

Join the discussion @ p2p.wrox.com



Wrox Programmer to Programmer™

Patterns, Principles, and Practices of Domain-Driven Design

Scott Millett with Nick Tune

PATTERNS, PRINCIPLES, AND PRACTICES OF DOMAIN-DRIVEN DESIGN

INTRODUCTION	XXXV
--------------------	------

► PART I THE PRINCIPLES AND PRACTICES OF DOMAIN-DRIVEN DESIGN

CHAPTER 1	What Is Domain-Driven Design?	3
CHAPTER 2	Distilling the Problem Domain	15
CHAPTER 3	Focusing on the Core Domain	31
CHAPTER 4	Model-Driven Design	41
CHAPTER 5	Domain Model Implementation Patterns	59
CHAPTER 6	Maintaining the Integrity of Domain Models with Bounded Contexts	73
CHAPTER 7	Context Mapping	91
CHAPTER 8	Application Architecture	105
CHAPTER 9	Common Problems for Teams Starting Out with Domain-Driven Design	121
CHAPTER 10	Applying the Principles, Practices, and Patterns of DDD	131

► PART II STRATEGIC PATTERNS: COMMUNICATING BETWEEN BOUNDED CONTEXTS

CHAPTER 11	Introduction to Bounded Context Integration	151
CHAPTER 12	Integrating via Messaging	181
CHAPTER 13	Integrating via HTTP with RPC and REST	245

► PART III TACTICAL PATTERNS: CREATING EFFECTIVE DOMAIN MODELS

CHAPTER 14	Introducing the Domain Modeling Building Blocks	309
CHAPTER 15	Value Objects	329
CHAPTER 16	Entities	361

Continues

CHAPTER 17 Domain Services 389

CHAPTER 18 Domain Events 405

CHAPTER 19 Aggregates 427

CHAPTER 20 Factories 469

CHAPTER 21 Repositories 479

CHAPTER 22 Event Sourcing 595

► **PART IV DESIGN PATTERNS FOR EFFECTIVE APPLICATIONS**

CHAPTER 23 Architecting Application User Interfaces. 645

CHAPTER 24 CQRS: An Architecture of a Bounded Context 669

CHAPTER 25 Commands: Application Service Patterns for
Processing Business Use Cases 687

CHAPTER 26 Queries: Domain Reporting 713

INDEX 737

Patterns, Principles, and Practices of Domain-Driven Design

Patterns, Principles, and Practices of Domain-Driven Design

Scott Millett

Nick Tune



Patterns, Principles, and Practices of Domain-Driven Design

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-118-71470-6
ISBN: 978-1-118-71465-2 (ebk)
ISBN: 978-1-118-71469-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2014951018

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

For my darling buds, Primrose and Albert.

—SCOTT MILLETT

ABOUT THE AUTHOR

SCOTT MILLETT is the Director of IT for Iglu.com and has been working with .NET since version 1.0. He was awarded the ASP.NET MVP in 2010 and 2011. He is also the author of *Professional ASP.NET Design Patterns* and *Professional Enterprise .NET*. If you would like to contact Scott about DDD or working at Iglu, feel free to write to him at scott@elbandit.co.uk, by giving him a tweet [@ScottMilleTT](https://twitter.com/ScottMilleTT), or becoming friends via <https://www.linkedin.com/in/scottmilleTT>.

ABOUT THE CONTRIBUTING AUTHOR

NICK TUNE is passionate about solving business problems, building ambitious products, and constantly learning. Being a software developer really is his dream job. His career highlight so far was working at 7digital, where he was part of self-organizing, business-focused teams that deployed to production up to 25 times per day. His future ambitions are to work on exciting new products, with passionate people, and continually become a more complete problem solver.

You can learn more about Nick and his views on software development, software delivery, and his favorite technologies on his website (www.ntcoding.co.uk) and Twitter ([@ntcoding](https://twitter.com/ntcoding)).

ABOUT THE TECHNICAL EDITOR

ANTONY DENYER works as a developer, consultant, and coach and has been developing software professionally since 2004. He has worked on various projects that have effectively used DDD concepts and practices. More recently, he has been advocating the use of CQRS and REST in the majority of his projects. You can reach him via e-mail at antonydenyer.co.uk, and he tweets from [@tonydenyer](https://twitter.com/tonydenyer).

