Domain-Driven Design

TACKLING COMPLEXITY IN THE HEART OF SOFTWARE

Eric Evans

✦Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco New York • Toronto • Montreal • London • Munich • Paris • Madrid Capetown • Sydney • Tokyo • Singapore • Mexico City Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

See page 517 for photo credits.

The publisher offers discounts on this book when ordered in quantity for bulk purchases and special sales. For more information, please contact:

U.S. Corporate and Government Sales (800) 382-3419 corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact:

International Sales international@pearsoned.com

Visit Addison-Wesley on the Web: www.awprofessional.com

Library of Congress Cataloging-in-Publication Data

Evans, Eric, 1962-

Domain-driven design : tackling complexity in the heart of software / Eric Evans.

p. cm. Includes bibliographical references and index.

ISBN 0-321-12521-5

1. Computer software—Development. 2. Object-oriented programming (Computer science) I. Title.

QA76.76.D47E82 2003 005.1—dc21

2003050331

Copyright © 2004 by Eric Evans

All rights reserved. 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, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

For information on obtaining permission for use of material from this work, please submit a written request to:

Pearson Education, Inc. Rights and Contracts Department 500 Boylston Street, Suite 900 Boston, MA 02116 Fax: (617) 671-3447

ISBN 0-321-12521-5 Text printed in the United States on recycled paper at Courier in Westford, Massachusetts. Twenty-First printing, July 2015

C O N T E N T S

Foreword Preface Acknowledgments	xvii xix xxix
Part I	
Putting the Domain Model to Work	1
Chapter 1: Crunching Knowledge	7
Ingredients of Effective Modeling	12
Knowledge Crunching	13
Continuous Learning	15
Knowledge-Rich Design	17
Deep Models	20
Chapter 2: Communication and the Use of Language	23
Ubiquitous Language	24
Modeling Out Loud	30
One Team, One Language	32
Documents and Diagrams	35
Written Design Documents	37
Executable Bedrock	40
Explanatory Models	41
Chapter 3: Binding Model and Implementation	45
Model-Driven Design	47
Modeling Paradigms and Tool Support	50
Letting the Bones Show: Why Models Matter to Users	57
Hands-On Modelers	60

Part II	
The Building Blocks of a Model-Driven Design	63
Chapter 4: Isolating the Domain	67
Layered Architecture	68
Relating the Layers	72
Architectural Frameworks	74
The Domain Layer Is Where the Model Lives	75
The Smart UI "Anti-Pattern"	76
Other Kinds of Isolation	79
Chapter 5: A Model Expressed in Software	81
Associations	82
Entities (a.k.a. Reference Objects)	89
Modeling ENTITIES	93
Designing the Identity Operation	94
VALUE OBJECTS	97
Designing VALUE OBJECTS	99
Designing Associations That Involve VALUE OBJECTS	102
Services	104
Services and the Isolated Domain Layer	106
Granularity	108
Access to Services	108
Modules (a.k.a. Packages)	109
Agile Modules	111
The Pitfalls of Infrastructure-Driven Packaging	112
Modeling Paradigms	116
Why the Object Paradigm Predominates	116
Nonobjects in an Object World	119
Sticking with Model-Driven Design When	
Mixing Paradigms	120
Chapter 6: The Life Cycle of a Domain Object	123
Aggregates	125
Factories	136
Choosing FACTORIES and Their Sites	139
When a Constructor Is All You Need	141
Designing the Interface	143

Where Does Invariant Logic Go?	144
Entity Factories Versus Value Object Factories	144
Reconstituting Stored Objects	145
Repositories	147
Querying a Repository	152
Client Code Ignores REPOSITORY Implementation;	
Developers Do Not	154
Implementing a REPOSITORY	155
Working Within Your Frameworks	156
The Relationship with FACTORIES	157
Designing Objects for Relational Databases	159
Chapter 7: Using the Language: An Extended Example	163
Introducing the Cargo Shipping System	163
Isolating the Domain: Introducing the Applications	166
Distinguishing ENTITIES and VALUE OBJECTS	167
Role and Other Attributes	168
Designing Associations in the Shipping Domain	169
Aggregate Boundaries	170
Selecting REPOSITORIES	172
Walking Through Scenarios	173
Sample Application Feature: Changing the Destination	
of a Cargo	173
Sample Application Feature: Repeat Business	173
Object Creation	174
FACTORIES and Constructors for Cargo	174
Adding a Handling Event	175
Pause for Refactoring: An Alternative Design of the	
Cargo Aggregate	177
MODULES in the Shipping Model	179
Introducing a New Feature: Allocation Checking	181
Connecting the Two Systems	182
Enhancing the Model: Segmenting the Business	183
Performance Tuning	185
A Final Look	186

