Building Object Applications That Work
Managing Object Technology Series

Charles F. Bowman
Series Editor
and
President
SoftWright Solutions, Suffern, New York

1. What Every Software Manager Must Know to Succeed with Object Technology, John D. Williams

2. Managing Your Move to Object Technology: Guidelines and Strategies for a Smooth Transition, Barry McGibbon


4. Getting Results with the Object-Oriented Enterprise Model, Thornton Gale and James Eldred

5. Deploying Distributed Business Software, Ted Lewis, Ph.D.

6. The Blueprint for Business Objects, Peter Ringar

7. Object Technology Strategies and Tactics, Gilbert Singer

8. Upgrading Relational Databases with Objects, Robert Vermeulen


10. CUC96: Component-Based Software Engineering, Collected and Introduced by Thomas Jell

11. Developing Business Objects, Edited by Andy Carmichael

12. Component-Based Development for Enterprise Systems: Applying the SELECT Perspective™, Paul Allen and Stuart Frost

Additional Volumes in Preparation
Building Object Applications That Work

Your Step-by-Step Handbook for Developing Robust Systems with Object Technology

Scott W. Ambler
To

Rick Berman, Jeri Taylor, and Michael Piller

Because I really, really, really want a walk-on part on
Star Trek Voyager
Contents

About the Author, xix
Foreword, xxi
Preface, xxiii
Acknowledgments, xxix

Part I
INTRODUCTION TO BUILDING OBJECT APPLICATIONS

Chapter 1 Where We've Been Before—Object-Oriented Concepts and Techniques, 3

1.1 Object-Oriented Concepts—A Really Quick Recap, 4
  1.1.1 Objects and Classes, 5
  1.1.2 Attributes and Methods, 6
  1.1.3 Abstraction, Encapsulation, and Information Hiding, 6
## Contents

1.1.4 Inheritance, 8
1.1.5 Persistence, 9
1.1.6 Instance Relationships, 10
1.1.7 Aggregation, 11
1.1.8 Collaboration, 12
1.1.9 Coupling and Cohesion, 12
1.1.10 Polymorphism, 13

1.2 Object-Oriented Analysis and Design Techniques—A Really Quick Recap, 14
1.2.1 CRC Modeling, 14
1.2.2 Use-Case Scenario Testing, 20
1.2.3 Class Diagramming, 25

1.3 Iterative Object-Oriented Development—The Pinball System Development Life Cycle (SDLC), 27
1.3.1 The Pinball Metaphor, 28
1.3.2 The Steps of the Pinball SDLC, 30

1.4 What We Missed in The Object Primer, 33
1.4.1 The Difference Between Class and Instance Attributes, 33
1.4.2 The Difference Between Class and Instance Methods, 35
1.4.3 The Implications of Class Attributes and Methods, 37
1.4.4 What We Mean When We Say “Attribute” or “Method,” 40
1.4.5 The Ambler Class Diagram Notation v2.0, 40
1.4.6 The Unified Modeling Language Class-Diagram Notation (Simplified), 41

1.5 Just in Case You’ve Forgotten—The Bank Case Study, 42
1.5.1 The ABC Case Study, 42

1.6 Where We’re Going in This Book, 45
Contents

ix

Part II

OBJECT-ORIENTED ANALYSIS, DESIGN, AND ARCHITECTURE

Chapter 2  Bubbles and Lines—Useful Diagrams for Object-Oriented Analysis and Design, 49

2.1  The Importance of Bubbles and Lines, 50
  2.1.1  Making Your Diagrams Look Good, 50

2.2  Other People's Notations—Understanding Class Diagrams, 54

2.3  Use-Case Diagrams—Understanding Use Cases, 57
  2.3.1  Drawing Use-Case Diagrams, 58
  2.3.2  Why and When Should We Create Use-Case Diagrams?, 59

2.4  Sequence Diagrams—Understanding Use Cases Even Better, 60
  2.4.1  How to Draw Sequence Diagrams, 62
  2.4.2  Why and When Should We Draw Sequence Diagrams?, 63

2.5  State Diagrams—Understanding Complicated Classes, 63
  2.5.1  How to Draw a State Diagram, 66
  2.5.2  Taking State Diagramming a Bit Further, 67
  2.5.3  When and Why Should We Draw State Diagrams?, 68