CREDITS

PROJECT EDITOR
Rosemarie Graham

TECHNICAL EDITOR
Antony Denyer

PRODUCTION EDITOR
Christine O'Connor

COPY EDITOR
Karen Gill

**MANAGER OF CONTENT DEVELOPMENT
AND ASSEMBLY**
Mary Beth Wakefield

MARKETING DIRECTOR
David Mayhew

MARKETING MANAGER
Carrie Sherrill

**PROFESSIONAL TECHNOLOGY &
STRATEGY DIRECTOR**
Barry Pruett

BUSINESS MANAGER
Amy Knies

ASSOCIATE PUBLISHER
Jim Minatel

PROJECT COORDINATOR, COVER
Brent Savage

PROOFREADER
Jenn Bennett, Word One

INDEXER
Johnna VanHoose Dinse

COVER DESIGNER
Wiley

COVER IMAGE
[@iStockphoto.com/andynwt](https://www.iStockphoto.com/andynwt)

ACKNOWLEDGMENTS

FIRSTLY I WOULD LIKE to give a massive thanks to Nick Tune for agreeing to help me out with this project and contributing greatly to many of the chapters. I would also like to thank Rosemarie Graham, Jim Minatel, and all those at Wrox who have helped to create this book. Thanks as well to Antony Denyer who did a sterling job as the technical editor. Lastly, many thanks to Isabel Mack for the grammar pointers and early feedback of the Leanpub draft.

CONTENTS

INTRODUCTION

xxxv

PART I: THE PRINCIPLES AND PRACTICES OF DOMAIN-DRIVEN DESIGN

CHAPTER 1: WHAT IS DOMAIN-DRIVEN DESIGN?	3
The Challenges of Creating Software for Complex Problem Domains	4
Code Created Without a Common Language	4
A Lack of Organization	5
The Ball of Mud Pattern Stifles Development	5
A Lack of Focus on the Problem Domain	6
How the Patterns of Domain-Driven Design Manage Complexity	6
The Strategic Patterns of DDD	6
Distilling the Problem Domain to Reveal What Is Important	7
Creating a Model to Solve Domain Problems	7
Using a Shared Language to Enable Modeling Collaboration	7
Isolate Models from Ambiguity and Corruption	8
Understanding the Relationships between Contexts	9
The Tactical Patterns of DDD	9
The Problem Space and the Solution Space	9
The Practices and Principles of Domain-Driven Design	11
Focusing on the Core Domain	11
Learning through Collaboration	11
Creating Models through Exploration and Experimentation	11
Communication	11
Understanding the Applicability of a Model	12
Constantly Evolving the Model	12
Popular Misconceptions of Domain-Driven Design	12
Tactical Patterns Are Key to DDD	12
DDD Is a Framework	13
DDD Is a Silver Bullet	13
The Salient Points	13

CHAPTER 2: DISTILLING THE PROBLEM DOMAIN	15
Knowledge Crunching and Collaboration	15
Reaching a Shared Understanding through a Shared Language	16
The Importance of Domain Knowledge	17
The Role of Business Analysts	17
An Ongoing Process	17
Gaining Domain Insight with Domain Experts	18
Domain Experts vs Stakeholders	18
Deeper Understanding for the Business	19
Engaging with Your Domain Experts	19
Patterns for Effective Knowledge Crunching	19
Focus on the Most Interesting Conversations	19
Start from the Use Cases	20
Ask Powerful Questions	20
Sketching	20
Class Responsibility Collaboration Cards	21
Defer the Naming of Concepts in Your Model	21
Behavior-Driven Development	22
Rapid Prototyping	23
Look at Paper-Based Systems	24
Look For Existing Models	24
Understanding Intent	24
Event Storming	25
Impact Mapping	25
Understanding the Business Model	27
Deliberate Discovery	28
Model Exploration Whirlpool	29
The Salient Points	29
 CHAPTER 3: FOCUSING ON THE CORE DOMAIN	 31
Why Decompose a Problem Domain?	31
How to Capture the Essence of the Problem	32
Look Beyond Requirements	32
Capture the Domain Vision for a Shared Understanding of What Is Core	32
How to Focus on the Core Problem	33
Distilling a Problem Domain	34
Core Domains	35

