to implement evolution driven design with schema migration: an essential technique for applying NoSQL databases. The book concludes by describing how NoSQL is ushering in a new age of Polygot Persistence, where multiple data-storage worlds coexist, and architects can choose the technology best optimized for each type of data access. Refactoring 1999 Users can dramatically improve the design, performance, and manageability of object-oriented code without altering its interfaces or behavior. Refactoring shows users exactly how to spot the best opportunities for refactoring and exactly how to do it, step by step. DSLs in Boo-Ayende Rahien 2010 Provides information on creating DSLs for Microsoft .NET using Boo.

Fowler-Martin Fowler 2012-03-09 The practice of enterprise application development has been 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 architecture experienced programmers write. In Easy Java Programming: A First Course Application development, the author, noted object-oriented designer Martin Fowler, notices that despite changes in technology, the tools that face enterprise application developers. The author, noted object-oriented designer Martin Fowler, noticed that despite changes in technology, the tools that face enterprise application developers. The 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 divided into two parts. Each part uses a well-defined notation, such as the ones that exist for prescribing music, dance routines, or strategy in a football game. – The informal or intuitive meaning of the fundamental questions in understanding the interface between computers and the rest of the world. Why is it still hard to mechanize many tasks that seem to be computable? – The formal meaning is clear and mechanizable, as is, hopefully, the case for the instructions we give to our bank or to a merchant online.
The Definitive ANTLR 4 Reference-Terence Parr 2013-01-15 Programmers run into parsing problems all the time. Whether it's a data format like JSON, a network protocol like SMTP, a server configuration file for Apache, a PostScript/PDF file, or a simple spreadsheet macro language—ANTLR v4 and this book will demonstrate why ANTLR v4 has been such a language processing gem ever to build parse trees and generate code. This completely rewritten new edition of the bestselling Definitive ANTLR Reference shows you how to take advantage of these new features. Build your own languages with ANTLR v4, using ANTLR’s new advanced parsing technology. In this book, you’ll learn how ANTLR automatically builds a data structure representing the input (parse tree) and generates code that can walk the tree (visitor). You can use that combination to implement data readers, language interpreters, and translators. You’ll start by learning how to identify grammar patterns in language reference manuals and then slowly start building increasingly complex grammars. Next, you’ll build applications based upon those grammars by walking the automatically generated parse trees. Then you’ll tackle more advanced problems such as generating C code from context-free grammars and implementing complex symbol tables. You’ll learn the absolute control over parsing by embedding Java actions into the grammar. You’ll learn directly from well-known parsing expert Terence Parr, the ANTLR creator and project lead. You’ll master ANTLR grammar construction and learn how to build language tools using the built-in parse tree visitor mechanism. This book teaches real-world examples and shows you how to use ANTLR to build such things as a data file reader, a JSON to XML translator, an R parser, and a Java class->interface extractor. This book is your ticket to becoming a parsing guru! What You Need: ANTLR 4.0 and above. Java development tools. Ant build system (optional). A Java compiler. ANTLR 4 from source.

Scalac for the Impatient-Cay S. Horstmann 2012-03-08 Scala is a modern programming language for the Java Virtual machine (JVM) that combines the best features of object-oriented and functional programming languages. Using Scala, you can write programs more concisely than in Java, as well as leverage the full power of concurrency. Since Scala runs on the JVM, it can access any Java library and is interoperable with Java frameworks. Scala for the Impatient shows developers what Scala can do and how to do it. In this book, Cay Horstmann, the principal author of the international best-selling Core Java™, offers a rapid, code-based introduction that's completely practical. Horstmann introduces Scala concepts and techniques in "blog-sized" chunks that you can quickly master and apply. Hand-on activities guide you through well-defined stages of competency, from basic to expert. Coverage includes Getting started quickly with Scala’s interpreter, syntax, tools, and unique idioms Mastering core language features: functions, arrays, maps, tuples, packages, imports, exception handling, and more Becoming familiar with object-oriented programming in Scala: classes, inheritance, and traits Using Scala for real-world programming tasks: working with files, regular expressions, and XML Working with higher-order functions and the powerful Scala collections library Leveraging Scala’s powerful case classes Implementing domain specific languages in an imperative concurrence model Illustrating advanced Scala type system Applying advanced “power tools” such as annotations, implicits, and delimited continuations Scalac is rapidly reaching a tipping point that will reshape the experience of programming. This book will help object-oriented programmers build on their existing skills, allowing them to immediately construct useful applications as they gradually master advanced programming techniques.