2.6  Collaboration Diagrams, 69
  2.6.1  Drawing Collaboration Diagrams, 69
  2.6.2  Why and When Should We Draw Collaboration Diagrams?, 70

2.7  Process Models—Understanding What's Currently Going On, 70
  2.7.1  Drawing Process Models, 72
  2.7.2  Why and When Should We Draw Process Models?, 74
Contents

2.8 Interface-Flow Diagrams—Understanding the User Interface, 74
  2.8.1 Drawing Interface-Flow Diagrams, 75
  2.8.2 Why and When Should We Draw Interface-Flow Diagrams?, 76

2.9 Data Models—Modeling Your Database Design, 77
  2.9.1 Drawing Data Models, 78
  2.9.2 Why and When Should We Draw Data Models?, 79

2.10 An Overall Analysis and Design Strategy, 79

2.11 What We’ve Learned, 81

Chapter 3  Improving Your Design—A Class-Type Architecture, 85

3.1 Why Do We Need a Class-Type Architecture?, 87

3.2 Building Up to a Class-Type Architecture, 87
  3.2.1 A Two-Layer Class-Type Architecture, 88
  3.2.2 A Three-Layer Class-Type Architecture, 90
  3.2.3 A Four-Layer Class Architecture, 93
  3.2.4 A Five-Layer Class-Type Architecture, 97

3.3 A Detailed Look at the Various Class Types, 98
  3.3.1 Interface Classes, 98
  3.3.2 Business Classes, 101
  3.3.3 Persistence Classes, 102
  3.3.4 System Classes, 105

3.4 Issues with the Five-Layer Class-Type Architecture, 106
  3.4.1 To Which Layer Does This Class Belong?, 106
  3.4.2 Can I Buy Some of These Classes?, 107
  3.4.3 What Are the Implications for Project Management?, 109
  3.4.4 What Are the Implications for Development?, 110
  3.4.5 Is This Enough?, 113
Contents

3.4.6 What Are the Advantages and Disadvantages?, 114

3.5 What We’ve Learned, 114

Chapter 4 Reusing Your Development Efforts—Object-Oriented Patterns, 117

4.1 Design Patterns, 119
4.1.1 Singleton, 119
4.1.2 Proxy, 120
4.1.3 Roles Played (State), 122

4.2 Analysis Patterns, 125
4.2.1 Item-Item Description, 126
4.2.2 Business Entity, 128
4.2.3 Contact Point, 128
4.2.4 Shipping/Billing, 130
4.2.5 Place, 131

4.3 How to Use Patterns Effectively, 132

4.4 How to Discover New Patterns, 133

4.5 The Advantages and Disadvantages of Patterns, 134
4.5.1 Advantages, 134
4.5.2 Disadvantages, 135

4.6 What We’ve Learned, 136

Chapter 5 Development in the 90s and Beyond—Designing Distributed Object-Oriented Applications, 139

5.1 Centralized Mainframes, 141

5.2 Client/Server, 142
5.2.1 Two-Tier Client/Server, 144
5.2.2 Three-Tier Client/Server, 146
Contents

5.3 Distributed Classes, 148
   5.3.1 "Traditional" Client/Server Using Object Technology, 149
   5.3.2 Taking a Peer-to-Peer Approach—Object-Oriented
       Client/Server, 149
   5.3.3 Applets, 160

5.4 Distributed Objects, 161
   5.4.1 Common Object Request Broker Architecture, 163

5.5 What We've Learned, 167

Part III

OBJECT-ORIENTED CONSTRUCTION

Chapter 6 Measuring and Improving the Quality of Your
Work—Object-Oriented Metrics, 171

6.1 What Are Metrics and What Can They Do for/to Me?, 172

6.2 Estimating Projects, 173
   6.2.1 Number of Key Classes, 174
   6.2.2 Person Days per Class, 174
   6.2.3 Classes per Developer, 175
   6.2.4 Number of Reused Classes, 176
   6.2.5 Function and Feature Points, 177

6.3 Improving Your Development Efforts, 178
   6.3.1 Analysis Metrics, 179
   6.3.2 Design Metrics, 180
   6.3.3 Construction Metrics, 185
   6.3.4 Testing Metrics, 189