Treat Your Core Domain as a Product Rather than a Project	36
Generic Domains	37
Supporting Domains	37
How Subdomains Shape a Solution	37
Not All Parts of a System will be Well Designed	37
Focus on Clean Boundaries over Perfect Models	38
The Core Domain Doesn't Always Have to Be Perfect the First Time	39
Build Subdomains for Replacement Rather than Reuse	39
What if You Have no Core Domain?	39
The Salient Points	40
CHAPTER 4: MODEL-DRIVEN DESIGN	41
What Is a Domain Model?	42
The Domain versus the Domain Model	42
The Analysis Model	43
The Code Model	43
The Code Model Is the Primary Expression of the Domain Model	44
Model-Driven Design	44
The Challenges with Upfront Design	44
Team Modeling	45
Using a Ubiquitous Language to Bind the Analysis to the Code Model	47
A Language Will Outlive Your Software	47
The Language of the Business	48
Translation between the Developers and the Business	48
Collaborating on a Ubiquitous Language	48
Carving Out a Language by Working with Concrete Examples	49
Teach Your Domain Experts to Focus on the Problem and Not Jump to a Solution	50
Best Practices for Shaping the Language	51
How to Create Effective Domain Models	52
Don't Let the Truth Get in the Way of a Good Model	52
Model Only What Is Relevant	54
Domain Models Are Temporarily Useful	54
Be Explicit with Terminology	54
Limit Your Abstractions	54
Focus Your Code at the Right Level of Abstraction	55
Abstract Behavior Not Implementations	55

Implement the Model in Code Early and Often	56
Don't Stop at the First Good Idea	56
When to Apply Model-Driven Design	56
If It's Not Worth the Effort Don't Try and Model It	56
Focus on the Core Domain	57
The Salient Points	57
 CHAPTER 5: DOMAIN MODEL IMPLEMENTATION PATTERNS	 59
The Domain Layer	60
Domain Model Implementation Patterns	60
Domain Model	62
Transaction Script	65
Table Module	67
Active Record	67
Anemic Domain Model	67
Anemic Domain Model and Functional Programming	68
The Salient Points	71
 CHAPTER 6: MAINTAINING THE INTEGRITY OF DOMAIN MODELS WITH BOUNDED CONTEXTS	 73
The Challenges of a Single Model	74
A Model Can Grow in Complexity	74
Multiple Teams Working on a Single Model	74
Ambiguity in the Language of the Model	75
The Applicability of a Domain Concept	76
Integration with Legacy Code or Third Party Code	78
Your Domain Model Is not Your Enterprise Model	79
Use Bounded Contexts to Divide and Conquer a Large Model	79
Defining a Model's Boundary	82
Define Boundaries around Language	82
Align to Business Capabilities	83
Create Contexts around Teams	83
Try to Retain Some Communication between Teams	84
Context Game	85
The Difference between a Subdomain and a Bounded Context	85
Implementing Bounded Contexts	85
The Salient Points	89

CHAPTER 7: CONTEXT MAPPING	91
A Reality Map	92
The Technical Reality	92
The Organizational Reality	93
Mapping a Relevant Reality	94
X Marks the Spot of the Core Domain	94
Recognising the Relationships between Bounded Contexts	95
Anticorruption Layer	95
Shared Kernel	96
Open Host Service	97
Separate Ways	97
Partnership	98
An Upstream/Downstream Relationship	98
Customer-Supplier	99
Conformist	100
Communicating the Context Map	100
The Strategic Importance of Context Maps	101
Retaining Integrity	101
The Basis for a Plan of Attack	101
Understanding Ownership and Responsibility	101
Revealing Areas of Confusion in Business Work Flow	102
Identifying Nontechnical Obstacles	102
Encourages Good Communication	102
Helps On-Board New Starters	102
The Salient Points	103
 CHAPTER 8: APPLICATION ARCHITECTURE	 105
Application Architecture	105
Separating the Concerns of Your Application	106
Abstraction from the Complexities of the Domain	106
A Layered Architecture	106
Dependency Inversion	107
The Domain Layer	107
The Application Service Layer	108
The Infrastructural Layers	108
Communication Across Layers	108
Testing in Isolation	109
Don't Share Data Schema between Bounded Contexts	109