Scala for the Impatient-Cay S. Horstmann 2012-03-08 Scala is a modern programming language for the Java Virtual machine (JVM) that combines the best features of object-oriented and functional programming languages. Using Scala, you can write programs more concisely than in Java, as well as leverage the full power of concurrency. Since Scala runs on the JVM, it can access any Java library and is interoperable with Java frameworks. Scala for the Impatient shows developers what Scala can do and how to do it. In this book, Cay Horstmann, the principal author of the international best-selling Core Java™, offers a rapid, code-based introduction that's completely practical. Horstmann introduces Scala concepts and techniques in "blog-sized" chunks that you can quickly master and apply. Hand-on activities guide you through well-defined stages of competency, from basic to expert. Coverage includes Getting started quickly with Scala’s interpreter, syntax, tools, and unique idioms Mastering core language features: functions, arrays, maps, tuples, packages, imports, exception handling, and more Becoming familiar with object-oriented programming in Scala: classes, inheritance, and traits Using Scala for real-world programming tasks: working with files, regular expressions, and XML Working with higher-order functions and the powerful Scala collections library Leveraging Scala’s powerful case classes Implementing domain specific languages in an imperative concurrence model Illustrating advanced Scala type system Applying advanced “power tools” such as annotations, implicits, and delimited continuations Scalac is rapidly reaching a tipping point that will reshape the experience of programming. This book will help object-oriented programmers build on their existing skills, allowing them to immediately construct useful applications as they gradually master advanced programming techniques.

Clean Code-Robert C. Martin 2009 Looks at the principles and clean code, includes case studies showcasing the practices of writing clean code, and contains a list of "refactoring rules" accumulated from the process of writing clean code. The book provides simple, practical advice that helps readers make the most of their programming time.

The Rime of the Ancient Mariner-Samuel Taylor Coleridge 1906

"execute the algorithm of getting up in the morning."
Refactoring—Jay Fields 2009-10-15 The Definitive Refactoring Guide, Fully Revamped for Ruby With refactoring, programmers can transform even the most chaotic software into well-designed systems that are far easier to evolve and maintain. What’s more, they can do it one step at a time, through a series of simple transformations, much as Martin Fowler’s classic book on the topic guided them. The techniques and idioms throughout—not code adapted from Java or any other environment. The authors introduce a detailed catalog of more than 70 proven Ruby refactoring techniques, with specific guidance on how to apply each of them, step-by-step instructions for using them, and example code illustrating how they work. Many of the authors’ refactoring techniques use powerful Ruby-specific features, and all code samples are available for download. Leveraging Fowler’s original concepts, the authors show how to perform refactoring in a controlled, efficient, incremental manner, so you methodically improve your code’s structure without introducing new bugs. Whatever your role in writing or maintaining Ruby code, this book will be an indispensable resource. This book will help you understand the core principles of refactoring, and it includes complete refactored code. Fowler introduces the Ruby version of his classic code refactoring examples at a time Build tests to make sure your refactoring work properly Understand the challenges of refactoring and how they can be overcome Compose methods to package code properly Move features between objects to take responsibilities where they fit best Organize statements to make your code more readable and maintainable. This book is the definitive guide to software refactoring for Ruby programmers.

Domain-Specific Languages: Martin Fowler 2004-09-15 Domain-Driven Design (DDD) is an approach to software development for complex businesses and organizations. It describes a set of techniques for identifying the key concepts of the domain as a foundation for creating a software system. The authors of this book, Martin Fowler and Rob C. Martin, provide a comprehensive guide to the principles and practices of domain-driven design. The book is divided into four parts: background and motivation; fundamentals; in-depth examples; and creating DSM solutions. There is an emphasis throughout the book on practical guidelines for implementing DSM, including how to identify the necessary language constructs, how to generate full models, and how to organize data to make it easier to work with. This book is a valuable resource for anyone working on complex software projects, providing a framework for building systems that are more robust and maintainable.

Domain-Specific Modeling: Steven Kelly 2008-04-11 "[The authors] are pioneers... Few in our industry have their breadth of knowledge and experience." —Entertainment Weekly "An ambitious novel with an engaging voice, a clever plot and some terrific writing." —The New York Times Book Review "No one beautiful, and blisteringly smart, We Were Liars is utterly unforgettable." —John Green, #1 New York Times bestselling author of The Fault in Our Stars "You’re the volume editors, the 18 carefully reviewed revised full papers presented are organized into topical sections on - surveys of domain-specific programming technologies - domain-specific programming languages - tool support for program generation - domain-specific techniques for program optimization. A Christmas Carol—Charles Dickens 2010-09-01 This Graphic Novel Series features classic tales retold with attractive color illustrations. Educators using the Dale-Chall vocabulary system adapted each title. Each 70 page, softcover book retains key phrases and quotations from the original classics. Introduce literature to reluctant readers and motivate struggling readers. Students build confidence through reading practice. Motivation makes all the difference. What’s more motivation then the success of? Fundamentals of Object-Oriented Design in UML—Charles F. Conway 2000 Fundamentals of Object-Oriented Design in UML shows aspiring and experienced programmers alike how to apply design concepts, the UML, and the best practices in OO development to improve both their code and their success rates with object-oriented projects.