Relactoring Toward Deeper Insight	10
Chapter 8: Breakthrough	19
Story of a Breakthrough	19
A Decent Model, and Yet	19
The Breakthrough	19
A Deeper Model	1
A Sobering Decision	1
The Payoff	2
Opportunities	2
Focus on Basics	2
Epilogue: A Cascade of New Insights	2
Chapter 9: Making Implicit Concepts Explicit	2
Digging Out Concepts	2
Listen to Language	2
Scrutinize Awkwardness	2
Contemplate Contradictions	2
<i>Read the Book</i>	2
Try, Try Again	2
How to Model Less Obvious Kinds of Concepts	2
Explicit Constraints	2
Processes as Domain Objects	2
Specification	2
Applying and Implementing Specification	2
Chapter 10: Supple Design	2
INTENTION-REVEALING INTERFACES	2
Side-Effect-Free Functions	2
Assertions	2
Conceptual Contours	2
Standalone Classes	2
Closure of Operations	2
Declarative Design	2
Domain-Specific Languages	2
A Declarative Style of Design	2
Extending SPECIFICATIONS in a Declarative Style	2
Angles of Attack	2

Carve Off Subdomains	283
Draw on Established Formalisms, When You Can	283
Chapter 11: Applying Analysis Patterns	293
Chapter 12: Relating Design Patterns to the Model	309
Strategy (a.k.a. Policy)	311
Composite	315
Why Not Flyweight?	320
Chapter 13: Refactoring Toward Deeper Insight	321
Initiation	321
Exploration Teams	322
Prior Art	323
A Design for Developers	324
Timing	324
Crisis as Opportunity	325
Part IV	
Strategic Design	327
Chapter 14: Maintaining Model Integrity	331
Bounded Context	335
Recognizing Splinters Within a BOUNDED CONTEXT	339
Continuous Integration	341
Context Map	344
Testing at the CONTEXT Boundaries	351
Organizing and Documenting Context MAPs	351
Relationships Between BOUNDED CONTEXTS	352
Shared Kernel	354
Customer/Supplier Development Teams	356
Conformist	361
ANTICORRUPTION LAYER	364
Designing the Interface of the ANTICORRUPTION LAYER	366
Implementing the ANTICORRUPTION LAYER	366
A Cautionary Tale	370
Separate Ways	371
Open Host Service	374
Published Language	375

Unifying an Elephant	378
Choosing Your Model Context Strategy	381
Team Decision or Higher	382
Putting Ourselves in Context	382
Transforming Boundaries	382
Accepting That Which We Cannot Change: Delineating	
the External Systems	383
Relationships with the External Systems	384
The System Under Design	385
Catering to Special Needs with Distinct Models	386
Deployment	387
The Trade-off	388
When Your Project Is Already Under Way	388
Transformations	389
Merging Contexts: Separate Ways → Shared Kernel	389
Merging Contexts: Shared Kernel \rightarrow Continuous	
Integration	391
Phasing Out a Legacy System	393
Open Host Service \rightarrow Published Language	394
Chapter 15: Distillation	397
Core Domain	400
Choosing the CORE	402
Who Does the Work?	403
An Escalation of Distillations	404
Generic Subdomains	406
Generic Doesn't Mean Reusable	412
Project Risk Management	413
Domain Vision Statement	415
Highlighted Core	417
The Distillation Document	418
The Flagged Core	419
The Distillation Document as Process Tool	420
Cohesive Mechanisms	422
Generic Subdomain Versus Cohesive Mechanism	424
When a Mechanism Is Part of the Core Domain	425
Distilling to a Declarative Style	426
	100

The Costs of Creating a SEGREGATED CORE	429
Evolving Team Decision	430
Abstract Core	435
Deep Models Distill	436
Choosing Refactoring Targets	437
Chapter 16: Large-Scale Structure	439
Evolving Order	444
System Metaphor	447
The "Naive Metaphor" and Why We Don't Need It	448
Responsibility Layers	450
Choosing Appropriate Layers	460
Knowledge Level	465
Pluggable Component Framework	475
How Restrictive Should a Structure Be?	480
Refactoring Toward a Fitting Structure	481
Minimalism	481
Communication and Self-Discipline	482
Restructuring Yields Supple Design	482
Distillation Lightens the Load	483
Chapter 17: Bringing the Strategy Together	485
Combining Large-Scale Structures and BOUNDED CONTEXTS	485
Combining Large-Scale Structures and Distillation	488
Assessment First	490
Who Sets the Strategy?	490
Emergent Structure from Application Development	491
A Customer-Focused Architecture Team	492
Six Essentials for Strategic Design Decision Making	492
The Same Goes for the Technical Frameworks	495
Beware the Master Plan	496
Conclusion	499
Appendix: The Use of Patterns in This Book	507
Glossary	511
References	515
Photo Credits	517
Index	519