Application Architectures versus Architectures for Bounded Contexts	111
Application Services	112
Application Logic versus Domain Logic	114
Defining and Exposing Capabilities	114
Business Use Case Coordination	115
Application Services Represent Use Cases, Not Create, Read, Update, and Delete	115
Domain Layer As an Implementation Detail	115
Domain Reporting	116
Read Models versus Transactional Models	116
Application Clients	117
The Salient Points	120
 CHAPTER 9: COMMON PROBLEMS FOR TEAMS STARTING OUT WITH DOMAIN-DRIVEN DESIGN	 121
Overemphasizing the Importance of Tactical Patterns	122
Using the Same Architecture for All Bounded Contexts	122
Striving for Tactical Pattern Perfection	122
Mistaking the Building Blocks for the Value of DDD	123
Focusing on Code Rather Than the Principles of DDD	123
Missing the Real Value of DDD: Collaboration, Communication, and Context	124
Producing a Big Ball of Mud Due to Underestimating the Importance of Context	124
Causing Ambiguity and Misinterpretations by Failing to Create a UL	125
Designing Technical-Focused Solutions Due to a Lack of Collaboration	125
Spending Too Much Time on What's Not Important	126
Making Simple Problems Complex	126
Applying DDD Principles to a Trivial Domain with Little Business Expectation	126
Disregarding CRUD as an Antipattern	127
Using the Domain Model Pattern for Every Bounded Context	127
Ask Yourself: Is It Worth This Extra Complexity?	127
Underestimating the Cost of Applying DDD	127
Trying to Succeed Without a Motivated and Focused Team	128
Attempting Collaboration When a Domain Expert Is Not Behind the Project	128
Learning in a Noniterative Development Methodology	128

Applying DDD to Every Problem	129
Sacrificing Pragmatism for Needless Purity	129
Wasted Effort by Seeking Validation	129
Always Striving for Beautiful Code	130
DDD Is About Providing Value	130
The Salient Points	130
CHAPTER 10: APPLYING THE PRINCIPLES, PRACTICES, AND PATTERNS OF DDD	131
Selling DDD	132
Educating Your Team	132
Speaking to Your Business	132
Applying the Principles of DDD	133
Understand the Vision	133
Capture the Required Behaviors	134
Distilling the Problem Space	134
Focus on What Is Important	134
Understand the Reality of the Landscape	135
Modeling a Solution	135
All Problems Are Not Created Equal	136
Engaging with an Expert	136
Select a Behavior and Model Around a Concrete Scenario	137
Collaborate with the Domain Expert on the Most Interesting Parts	137
Evolve UL to Remove Ambiguity	138
Throw Away Your First Model, and Your Second	138
Implement the Model in Code	139
Creating a Domain Model	139
Keep the Solution Simple and Your Code Boring	139
Carve Out an Area of Safety	140
Integrate the Model Early and Often	140
Nontechnical Refactoring	140
Decompose Your Solution Space	140
Rinse and Repeat	141
Exploration and Experimentation	142
Challenge Your Assumptions	142
Modeling Is a Continuous Activity	142
There Are No Wrong Models	142
Supple Code Aids Discovery	143
Making the Implicit Explicit	143

Tackling Ambiguity	144
Give Things a Name	145
A Problem Solver First, A Technologist Second	146
Don't Solve All the Problems	146
How Do I Know That I Am Doing It Right?	146
Good Is Good Enough	147
Practice, Practice, Practice	147
The Salient Points	147

PART II: STRATEGIC PATTERNS: COMMUNICATING BETWEEN BOUNDED CONTEXTS

CHAPTER 11: INTRODUCTION TO BOUNDED CONTEXT INTEGRATION	151
How to Integrate Bounded Contexts	152
Bounded Contexts Are Autonomous	153
The Challenges of Integrating Bounded Contexts at the Code Level	153
Multiple Bounded Contexts Exist within a Solution	153
Namespaces or Projects to Keep Bounded Contexts Separate	154
Integrating via the Database	155
Multiple Teams Working in a Single Codebase	156
Models Blur	156
Use Physical Boundaries to Enforce Clean Models	157
Integrating with Legacy Systems	158
Bubble Context	158
Autonomous Bubble Context	158
Exposing Legacy Systems as Services	160
Integrating Distributed Bounded Contexts	161
Integration Strategies for Distributed Bounded Contexts	161
Database Integration	162
Flat File Integration	163
RPC	164
Messaging	165
REST	165
The Challenges of DDD with Distributed Systems	165
The Problem with RPC	166
RPC Is Harder to Make Resilient	167
RPC Costs More to Scale	167
RPC Involves Tight Coupling	168

