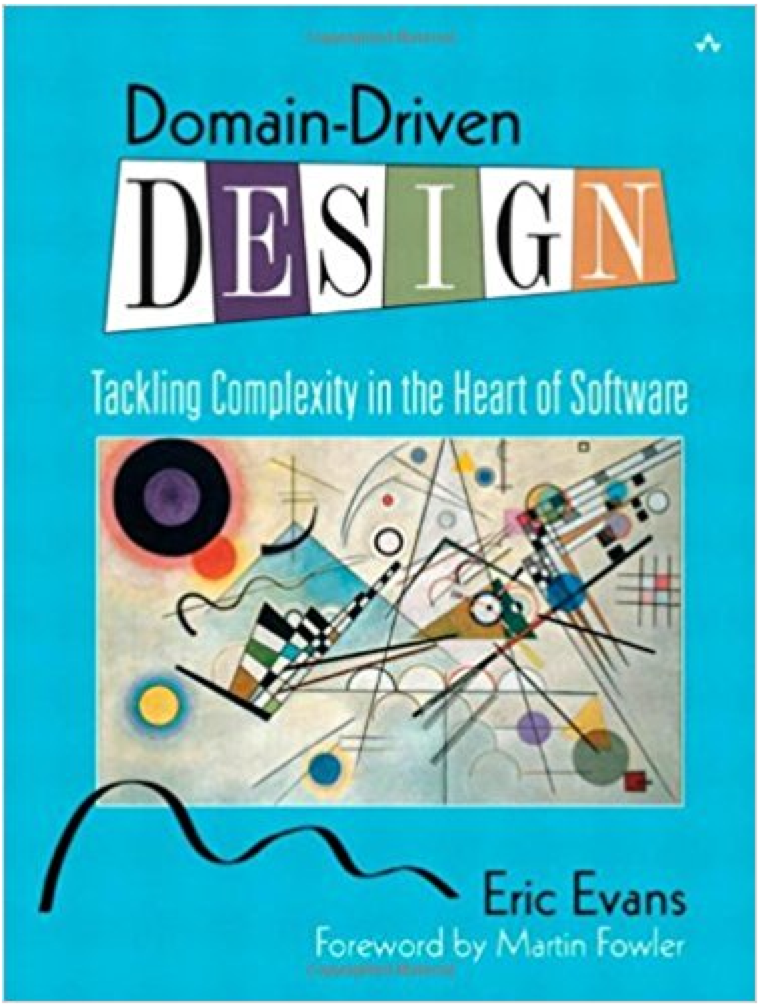
Domain-Driven Design: Tackling Complexity In The Heart Of Software


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Synopsis

Eric Evans has written a fantastic book on how you can make the design of your software match your mental model of the problem domain you are addressing His book is very compatible with XP It is not about drawing pictures of a domain it is about how you think of it the language you use to talk about it and how you organize your software to reflect your improving understanding of it Eric thinks that learning about your problem domain is as likely to happen at the end of your project as at the beginning and so refactoring is a big part of his technique The book is a fun read Eric has lots of interesting stories and he has a way with words I see this book as essential reading for software developers it is a future classic Ralph Johnson author of Design Patterns If you don t think you are getting value from your investment in object oriented programming this book will tell you what you ve forgotten to do Eric Evans convincingly argues for the importance of domain modeling as the central focus of development and provides a solid framework and set of techniques for accomplishing it This is timeless wisdom and will hold up long after the methodologies du jour have gone out of fashion Dave Collins author of Designing Object Oriented User Interfaces Eric weaves real world experience modeling and building business applications into a practical useful book Written from the perspective of a trusted practitioner Eric s descriptions of ubiquitous language the benefits of sharing models with users object life cycle management logical and physical application structuring and the process and results of deep refactoring are major contributions to our field Luke Hohmann author of Beyond Software Architecture This book belongs on the shelf of every thoughtful software developer Kent Beck What Eric has managed to capture is a part of the design process that experienced object designers have always used but that we have been

Book Information

Hardcover: 560 pages

Publisher: Addison-Wesley Professional; 1 edition (August 30, 2003)

Language: English

ISBN-10: 0321125215

ISBN-13: 978-0321125217

Product Dimensions: 7.2 x 1.3 x 9.3 inches

Shipping Weight: 2.8 pounds (View shipping rates and policies)

