MASTERRING THE REQUIREMENTS PROCESS
GETTING REQUIREMENTS RIGHT
Suzanne Robertson • James Robertson
FREE SAMPLE CHAPTER
SHARE WITH OTHERS
Requirements Strategy Maps

External Requirements Strategy

Iterative Requirements Strategy

Sequential Requirements Strategy
Iterative Requirements Process

1. **Analyze Business Needs**
   - Business Event List (Analysis Backlog)
   - Business Needs
   - Prioritized
   - Analysis Artifacts
     - Context
     - BUCs
     - Data Dictionary
     - Stakeholders

2. **Write Requirements**
   - Requirement
   - Requirement
   - Requirement
   - Feedback
   - Prioritized

3. **Develop Product**
   - Working Product
   - Prioritized
   - Feedback

4. **The Work**
   - Development Backlog
   - Requirement
   - Feedback
   - Prioritized

**Iteration:**
- Analyze Business Needs
- Write Requirements
- Develop Product
- Feedback and Prioritization

The process iterates between analyzing business needs, writing requirements, and developing the product, with feedback and prioritization at each step.
This page intentionally left blank
Mastering the Requirements Process
Third Edition
This page intentionally left blank
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 the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

Rabbit, horse, and elephant icons courtesy of all-silhouettes.com. Owl icon courtesy of iStockphoto.com; all rights reserved.

The authors 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.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

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

For sales outside the United States, please contact:  
International Sales  
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data
Robertson, Suzanne.  
Mastering the requirements process : getting requirements right / Suzanne Robertson, James Robertson.—3rd ed.  
p. cm.  
Includes bibliographical references and index.  
TA190.R48 2012  
005.1068’4—dc23  
2012018961

Copyright © 2013 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.  
Second printing, September 2013
For one generation,

Reginald, Margaret, Nick, and Helen,

and another,

Carlotta, Cameron, and Louise
This page intentionally left blank
Contents

Preface to the Third Edition  xxi
Foreword to the First Edition  xxiii
Acknowledgments  xxv

1 Some Fundamental Truths  1
in which we consider the essential contribution of requirements

Truth 1  1
Truth 2  2
Truth 3  3
Truth 4  4
Truth 5  5
Truth 6  6
Truth 7  7
Truth 8  7
Truth 9  8
Truth 10  8
Truth 11  9
What Are These Requirements Anyway?
  Functional Requirements  10
  Non-functional Requirements  10
  Constraints  11
The Volere Requirements Process  11

2 The Requirements Process  13
in which we present a process for discovering requirements and
discuss how you might use it

The Requirements Process in Context  14
A Case Study  15
Project Blastoff  15
Trawling for Requirements  17
Quick and Dirty Modeling  19
Scenarios  20
Writing the Requirements  20
3 Scoping the Business Problem

in which we establish a definition of the business area to be changed, thereby ensuring that the project team has a clear vision of what their project is meant to achieve

Project Blastoff 35
Formality Guide 38
Setting the Scope 38
Separate the Work from its Environment 40
IceBreaker 41
First-Cut Work Context 42
Scope, Stakeholders, and Goals 43
Stakeholders 44
The Sponsor 45
The Customer 47
Users: Understand Them 48
Other Stakeholders 50
Consultants 51
Management 51
Subject-Matter Experts 51
Core Team 51
Inspectors 52
Market Forces 52
Legal Experts 52
Negative Stakeholders 52
Industry Standard Setters 52
Public Opinion 53
Government 53
Special-Interest Groups 53
Technical Experts 53
Cultural Interests 53
Adjacent Systems 53
Finding the Stakeholders 54
Goals: What Do You Want to Achieve? 54
Purpose 55
Advantage 56
Measurement 56
Constraints 59
  Solution Constraints 59
  Project Constraints 60
Naming Conventions and Definitions 60
How Much Is This Going to Cost? 61
Risks 62
To Go or Not to Go 63
Blastoff Meetings 64
Summary 65

4 Business Use Cases 67
  in which we discuss a fail-safe way of partitioning the work and so smooth the way for your requirements investigation
Understanding the Work 67
Formality Guide 69
Use Cases and Their Scope 69
The Scope of the Work 70
  The Outside World 72
Business Events 73
  Time-Triggered Business Events 74
Why Business Events and Business Use Cases Are a Good Idea 75
  The “System” Cannot Be Assumed 76
  Step Back 77
Finding the Business Events 78
Business Use Cases 80
Business Use Cases and Product Use Cases 82
  Actors 84
Summary 85

5 Investigating the Work 87
  in which we come to an understanding of what the business is doing, and start to think about what it might like to do
Trawling the Business 87
Formality Guide 89
Trawl for Knowledge 89
The Business Analyst 91
Trawling and Business Use Cases 92
The Brown Cow Model 93
The Current Way of Doing Things (How-Now) 94
Apprenticing 98
Business Use Case Workshops 99
  Outcome 101
  Scenarios 101
  Business Rules 101
Interviewing the Stakeholders 102
  Asking the Right Questions 104
  Listening to the Answers 105
# Contents

Looking for Reusable Requirements 106
Quick and Dirty Process Modeling 107
Prototypes and Sketches 109
  
  Low-Fidelity Prototypes 111
  High-Fidelity Prototypes 115
Mind Maps 116
The Murder Book 119
Video and Photographs 120
Wikis, Blogs, Discussion Forums 122
Document Archeology 123
Family Therapy 125
Choosing the Best Trawling Technique 125
Finally . . . 127