Distributed Transactions Hurt Scalability and Reliability	169
Bounded Contexts Don't Have to Be Consistent with Each Other	169
Eventual Consistency	169
Event-Driven Reactive DDD	170
Demonstrating the Resilience and Scalability of Reactive Solutions	171
Challenges and Trade-Offs of Asynchronous Messaging	173
Is RPC Still Relevant?	173
SOA and Reactive DDD	174
View Your Bounded Contexts as SOA Services	175
Decompose Bounded Contexts into Business Components	175
Decompose Business Components into Components	176
Going Even Further with Micro Service Architecture	178
The Salient Points	180
 CHAPTER 12: INTEGRATING VIA MESSAGING	 181
Messaging Fundamentals	182
Message Bus	182
Reliable Messaging	184
Store-and-Forward	184
Commands and Events	185
Eventual Consistency	186
Building an E-Commerce Application with NServiceBus	186
Designing the System	187
Domain-Driven Design	187
Containers Diagrams	188
Evolutionary Architecture	191
Sending Commands from a Web Application	192
Creating a Web Application to Send Messages with NServiceBus	192
Sending Commands	197
Handling Commands and Publishing Events	200
Creating an NServiceBus Server to Handle Commands	200
Configuring the Solution for Testing and Debugging	201
Publishing Events	204
Subscribing to Events	206
Making External HTTP Calls Reliable with Messaging Gateways	208
Messaging Gateways Improve Fault Tolerance	208
Implementing a Messaging Gateway	209
Controlling Message Retries	212
Eventual Consistency in Practice	215
Dealing with Inconsistency	215

Rolling Forward into New States	215
Bounded Contexts Store All the Data They Need Locally	216
Storage Is Cheap—Keep a Local Copy	217
Common Data Duplication Concerns	223
Pulling It All Together in the UI	224
Business Components Need Their Own APIs	225
Be Wary of Server-Side Orchestration	226
UI Composition with AJAX Data	226
UI Composition with AJAX HTML	226
Sharing Your APIs with the Outside World	227
Maintaining a Messaging Application	227
Message Versioning	228
Backward-Compatible Message Versioning	228
Handling Versioning with NServiceBus’s Polymorphic Handlers	229
Monitoring and Scaling	233
Monitoring Errors	233
Monitoring SLAs	234
Scaling Out	235
Integrating a Bounded Context with Mass Transit	235
Messaging Bridge	236
Mass Transit	236
Installing and Configuring Mass Transit	236
Declaring Messages for Use by Mass Transit	238
Creating a Message Handler	239
Subscribing to Events	239
Linking the Systems with a Messaging Bridge	240
Publishing Events	242
Testing It Out	243
Where to Learn More about Mass Transit	243
The Salient Points	243
CHAPTER 13: INTEGRATING VIA HTTP WITH RPC AND REST	245
Why Prefer HTTP?	247
No Platform Coupling	247
Everyone Understands HTTP	247
Lots of Mature Tooling and Libraries	247
Dogfooding Your APIs	247
RPC	248
Implementing RPC over HTTP	248
SOAP	249

Plain XML or JSON: The Modern Approach to RPC	259
Choosing a Flavor of RPC	263
REST	264
Demystifying REST	264
Resources	264
Hypermedia	265
Statelessness	265
REST Fully Embraces HTTP	266
What REST Is Not	267
REST for Bounded Context Integration	268
Designing for REST	268
Building Event-Driven REST Systems with ASP.NET Web API	273
Maintaining REST Applications	303
Versioning	303
Monitoring and Metrics	303
Drawbacks with REST for Bounded Context Integration	304
Less Fault Tolerance Out of the Box	304
Eventual Consistency	304
The Salient Points	305

PART III: TACTICAL PATTERNS: CREATING EFFECTIVE DOMAIN MODELS

CHAPTER 14: INTRODUCING THE DOMAIN MODELING BUILDING BLOCKS	309
Tactical Patterns	310
Patterns to Model Your Domain	310
Entities	310
Value Objects	314
Domain Services	317
Modules	318
Lifecycle Patterns	318
Aggregates	318
Factories	322
Repositories	323
Emerging Patterns	324
Domain Events	324
Event Sourcing	326
The Salient Points	327