6.4 Choosing the Right Tools, 190
   6.4.1 Comments per Method and Percentage of
       Commented Methods, 190
Contents

6.5 Improving Your Development Approach, 191
6.6 Metrics Success Factors, 191
6.7 What We’ve Learned, 194

Chapter 7 Choosing an Object-Oriented Language—Comparing the Leading Languages, 197
7.1 What to Look For in an OO Language, 198
7.2 The Leading OO Languages, 202
  7.2.1 C++, 202
  7.2.2 Java, 203
  7.2.3 ObjectCOBOL, 205
  7.2.4 Smalltalk, 207
  7.2.5 Comparing the Leading OO Languages, 207
7.3 Understanding Electronic Commerce on the Internet, 207
  7.3.1 An Architecture for Supporting Electronic Commerce, 209
  7.3.2 Payment Processing on the Internet, 210
  7.3.3 It’s the WORLD Wide Web—International Issues, 211
  7.3.4 You Can Sell Both Physical and Virtual Products Internationally, 212
  7.3.5 Don’t Forget Taxes, 212
  7.3.6 Using Smalltalk/ObjectCOBOL and Java to Develop Electronic Commerce Applications, 213
7.4 Beyond Programming Languages—Other Development Tools, 216
7.5 What We’ve Learned, 218

Chapter 8 Building Your Application—Effective Object-Oriented Construction Techniques, 221
8.1 Attitude Is Everything, 222
  8.1.1 There’s More to Development Than Just Coding, 222
Contents

8.1.2 There's More to Development Than Just Development, 223
8.1.3 You Need to Get the Design Right First, 224
8.1.4 You Need to Develop in Small Steps, 225
8.1.5 You Need to Work Closely with Your Users, 226

8.2 Working with Attributes Effectively, 226
  8.2.1 Naming Attributes, 226
  8.2.2 Accessor Methods, 229

8.3 Writing High-Quality Methods, 237
  8.3.1 Naming Methods, 237
  8.3.2 Documenting Methods, 237
  8.3.3 Paragraphing Your Code, 241
  8.3.4 Methods Should Always Do Something, 242
  8.3.5 Methods Should Do One Thing Only, 243
  8.3.6 Do One Thing per Line, 244
  8.3.7 The 30-Second Rule, 245
  8.3.8 Specify Order of Message Sends, 245
  8.3.9 Polymorphism the Right Way—Avoiding Case Statements, 246

8.4 Creating Maintainable Classes, 247
  8.4.1 The Law of Demeter, 247

8.5 Programming Techniques and Approaches, 249
  8.5.1 Implementing Instance Relationships, 249
  8.5.2 Error Handling, 251
  8.5.3 Callback Methods and Message Dispatchers, 253

8.6 Organizing Construction Around the Class-Type Architecture, 255

8.7 What We've Learned, 256
Contents

Chapter 9 Making Your Applications Usable—Object-Oriented User Interface Design, 261

9.1 What Are Object-Oriented User Interfaces?, 262
  9.1.1 An OO Bank Teller Application, 264
  9.1.2 Why OOUs Are Different but the Same, 268

9.2 Designing Effective User Interfaces, 269
  9.2.1 Human Factors, the Study of People Using Machines, 269
  9.2.2 Enough User Interface Design Tips to Sink a Ship, and Then Some, 273

9.3 Developing Effective Object-Oriented User Interfaces, 278
  9.3.1 Applications Are No More, Just Objects, 278
  9.3.2 You’re Working with Objects Too, Not Just Windows, 280
  9.3.3 Modeling Object-Oriented User Interfaces, 281
  9.3.4 Prototyping Object-Oriented User Interfaces, 282

9.4 What We’ve Learned, 287

Chapter 10 Making Your Objects Persistent—Object-Orientation and Databases, 291

10.1 Getting Started—Some Common Terminology, 292

10.2 Saving Your Objects into Flat Files, 295
  10.2.1 An Approach to Mapping Objects to Flat Files, 295

10.3 The Current Reality—Relational Technology and OO Applications, 298
  10.3.1 Normalization, 299
  10.3.2 Overcoming the Object/Relational Impedance Mismatch, 311
  10.3.3 An Approach for Mapping Objects to Relational Databases, 326
  10.3.4 The Seven Commandments for Mapping Objects to Relational Databases, 332
Contents