Average Customer Review: 4.4 out of 5 stars 102 customer reviews

Best Sellers Rank: #28,153 in Books (See Top 100 in Books) #14 inÂ Â Books > Textbooks > Computer Science > Object-Oriented Software Design #20 inÂ Â Books > Computers &

Technology > Computer Science > Systems Analysis & Design   #24 inÃ Â Books > Textbooks > Computer Science > Software Design & Engineering

Customer Reviews

Ã¢â ¬Å“Eric Evans has written a fantastic book on how you can make the design of your software match your mental model of the problem domain you are addressing. Ã¢â ¬Å“His book is very compatible with XP. It is not about drawing pictures of a domain; it is about how you think of it, the language you use to talk about it, and how you organize your software to reflect your improving understanding of it. Eric thinks that learning about your problem domain is as likely to happen at the end of your project as at the beginning, and so refactoring is a big part of his technique.
Ã¢â ¬Å“The book is a fun read. Eric has lots of interesting stories, and he has a way with words. I see this book as essential reading for software developersÃ¢â ¬â •it is a future classic.Ã¢â ¬Â•
Ã Â Ã Â Ã Â Ã Â Ã Â Ã¢â ¬â •Ralph Johnson, author of Design Patterns Ã¢â ¬Å“If you donÃ¢â ¬â„¢t think you are getting value from your investment in object-oriented programming, this book will tell you what youÃ¢â ¬â„¢ve forgotten to do. Ã¢â ¬Å“Eric Evans convincingly argues for the importance of domain modeling as the central focus of development and provides a solid framework and set of techniques for accomplishing it. This is timeless wisdom, and will hold up long after the methodologies du jour have gone out of fashion.Ã¢â ¬Â•
Ã Â Ã Â Ã Â Ã Â Ã Â Ã¢â ¬â •Dave Collins, author of Designing Object-Oriented User Interfaces
Ã¢â ¬Å“Eric weaves real-world experience modelingÃ¢â ¬â •and buildingÃ¢â ¬â •business applications into a practical, useful book. Written from the perspective of a trusted practitioner, EricÃ¢â ¬â„¢s descriptions of ubiquitous language, the benefits of sharing models with users, object life-cycle management, logical and physical application structuring, and the process and results of deep refactoring are major contributions to our field.Ã¢â ¬Â•
Ã Â Ã Â Ã Â Ã Â Ã Â Ã¢â ¬â •Luke Hohmann, author of Beyond Software Architecture   "This book belongs on the shelf of every thoughtful software developer." --Kent Beck "What Eric has managed to capture is a part of the design process that experienced object designers have always used, but that we have been singularly unsuccessful as a group in conveying to the rest of the industry. We've given away bits and pieces of this knowledge...but we've never organized and systematized the principles of building domain logic. This book is important." --Kyle Brown, author of Enterprise JavaÃ¢â Â¢ Programming with IBMÃ Â® WebSphereÃ Â®   The software development community widely acknowledges that domain modeling is central to software design. Through domain models, software developers are able to express rich functionality and translate it

into a software implementation that truly serves the needs of its users. But despite its obvious importance, there are few practical resources that explain how to incorporate effective domain modeling into the software development process.   Domain-Driven Design fills that need. This is not a book about specific technologies. It offers readers a systematic approach to domain-driven design, presenting an extensive set of design best practices, experience-based techniques, and fundamental principles that facilitate the development of software projects facing complex domains. Intertwining design and development practice, this book incorporates numerous examples based on actual projects to illustrate the application of domain-driven design to real-world software development. Readers learn how to use a domain model to make a complex development effort more focused and dynamic. A core of best practices and standard patterns provides a common language for the development team. A shift in emphasis--refactoring not just the code but the model underlying the code--in combination with the frequent iterations of Agile development leads to deeper insight into domains and enhanced communication between domain expert and programmer. Domain-Driven Design then builds on this foundation, and addresses modeling and design for complex systems and larger organizations.Specific topics covered include: Getting all team members to speak the same language Connecting model and implementation more deeply Sharpening key distinctions in a model Managing the lifecycle of a domain object Writing domain code that is safe to combine in elaborate ways Making complex code obvious and predictable Formulating a domain vision statement Distilling the core of a complex domain Digging out implicit concepts needed in the model Applying analysis patterns Relating design patterns to the model Maintaining model integrity in a large system Dealing with coexisting models on the same project Organizing systems with large-scale structures Recognizing and responding to modeling breakthroughs With this book in hand, object-oriented developers, system analysts, and designers will have the guidance they need to organize and focus their work, create rich and useful domain models, and leverage those models into quality, long-lasting software implementations.