CHAPTER 15: VALUE OBJECTS	329
When to Use a Value Object	330
Representing a Descriptive, Identity-Less Concept	330
Enhancing Explicitness	331
Defining Characteristics	333
Identity-Less	333
Attribute-Based Equality	333
Behavior-Rich	337
Cohesive	337
Immutable	337
Combinable	339
Self-Validating	341
Testable	344
Common Modeling Patterns	345
Static Factory Methods	345
Micro Types (Also Known as Tiny Types)	347
Collection Aversion	349
Persistence	351
NoSQL	352
SQL	353
Flat Denormalization	353
Normalizing into Separate Tables	357
The Salient Points	359
CHAPTER 16: ENTITIES	361
Understanding Entities	362
Domain Concepts with Identity and Continuity	362
Context-Dependent	363
Implementing Entities	363
Assigning Identifiers	363
Natural Keys	363
Arbitrarily Generated IDs	364
Datastore-Generated IDs	368
Pushing Behavior into Value Objects and Domain Services	369
Validating and Enforcing Invariants	371
Focusing on Behavior, Not Data	374
Avoiding the “Model the Real-World” Fallacy	377
Designing for Distribution	378
Common Entity Modeling Principles and Patterns	380

Implementing Validation and Invariants with Specifications	380
Avoid the State Pattern; Use Explicit Modeling	382
Avoiding Getters and Setters with the Memento Pattern	385
Favor Hidden-Side-Effect-Free Functions	386
The Salient Points	388
CHAPTER 17: DOMAIN SERVICES	389
Understanding Domain Services	390
When to Use a Domain Service	390
Encapsulating Business Policies and Processes	390
Representing Contracts	394
Anatomy of a Domain Service	395
Avoiding Anemic Domain Models	395
Contrasting with Application Services	396
Utilizing Domain Services	397
In the Service Layer	397
In the Domain	398
Manually Wiring Up	399
Using Dependency Injection	400
Using a Service Locator	400
Applying Double Dispatch	401
Decoupling with Domain Events	402
Should Entities Even Know About Domain Services?	403
The Salient Points	403
CHAPTER 18: DOMAIN EVENTS	405
Essence of the Domain Events Pattern	406
Important Domain Occurrences That Have Already Happened	406
Reacting to Events	407
Optional Asynchrony	407
Internal vs External Events	408
Event Handling Actions	409
Invoke Domain Logic	409
Invoke Application Logic	410
Domain Events' Implementation Patterns	410
Use the .Net Framework's Events Model	410
Use an In-Memory Bus	412
Udi Dahan's Static DomainEvents Class	415
Handling Threading Issues	417

Avoid a Static Class by Using Method Injection	418
Return Domain Events	419
Use an IoC Container as an Event Dispatcher	421
Testing Domain Events	422
Unit Testing	422
Application Service Layer Testing	424
The Salient Points	425

CHAPTER 19: AGGREGATES	427
-------------------------------	------------

Managing Complex Object Graphs	428
Favoring a Single Traversal Direction	428
Qualifying Associations	430
Preferring IDs Over Object References	431
Aggregates	434
Design Around Domain Invariants	435
Higher Level of Domain Abstraction	435
Consistency Boundaries	435
Transactional Consistency Internally	436
Eventual Consistency Externally	439
Special Cases	440
Favor Smaller Aggregates	441
Large Aggregates Can Degrade Performance	441
Large Aggregates Are More Susceptible to Concurrency Conflicts	442
Large Aggregates May Not Scale Well	442
Defining Aggregate Boundaries	442
eBidder: The Online Auction Case Study	443
Aligning with Invariants	444
Aligning with Transactions and Consistency	446
Ignoring User Interface Influences	448
Avoiding Dumb Collections and Containers	448
Don't Focus on HAS-A Relationships	449
Refactoring to Aggregates	449
Satisfying Business Use Cases—Not Real Life	449
Implementing Aggregates	450
Selecting an Aggregate Root	450
Exposing Behavioral Interfaces	452
Protecting Internal State	453
Allowing Only Roots to Have Global Identity	454
Referencing Other Aggregates	454