10.3.5 Looking into My Crystal Ball—OO and RDB in the Future, 334

10.4 Persistence on the Leading Edge—Object-Oriented Databases, 334
  10.4.1 What to Look For in an OODBMS, 335
  10.4.2 Debunking the Myths of Object Databases, 338

10.5 Taking a Hybrid Approach—Object/Relational Databases, 339

10.6 What We’ve Learned, 341

Chapter 11 Integrating Legacy Code—Wrapping, 343

11.1 Why Wrap?, 344

11.2 Approaches to Wrapping, 346
  11.2.1 Leaving Things Well Enough Alone, 347
  11.2.2 Wrapping Only Hardware and Operating System Calls, 347
  11.2.3 Scaffolding Legacy Applications, 347

11.3 Wrapping Technologies, 348
  11.3.1 Wrapping Hardware and Operating-System Calls, 348
  11.3.2 Wrapping Legacy Applications, 351
  11.3.3 Comparing Wrapping Technologies—When to Use Each One, 355

11.4 The Advantages and Disadvantages of Wrapping, 357

11.5 What We’ve Learned, 358

Part IV
OBJECT-ORIENTED TESTING

Chapter 12 Making Sure Your Applications Work—Full Life-Cycle Object-Oriented Testing (FLOOT), 363
Contents

12.1 Why FLOOT?, 365
12.2 Testing Your Previous Efforts—Regression Testing, 367
12.3 Testing Your Analysis, 369
  12.3.1 Use-Case Scenario Testing, 370
  12.3.2 Prototype Walkthroughs, 370
  12.3.3 User-Requirement Reviews, 371
12.4 Testing Your Design, 372
  12.4.1 Technical-Design Reviews, 372
  12.4.2 Requirement-Verification Matrices, 379
12.5 Testing Your Program Code, 380
  12.5.1 Code Reviews, 382
  12.5.2 Traditional Testing Methods, 383
  12.5.3 New Testing Techniques for OO Program Code, 388
  12.5.4 Language-Specific Testing Issues, 398
  12.5.5 Implementing Program Code Test Cases, 398
  12.5.6 A Process for Successfully Testing Your Code, 401
  12.5.7 The Strengths of Each Program Code Testing Technique, 402
12.6 Testing Your Application As a Whole, 403
  12.6.1 System Testing, 403
  12.6.2 User-Acceptance Testing, 407
  12.6.3 Alpha, Beta, and Pilot Testing, 409
12.7 Software Quality Assurance and ISO 9000, 410
12.8 Testing Tips and Techniques, 412
12.9 Full Life-Cycle Object-Oriented Testing Overview, 414
12.10 What We’ve Learned, 415
Contents

Part V
CONCLUSION

Chapter 13  Where to Go from Here—Personal Success Strategies, 419
  13.1 Advice for Overcoming the OO Learning Curve, 420
  13.2 What We Learned in This Book, 422
  13.3 Parting Words, 424

APPENDICES

Appendix A  Notation Summary, 427
  A.1 The Ambler Class Diagramming Notation, v2.0, 427
  A.2 The Unified Modeling Language Class-Diagramming Notation v1.0 (Simplified), 428

Appendix B  Visual Glossary, 429

Index, 457
About the Author

Scott Ambler is an instance of an SeniorOOConsultant with ambySoftInc based in Sharon, Ontario (http://www.ambysoft.com). Messages can be sent to him via the electronic mail facade scott@ambysoft.com.

scottAmbler is a very versatile object that will change type in order to meet the needs of his clients. For example, he often takes on the role of OOMentor, OOTrainer, OOProcessExpert, or OODeveloper. Scott has been an instance of an OOConsultant since 1991. ScottAmbler has instantiated the books The Object Primer (sigsBooksCambridgeUniversityPress, nyCity, 1995), Building Object Applications That Work (sigs-BooksCambridgeUniversityPress, nyCity 1998), and Process Patterns (sigsBooksCambridgeUniversityPress, nyCity 1998). He holds the roles of contributing Editor with SoftwareDevelopment (millerFreemanPress), columnist with ObjectMagazine (sigsPublications), columnist with ComputingCanada (plesmanPublications). scottAmbler is an avid watcher of StarTrekEpisodes, and intends to one day do his doctorate degree at starFleetAcademy.
About the Author

scottAmbler used to be a MastersStudent object, having received a InformationScienceDegree from the UniversityOfToronto. As a MastersStudent, scottAmbler did a lot of work in OO CASE and instantiated a ThesisPaper object in computer-supported co-operative work (an academic alias for groupware). Before being a MastersStudent, he was an instance of a TechnicalSystemAnalyst at RoyalBankOfCanada where he originally became interested in object-orientation.
Foreword

As a broad-faced introduction to object-oriented (OO) technology, this is one of the best books that I have read. It is well written, employing a style that makes it enjoyable to read and easy to understand. Although it covers a diverse range of topics, the coverage is balanced and the topic selection is well chosen.