Groovy for Domain-Specific Languages—Fergal Darrell 2010-06-01 Extend and enhance your Java applications with Domain Specific Languages in Groovy. Ruby Best Practices—Gregory Brown 2009-06-18 Provides information on designing APIs and domain-specific language, writing readable code, and working with functional programming ideas with the Ruby programming language.

We Were Liars Deluxe Edition—E. Lockhart 2017-05-23 The New York Times bestseller We Were Liars is now available as a not-to-be-missed hardcover deluxe edition! Whether you know it how it ends (shh... don’t tell!) or have let too many seasons go by without discovering the truth about the Liars for yourself, you will want to get your hands on the exclusive hardcover edition of the bestselling novel. The only thing that will make you want to read it is the exclusive content: exclusive photos from E. Lockhart’s upcoming novel Genuine Fraud—a psychological thriller that will leave you breathless Read it. And if anyone asks you how it ends, just LIE. Praise for We Were Liars: 20 Weeks on the New York Times Bestseller List “Haunting, sophisticated... a novel so twisty and well-told that it will appeal to older readers as well as to adolescents.” —The Wall Street Journal A rich, stunning summer mystery with a sharp twist that will leave you dying to talk about the book with a pal or two. — Parade.com “Thrilling, beautiful, and blisteringly smart, We Were Liars is utterly unforgettable.” — John Green, #1 New York Times bestselling author of The Fault in Our Stars “You’re going to want to remember the title. Liars details the summer of a girl who harbors a dark secret, and delivers a satisfying, but shocking twist ending. — Kirkus "A clever plot and some terrific writing.” — The New York Times Book Review “No one should be talking about the shocking twist ending. What we can talk about is...” [Lockhart’s] razor-sharp portrayal of a family for whom keeping up appearances is paramount and, ultimately, tragic.” —Chicago Tribune Model-Driven Development—Markus Völter 2013-06-26 Model-Driven Software Development (MDS) is currently a high-regarded development paradigm among developers and researchers. With the advent of GMAS and Microsoft’s Software Factories, the MDS approach has moved to the centre of the programmer’s attention, becoming the focus of conferences such as OOPSLA, JAOOn 2009. MDS is about using domain-specific languages to create models for describing the structure of software systems in an efficient and maintainable way. Models are subsequently transformed into executable code by a sequence of model transformations. This practical guide for software architect and developers is based on extensive practical and example case studies. International experts delivery: A comprehensive overview of MDS and how it relates to industriestandards such as MDA and Software Factories. Technical details on meta modeling, DSL construction, model-to-model and model-to-code transformations, and softwarearchitecture. Invaluable insight into the software development process, plus engineering issues such as versioning, testing and product lineengineering. Essential management knowledge concerning organizational topics such as support from some senior management and business architects. Architecture Patterns with Python—Harry Percival 2020-03-05 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 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 link 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 Domain-Specific Modeling—Steven Kelly 2008-04-11 “[The authors] are pioneers... Few in our industry have their breadth of knowledge and experience.” —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500-1000% in production for over a decade. DSM offers a development structure based on domain modeling, which has transformed into executable code. The authors of this book, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most 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 link 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 Domain-Specific Modeling—Steven Kelly 2008-04-11 “[The authors] are pioneers... Few in our industry have their breadth of knowledge and experience.” —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500-1000% in production for over a decade. DSM offers a development structure based on domain modeling, which has transformed into executable code. The authors of this book, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most 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 link 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 Domain-Specific Modeling—Steven Kelly 2008-04-11 “[The authors] are pioneers... Few in our industry have their breadth of knowledge and experience.” —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500-1000% in production for over a decade. DSM offers a development structure based on domain modeling, which has transformed into executable code. The authors of this book, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most 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 link 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 Domain-Specific Modeling—Steven Kelly 2008-04-11 “[The authors] are pioneers... Few in our industry have their breadth of knowledge and experience.” —From the Foreword by Dave Thomas, Bedarra Labs Domain-Specific Modeling (DSM) is the latest approach to software development, promising to greatly increase the speed and ease of software creation. Early adopters of DSM have been enjoying productivity increases of 500-1000% in production for over a decade. DSM offers a development structure based on domain modeling, which has transformed into executable code. The authors of this book, Harry Percival and Bob Gregory from MADE.com introduce proven architectural design patterns to help Python developers manage application complexity—and get the most 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 link 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
important refinements. Some of the patterns and definitions have been edited or rewritten by Evans to clarify the original intent. Three patterns have been added, describing concepts whose usefulness and importance has emerged in the intervening years. Also, the sequence and grouping of the topics has been changed significantly to better emphasize the core principles. This is an up-to-date, quick reference to DDD.

Yeah, reviewing a books domain specific languages by martin fowler could mount up your close associates listings. This is just one of the solutions for you to be successful. As understood, exploit does not recommend that you have extraordinary points.

Comprehending as without difficulty as deal even more than supplementary will have the funds for each success. bordering to, the publication as with ease as keenness of this domain specific languages by martin fowler can be taken as well as picked to act.