DDD Community

IMPLEMENTING DOMAIN-DRIVEN DESIGN

VAUGHN VERNON

FOREWORD BY ERIC EVANS

Praise for *Implementing Domain-Driven Design*

"With *Implementing Domain-Driven Design*, Vaughn has made an important contribution not only to the literature of the Domain-Driven Design community, but also to the literature of the broader enterprise application architecture field. In key chapters on Architecture and Repositories, for example, Vaughn shows how DDD fits with the expanding array of architecture styles and persistence technologies for enterprise applications—including SOA and REST, NoSQL and data grids—that has emerged in the decade since Eric Evans' seminal book was first published. And, fittingly, Vaughn illuminates the blocking and tackling of DDD—the implementation of entities, value objects, aggregates, services, events, factories, and repositories—with plentiful examples and valuable insights drawn from decades of practical experience. In a word, I would describe this book as *thorough*. For software developers of all experience levels looking to improve their results, and design and implement domain-driven enterprise applications consistently with the best current state of professional practice, *Implementing Domain-Driven Design* will impart a treasure trove of knowledge hard won within the DDD and enterprise application architecture communities over the last couple decades."

—Randy Stafford, Architect At-Large, Oracle Coherence Product Development

"Domain-Driven Design is a powerful set of thinking tools that can have a profound impact on how effective a team can be at building software-intensive systems. The thing is that many developers got lost at times when applying these thinking tools and really needed more concrete guidance. In this book, Vaughn provides the missing links between theory and practice. In addition to shedding light on many of the misunderstood elements of DDD, Vaughn also connects new concepts like Command/Query Responsibility Segregation and Event Sourcing that many advanced DDD practitioners have used with great success. This book is a must-read for anybody looking to put DDD into practice."

—Udi Dahan, Founder of NServiceBus

"For years, developers struggling to practice Domain-Driven Design have been wishing for more practical help in actually implementing DDD. Vaughn did an excellent job in closing the gap between theory and practice with a complete implementation reference. He paints a vivid picture of what it is like to do DDD in a contemporary project, and provides plenty of practical advice on how to approach and solve typical challenges occurring in a project life cycle."

—Alberto Brandolini, DDD Instructor, Certified by Eric Evans and
Domain Language, Inc.

"*Implementing Domain-Driven Design* does a remarkable thing: it takes a sophisticated and substantial topic area in DDD and presents it clearly, with nuance, fun and finesse. This book is written in an engaging and friendly style, like a trusted advisor giving you expert counsel on how to accomplish what is most important. By the time you finish the book you will be able to begin applying all the important concepts of

Implementing Domain-Driven Design

Table of Contents

Contents

Foreword

Preface

Acknowledgments

About the Author

Guide to This Book

Chapter 1 Getting Started with DDD

- Can I DDD?
- Why You Should Do DDD
- How to Do DDD
- The Business Value of Using DDD
  1. The Organization Gains a Useful Model of Its Domain
  2. A Refined, Precise Definition and Understanding of the Business Is Developed
  3. Domain Experts Contribute to Software Design
  4. A Better User Experience Is Gained
  5. Clean Boundaries Are Placed around Pure Models
  6. Enterprise Architecture Is Better Organized
  7. Agile, Iterative, Continuous Modeling Is Used
  8. New Tools, Both Strategic and Tactical, Are Employed
- The Challenges of Applying DDD
- Fiction, with Bucketfuls of Reality
- Wrap-Up

Pearson

Table of Contents

Chapter 2 Domains, Subdomains, and Bounded Contexts
- Big Picture
  - Subdomains and Bounded Contexts at Work
  - Focus on the Core Domain
- Why Strategic Design Is So Incredibly Essential
- Real-World Domains and Subdomains
- Making Sense of Bounded Contexts
  - Room for More than the Model
  - Size of Bounded Contexts
  - Aligning with Technical Components
- Sample Contexts
  - Collaboration Context
  - Identity and Access Context
  - Agile Project Management Context
- Wrap-Up

Chapter 3 Context Maps
- Why Context Maps Are So Essential
  - Drawing Context Maps
  - Projects and Organizational Relationships
  - Mapping the Three Contexts
- Wrap-Up

Chapter 4 Architecture
- Interviewing the Successful CIO
- Layers
  - Dependency Inversion Principle
- Hexagonal or Ports and Adapters
- Service-Oriented

Pearson

Table of Contents

Representational State TransferREST

REST as an Architectural Style

Key Aspects of a RESTful HTTP Server

Key Aspects of a RESTful HTTP Client

REST and DDD

Why REST?

Command-Query Responsibility Segregation, or CQRS

Examining Areas of CQRS

Dealing with an Eventually Consistent Query Model

Event-Driven Architecture

Pipes and Filters

Long-Running Processes, aka Sagas

Event Sourcing

Data Fabric and Grid-Based Distributed Computing

Data Replication

Event-Driven Fabrics and Domain Events

Continuous Queries

Distributed Processing

Wrap-Up

Chapter 5 Entities

Why We Use Entities

Unique Identity

User Provides Identity

Application Generates Identity

Persistence Mechanism Generates Identity

Another Bounded Context Assigns Identity

When the Timing of Identity Generation Matters

Surrogate Identity

Pearson

Table of Contents

Identity Stability

Discovering Entities and Their Intrinsic Characteristics

Uncovering Entities and Properties

Digging for Essential Behavior

Roles and Responsibilities

Construction

Validation

Change Tracking

Wrap-Up

Chapter 6 Value Objects

Value Characteristics

Measures, Quantifies, or Describes

Immutable

Conceptual Whole

Replaceability

Value Equality

Side-Effect-Free Behavior

Integrate with Minimalism

Standard Types Expressed as Values

Testing Value Objects

Implementation

Persisting Value Objects

Reject Undue Influence of Data Model Leakage

ORM and Single Value Objects

ORM and Many Values Serialized into a Single Column

ORM and Many Values Backed by a Database Entity

ORM and Many Values Backed by a Join Table

ORM and Enum-as-State Objects

Table of Contents

Wrap-Up

Chapter 7 Services

What a Domain Service Is (but First, What It Is Not)