Topics that usually receive only passing mention in other texts of this kind are included. The first part of the book looks at where we have been in the field and lays a foundation for the rest of the text by introducing general concepts. The second part focuses on architectural issues, and the subjects of object-oriented analysis, design, and patterns are introduced. The author chooses to feature UML notation in the short examples that are presented. The third part introduces OO matrices, user interface design, and OO databases and presents a brief survey of some OO languages. The fourth part of the book focuses on OO testing. All these subjects are covered in much greater detail in more specialized books—but that is to be expected in a book of this kind. Experienced practitioners may object to the omission of important topics and incomplete coverage of some subject areas, but this book is not meant to be a rigorous treatise on object technology; rather, it is a compelling early read for those interested in this area of software development.
Foreword

I recommend this book to programmers and managers who are relatively new to the field of object technology.

Richard Wiener
Editor, JOOP
Associate Professor of Computer Science,
University of Colorado at Colorado Springs
Preface

This book is about:

- The Unified Modeling Language (UML), and how to use it effectively
- Architecting your applications so that they’re maintainable and extensible
- Analysis, so you know what you have to build
- Design techniques, so that you know how you’re going to build your application
- Creating applications for stand-alone, client/server, and distributed environments
- Using both relational and object-oriented (OO) databases to make your objects persistent
- Collecting the right metrics to improve your development approach
- Applying OO patterns to improve the quality of your applications
- Testing approaches to ensure that your applications work
Preface

• User interface design, so your users can actually work with the systems that you build

• Coding applications in a way that makes them maintainable and extensible

Welcome to Building Object Applications. To make this book easier to work with it is divided into the following five parts:

Part I Introduction
Chapter 1 Where We’ve Been Before—Object-Oriented Concepts and Techniques

Part II Object-Oriented Analysis, Design, and Architecture
Chapter 2 Bubbles and Lines—Useful Diagrams for Object-Oriented Analysis and Design
Chapter 3 Improving Your Design—A Class-Type Architecture
Chapter 4 Reusing Your Development Efforts—Object-Oriented Patterns
Chapter 5 Development in the 90s and Beyond—Designing Distributed Object-Oriented Applications

Part III Object-Oriented Construction
Chapter 6 Measuring and Improving the Quality of Your Work—Object-Oriented Metrics
Chapter 7 Choosing an Object-Oriented Language—Comparing the Leading Languages
Chapter 8 Building Your Application—Effective Object-Oriented Construction Techniques
Chapter 9 Making Your Applications Usable—Object-Oriented User Interface Design
Chapter 10 Making Your Objects Persistent—Object-Orientation and Databases
Chapter 11 Integrating Legacy Code—Wrapping

Part IV Object-Oriented Testing
Chapter 12 Making Sure Your Applications Work—Full Life-Cycle Object-Oriented Testing
Preface

Part V Conclusion

Chapter 13 Where to Go from Here—Personal Success Strategies

Introduction to Building Object Applications

This volume starts where The Object Primer left off—OO design. Well, actually it starts with a review of, and a few improvements to The Object Primer. To keep things simple I purposely left out a few things in the first volume, specifically class methods and class attributes. These two concepts are just too darn confusing for OO beginners, so I skipped them until this book. Besides, I wanted to make sure there was enough new stuff in the first chapter to make reading it worthwhile.

Object-Oriented Analysis, Design, and Architecture

In the second chapter we'll discuss several diagramming techniques that we didn't cover in the first book. We'll also learn some really good tips and techniques to make your diagrams look better, as well as how to convert back and forth from an Ambler Class Model to the notations of the other leading OO methodologies.

