

- [Table of Contents](#)

Domain-Driven Design: Tackling Complexity in the Heart of Software

By [Eric Evans](#)

[Start Reading ▶](#)

Publisher: Addison Wesley

Pub Date: August 20, 2003

ISBN: 0-321-12521-5

Pages: 560

The software development community widely acknowledges that domain modeling is central to software design. Through domain modeling, software developers are able to express rich functionality and translate that functionality into software implementation that truly serves the needs of its users. Despite its obvious importance, however, there are few practical resources that show how to incorporate effective domain modeling into the software development process.

*Domain-Driven Design* fills that need. 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, Domain-Driven Design incorporates numerous examples in Java-case studies taken from actual projects that illustrate the application of domain-driven design to real-world software development.

Readers will find an overview of domain-driven design that highlights key principles, terms, and implications. The book presents a core of best practices and standard patterns that provide a common language for the development team. In addition, it highlights how refactoring in domain modeling, integrated with the frequent iterations of Agile development, leads to deeper insight into domains and enhanced communication between domain expert and programmer. Building on this foundation, the book then addresses domain-driven design for complex systems and larger organizations.

Specific topics covered include:

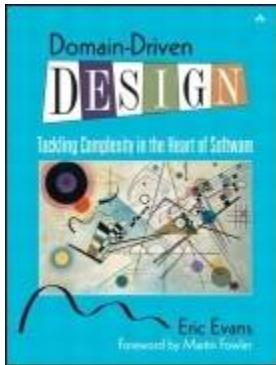
- Isolating the domain
- Entities, value objects, services, and modules
- The lifecycle of a domain object
- 
-

- Representing processes as domain objects
- Creating functions free of side effects
- Conceptual contours
- Standalone classes
- Extending specifications
- Applying analysis patterns
- Relating design patterns to the model
- Maintaining model integrity
- Formulating the domain vision statement
- Choosing refactoring targets
- Responsibility layers
- Creating a pluggable component framework
- Bringing together large-scale structures and bounded contexts

With this book in hand, object-oriented developers, system analysts, and designers will have the guidance they need to think deeply about domains, create rich and accurate domain models, and transform these models into high-quality, long-lasting software implementations.

[\[ Team LiB \]](#)

[NEXT ▶](#)



- [Table of Contents](#)

Domain-Driven Design: Tackling Complexity in the Heart of Software

By [Eric Evans](#)

[Start Reading ▶](#)

Publisher: Addison Wesley

Pub Date: August 20, 2003

ISBN: 0-321-12521-5

Pages: 560

[Copyright](#)

[Praise for \*Domain-Driven Design\*](#)

[Foreword](#)

[Preface](#)

[Contrasting Three Projects](#)

[The Challenge of Complexity](#)

[Design Versus Development Process](#)

[The Structure of This Book](#)

[Who Should Read This Book](#)

[A Domain-Driven Team](#)

[Acknowledgments](#)

[Part I: Putting the Domain Model to Work](#)

[Chapter One. Crunching Knowledge](#)

[Ingredients of Effective Modeling](#)

[Knowledge Crunching](#)

[Continuous Learning](#)

[Knowledge-Rich Design](#)

[Deep Models](#)

[Chapter Two. Communication and the Use of Language](#)

[Ubiquitous Language](#)

[Modeling Out Loud](#)

[One Team, One Language](#)

[Documents and Diagrams](#)

[Explanatory Models](#)

[Chapter Three. Binding Model and Implementation](#)

[Model-Driven Design](#)

[Modeling Paradigms and Tool Support](#)

Letting the Bones Show: Why Models Matter to Users

Hands-On Modelers

## Part II: The Building Blocks of a Model-Driven Design

### Chapter Four. Isolating the Domain

Layered Architecture

The Domain Layer Is Where the Model Lives

The Smart UI "Anti-Pattern"

Other Kinds of Isolation

### Chapter Five. A Model Expressed in Software

Associations

Entities (a.k.a. Reference Objects)

Value Objects

Services

Modules (a.k.a. Packages)

Modeling Paradigms

### Chapter Six. The Life Cycle of a Domain Object

Aggregates

Factories

Repositories

Designing Objects for Relational Databases

### Chapter Seven. Using the Language: An Extended Example

Introducing the Cargo Shipping System

Isolating the Domain: Introducing the Applications

Distinguishing ENTITIES and VALUE Objects

Designing Associations in the Shipping Domain

AGGREGATE Boundaries

Selecting REPOSITORIES

Walking Through Scenarios

Object Creation

Pause for Refactoring: An Alternative Design of the Cargo AGGREGATE

MODULES in the Shipping Model

Introducing a New Feature: Allocation Checking

A Final Look

## Part III: Refactoring Toward Deeper Insight

### Chapter Eight. Breakthrough

Story of a Breakthrough

Opportunities

Focus on Basics

Epilogue: A Cascade of New Insights

### Chapter Nine. Making Implicit Concepts Explicit

Digging Out Concepts

How to Model Less Obvious Kinds of Concepts

### Chapter Ten. Supple Design

Intention-Revealing Interfaces

Side-Effect-Free Functions

Assertions

Conceptual Contours

Standalone Classes

Closure of Operations

Declarative Design