Make Sure You Need a Service

Modeling a Service in the Domain

Is Separated Interface a Necessity?

A Calculation Process

Transformation Services

Using a Mini-Layer of Domain Services

Testing Services

Wrap-Up

Chapter 8 Domain Events

The When and Why of Domain Events

Modeling Events

With Aggregate Characteristics

Identity

Publishing Events from the Domain Model

Publisher

Subscribers

Spreading the News to Remote Bounded Contexts

Messaging Infrastructure Consistency

Autonomous Services and Systems

Latency Tolerances

Event Store

Architectural Styles for Forwarding Stored Events

Publishing Notifications as RESTful Resources

Publishing Notifications through Messaging Middleware

Table of Contents

Implementation

Publishing the NotificationLog

Publishing Message-Based Notifications

Wrap-Up

Chapter 9 Modules

Designing with Modules

Basic Module Naming Conventions

Module Naming Conventions for the Model

Modules of the Agile Project Management Context

Modules in Other Layers

Module before Bounded Context

Wrap-Up

Chapter 10 Aggregates

Using Aggregates in the Scrum Core Domain

First Attempt: Large-Cluster Aggregate

Second Attempt: Multiple Aggregates

Rule: Model True Invariants in Consistency Boundaries

Rule: Design Small Aggregates

Dont Trust Every Use Case

Rule: Reference Other Aggregates by Identity

Making Aggregates Work Together through Identity References

Model Navigation

Scalability and Distribution

Rule: Use Eventual Consistency Outside the Boundary

Ask Whose Job It Is

Reasons to Break the Rules

Reason One: User Interface Convenience

Pearson

Table of Contents

Reason Two: Lack of Technical Mechanisms

Reason Three: Global Transactions

Reason Four: Query Performance

Adhering to the Rules

Gaining Insight through Discovery

Rethinking the Design, Again

Estimating Aggregate Cost

Common Usage Scenarios

Memory Consumption

Exploring Another Alternative Design

Implementing Eventual Consistency

Is It the Team Members Job?

Time for Decisions

Implementation

Create a Root Entity with Unique Identity

Favor Value Object Parts

Using Law of Demeter and Tell, Dont Ask

Optimistic Concurrency

Avoid Dependency Injection

Wrap-Up

Chapter 11 Factories

Factories in the Domain Model

Factory Method on Aggregate Root

Creating CalendarEntry Instances

Creating Discussion Instances

Factory on Service

Wrap-Up

Chapter 12 Repositories

Table of Contents

Collection-Oriented Repositories

Hibernate Implementation

Considerations for a TopLink Implementation

Persistence-Oriented Repositories

Coherence Implementation

MongoDB Implementation

Additional Behavior

Managing Transactions

A Warning

Type Hierarchies

Repository versus Data Access Object

Testing Repositories

Testing with In-Memory Implementations

Wrap-Up

Chapter 13 Integrating Bounded Contexts

Integration Basics

Distributed Systems Are Fundamentally Different

Exchanging Information across System Boundaries

Integration Using RESTful Resources

Implementing the RESTful Resource

Implementing the REST Client Using an Anticorruption Layer

Integration Using Messaging

Staying Informed about Product Owners and Team Members

Can You Handle the Responsibility?

Long-Running Processes, and Avoiding Responsibility

Process State Machines and Time-out Trackers

Designing a More Sophisticated Process

When Messaging or Your System Is Unavailable

Table of Contents

Wrap-Up

Chapter 14 Application

User Interface

Rendering Domain Objects

Render Data Transfer Object from Aggregate Instances

Use a Mediator to Publish Aggregate Internal State

Render Aggregate Instances from a Domain Payload Object

State Representations of Aggregate Instances

Use Case Optimal Repository Queries

Dealing with Multiple, Disparate Clients

Rendition Adapters and Handling User Edits

Application Services

Sample Application Service

Decoupled Service Output

Composing Multiple Bounded Contexts

Infrastructure

Enterprise Component Containers

Wrap-Up

Appendix A: Aggregates and Event Sourcing: A+ES

Inside an Application Service

Command Handlers

Lambda Syntax

Concurrency Control

Structural Freedom with A+ES

Performance

Implementing an Event Store

Relational Persistence

Pearson

Table of Contents

- BLOB Persistence
- Focused Aggregates
- Read Model Projections
- Use with Aggregate Design
- Events Enrichment
- Supporting Tools and Patterns
  - Event Serializers
  - Event Immutability
  - Value Objects
- Contract Generation
- Unit Testing and Specifications
- Event Sourcing in Functional Languages

Bibliography

Index

Pearson