Chapter 3 introduces us to an OO class architecture. We'll propose a four-level architecture that actually has five and maybe even six levels, but who's counting? We then get into OO patterns in chapter 4. You've heard a lot about them, you've seen the huge tomes, now it's time to actually find out what this stuff is really all about. Because the proof is always in the pudding, but never in the jelly (for some reason) we'll look at several patterns that are being used in live applications today.

In chapter 5 we'll delve into distributed OO architectures. Because the client/server (C/S) architecture is currently the dominant approach to systems development, I figured we'd better show you how to design for C/S. We'll discuss the seven steps for creating an object-oriented client/server (OOCs) design. C/S can be very complex, and experience has shown that OO is the surest way to guarantee success in a C/S environment. We'll also discuss how OOCs leads to distributed objects, an architecture that is quickly catching on and that is most likely in your future.
Preface

Object-Oriented Construction

Chapter 6 covers object-oriented metrics that you can use both to estimate the size of a project and to evaluate how well you are doing. The design and coding implications that they make are what makes metrics important for developers. Understanding the basis for metrics will help you to improve your development skills.

We're going to spend a fair bit of time talking about OO languages and OO construction issues in chapters 7 and 8, so all you programmers out there might want to skip immediately to those chapters. Although there are a fair number of OO programming languages, we're going to concentrate on Smalltalk, C++, Java, and ObjectCOBOL. We're going to cover a lot of programming tips in these chapters, so I think that you'll really like it.

Chapter 9 covers interface design because it's critical that we design applications that are user-friendly. I begin by arguing that just because you use a graphical-user interface (GUI) it isn't necessarily OO. We'll see designs for both a standard GUI interface for the bank and an object-oriented user interface (OOUI) for it. When you see the difference you'll be astounded.

As saving objects is a reasonably common thing to do, we're going to cover this topic in detail in chapter 10. Relational technology is the reality out there, not object databases, so for most of the chapter I'll assume you're trying to store your objects in a relational database (DB). Unfortunately, this is easier said than done. We're going to discuss issues like OO normalization, converting objects to relational records, inheritance issues, and design issues when using a relational DB to store objects. The chapter ends with a discussion of both object/relational and object databases, what they're all about and where they're going. The bottom line is that object databases are an important topic that must be covered.

Chapter 11 discusses wrapping, which is a technique for getting a few more miles out of your legacy applications. If the legacy applications were horses we would have shot them long ago, but unfortunately we've made a huge investment in them that we'd prefer not to lose.
Preface

Object-Oriented Testing

As testing is always left to the very end of application development, that's exactly where we've put the testing chapter, at the back of the book. Until now, OO testing hasn't been well covered by the mainstream OO methodologies, and that's a problem we want to address. Users really like it when their application works, so it's not a bad idea to test your application before you release it. The bad news is that traditional testing techniques don't always work for OO applications. The good news is that chapter 12 describes several OO testing methods that are quite effective.

Conclusion

In chapter 13 we'll wrap up the book with a few pointers on how to succeed at learning object-oriented development techniques. They might seem obvious, but considering how many people have run into trouble learning this stuff, I highly suggest that you follow the advice in this chapter.

Who Should Read Building Object Applications?

For the most part, this book is geared toward designers, programmers, and testers. Analysts will find this material useful for filling out their knowledge of OO, but probably won't find it directly applicable to their daily jobs. Project managers of OO development teams should definitely read this book to gain a better understanding of what their people are doing.

Why the Unified Modeling Language?

In this book I've decided to move away from my original modeling notation to the Unified Modeling Language (UML), the notation that is quickly becoming the industry standard. I wanted to keep this book as simple as possible to communicate the fundamentals of OO development. I also wanted to do so in a manner that the knowledge you
Preface

Gain is directly applicable on your job. This is why I've adopted the UML, and why I hope that your organization adopts the UML as its standard notation as well. If you've read The Object Primer and have learned my notation, you don't have to worry; in chapters 1 and 2 we'll transition from the Ambler Notation to the UML very smoothly.
Acknowledgments

SPECIAL THANKS GO TO my friends at Mark Winter and Associates, who provided both excellent input and sounding boards for many of the concepts presented in this book. I’d also like to thank the friends that I have made at several client organizations while writing this book (although I can’t reveal the organizations due to confidentiality issues, they were in Toronto, Rio, Buffalo, and Anchorage, so you folks know who you are). All of you helped to shape the experiences on which this book is based and I greatly appreciate it.