Eric Evans is the founder of Domain Language, a consulting group dedicated to helping companies build evolving software deeply connected to their businesses. Since the 1980s, Eric has worked as a designer and programmer on large object-oriented systems in several complex business and technical domains. He has also trained and coached development teams in Extreme Programming.

IMHO required reading for enterprise level developers, architects and IT Project Managers. The principles are sound and the writing style actually makes what would otherwise be a dry read quite enjoyable.

Well, I’m still reading it, but it’s a mandatory reading for those who want to deliver great software

Challenging but worthwhile.

The reference on domain-driven design. A good book, but at times difficult to read and really get involved in. Takes a very high-level approach to design and doesn’t discuss some implementation details of going down the domain model path. It took me some time to get through this book.

This book gets longwinded and could have shed about 250 pages on the backend. However, I would recommend the first 200+ pages to every OO developer who is laboring with domain experts, stakeholders and users to match their requirements up with an object model. Like most development books I can’t recommend everything the author writes, but that said I think this book’s main message is very solid and offers plenty of good ideas to the OO developer writing common business applications.

If your process for writing an application is to start with the database, or to start designing UI, you should read this book.There is nothing new in this book -- but you really should read it.This book details the way many architects and analysts have been creating their applications for years, and for good reason. They start by creating domain classes that help represent data in a way that is understandable to both developer and customer. Then building out the application from there.Read this book.

I read this book in its draft form on a cross-country flight and was just blown away by it, enough so that I bought a bound version to make it easier to carry around and reread.I suppose what blew me away was that Evans crystallized and laid out quite clearly about a dozen ideas which were existing at the edge of my consciousness, but which I could not clearly verbalize.It fits quite nicely between the patterns books and the process books, but it’s not a cookbook and it’s not strictly a method. It’s a must read for the multitude of Java/C#/C++ developers who continue to write procedural code while claiming they’re OO developers because they’re using an OO language and they’ve read Design Patterns.

Incredibly smart aggregation of existing patterns and practices, along with Evans' own ideas to tie everything together. Dogmatic implementation of DDD on every project is not recommended, but understanding how all of the concepts in this book tie together and being able to pragmatically apply them will make for more successful projects.

Domain-Driven Design: Tackling Complexity in the Heart of Software Patterns, Principles, and Practices of Domain-Driven Design Software Engineering: The Current Practice (Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series) Simply Complexity: A Clear Guide to Complexity Theory What Customers Want: Using Outcome-Driven Innovation to Create Breakthrough Products and Services: Using Outcome-Driven Innovation to Create Breakthrough ... (Marketing/Sales/Advertising & Promotion) Cable-Driven Parallel Robots: Proceedings of the Third International Conference on Cable-Driven Parallel Robots (Mechanisms and Machine Science) The Software Requirements Memory Jogger: A Pocket Guide to Help Software And Business Teams Develop And Manage Requirements (Memory Jogger) Head First Software Development: A Learner's Companion to Software Development Agile Project Management: Agile Revolution, Beyond Software Limits: A Practical Guide to Implementing Agile Outside Software Development (Agile Business Leadership, Book 4) Don't Buy Software For Your Small Business Until You Read This Book: A guide to choosing the right software for your SME & achieving a rapid return on your investment Software Agreements Line by Line, 2nd ed.: A Detailed Look at Software Agreements and How to Draft Them to Meet Your Needs IEC 62304 Ed. 1.0 b:2006, Medical device software - Software life cycle processes Agile Software Development with Scrum (Series in Agile Software Development) The Secret Lives of Hoarders: True Stories of Tackling Extreme Clutter The World Food Problem: Tackling the Causes of Undernutrition in the Third World Born on Third Base: A One Percenter Makes the Case for Tackling Inequality, Bringing Wealth Home, and Committing to the Common Good Tackling Life Head on: Lessons for Kids' Lives With Ronnie Lott As "Coach Tackling Tough Choices: Discussion-Starting Skits for Teens (Acting It Out Series) Hawaii Big Island Unanchor Travel Guide - Tackling 10 Must-Dos on the Big Island in 3 Days Graphic Design Success: Over 100 Tips for Beginners in Graphic Design: Graphic Design Basics for Beginners, Save Time and Jump Start Your Success (graphic ... graphic design beginner, design skills)