## 6 Scenarios

in which we look at scenarios, and how the business analyst uses them to communicate with the stakeholders

Formality Guide 129
Scenarios 130
The Essence of the Business 135
Diagramming the Scenario 138
Alternatives 139
Exceptions 140
What if? Scenarios 142
Misuse Cases and Negative Scenarios 142
Scenario Template 143
Summary 145

## 7 Understanding the Real Problem

in which we “think above the line” to find the true essence of the business, and so deliver the right product—one that solves the right problem

Formality Guide 149
The Brown Cow Model: Thinking Above the Line 149
  
  The Essence 150
  Abstraction 153
  Swim Lanes Begone 154
Solving the Right Problem 156
Moving into the Future 157
How to Be Innovative 160
Systemic Thinking 162
Value 165
Personas 166
Challenging Constraints 169
Innovation Workshops 171
Brainstorming 173
Back to the Future 174
## 8 Starting the Solution

*in which we bring the essence of the business into the technological world of the implementation*

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Development</td>
<td>179</td>
</tr>
<tr>
<td>Essential Business</td>
<td>179</td>
</tr>
<tr>
<td>Determine the Extent of the Product</td>
<td>180</td>
</tr>
<tr>
<td>Consider the Users</td>
<td>181</td>
</tr>
<tr>
<td>Designing the User Experience</td>
<td>183</td>
</tr>
<tr>
<td>Innovation</td>
<td>184</td>
</tr>
<tr>
<td> Convenience</td>
<td>184</td>
</tr>
<tr>
<td> Connections</td>
<td>185</td>
</tr>
<tr>
<td> Information</td>
<td>186</td>
</tr>
<tr>
<td> Feeling</td>
<td>187</td>
</tr>
<tr>
<td>Sketching the Interface</td>
<td>188</td>
</tr>
<tr>
<td>The Real Origin of the Business Event</td>
<td>189</td>
</tr>
<tr>
<td>Adjacent Systems and External Technology</td>
<td>190</td>
</tr>
<tr>
<td> Active Adjacent Systems</td>
<td>190</td>
</tr>
<tr>
<td> Autonomous Adjacent Systems</td>
<td>192</td>
</tr>
<tr>
<td> Cooperative Adjacent Systems</td>
<td>193</td>
</tr>
<tr>
<td>Cost, Benefit, and Risks</td>
<td>194</td>
</tr>
<tr>
<td>Document Your Design Decisions</td>
<td>195</td>
</tr>
<tr>
<td>Product Use Case Scenarios</td>
<td>196</td>
</tr>
<tr>
<td>Putting It All Together</td>
<td>199</td>
</tr>
</tbody>
</table>

## 9 Strategies for Today’s Business Analyst

*in which we consider strategies for the business analyst to guide requirements discovery in today’s changing environments*

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balancing Knowledge, Activities, and People</td>
<td>204</td>
</tr>
<tr>
<td>Common Project Requirements Profiles</td>
<td>204</td>
</tr>
<tr>
<td>How Much Knowledge Is Needed Before Each Breakout?</td>
<td>205</td>
</tr>
<tr>
<td>External Strategy</td>
<td>206</td>
</tr>
<tr>
<td> Conception to Scoping</td>
<td>207</td>
</tr>
<tr>
<td> Scoping to Work Investigation</td>
<td>207</td>
</tr>
<tr>
<td> Work Investigation to Product Determination</td>
<td>208</td>
</tr>
<tr>
<td> Work Investigation to Atomic Requirements Definition</td>
<td>208</td>
</tr>
<tr>
<td> Work Investigation to Building</td>
<td>208</td>
</tr>
<tr>
<td> Product Determination to Atomic Requirements Definition</td>
<td>209</td>
</tr>
<tr>
<td> Product Determination to Construction</td>
<td>209</td>
</tr>
<tr>
<td> Atomic Requirements Definition to Building</td>
<td>209</td>
</tr>
<tr>
<td>Iterative Strategy</td>
<td>210</td>
</tr>
<tr>
<td> Conception to Scoping</td>
<td>210</td>
</tr>
<tr>
<td> Scoping to Work Investigation</td>
<td>210</td>
</tr>
<tr>
<td> Work Investigation to Product Determination</td>
<td>211</td>
</tr>
<tr>
<td> Work Investigation to Requirements Definition</td>
<td>211</td>
</tr>
<tr>
<td> Product Determination to Requirements Definition</td>
<td>212</td>
</tr>
<tr>
<td> Requirements Definition to Construction</td>
<td>212</td>
</tr>
<tr>
<td>Sequential Strategy</td>
<td>212</td>
</tr>
<tr>
<td> Conception to Scoping</td>
<td>213</td>
</tr>
<tr>
<td> Scoping to Work Investigation</td>
<td>213</td>
</tr>
</tbody>
</table>
10 Functional Requirements

in which we look at those requirements that cause the product to do something

Formality Guide 224
Functional Requirements 225
Uncovering the Functional Requirements 225
Level of Detail or Granularity 228
Description and Rationale 229
Data, Your Secret Weapon 231
Data Models 231
Data Dictionary 232
Exceptions and Alternatives 233
Conditional Requirements 234
Avoiding Ambiguity 234
Technological Requirements 237
Grouping Requirements 237
Alternatives to Functional Requirements 238
Scenarios 239
User Stories 239
Business Process Models 240
Requirements for COTS 241
Summary 242

11 Non-functional Requirements

in which we look at the requirements that specify how well your product does what it does

An Introduction to Non-functional Requirements 246
Formality Guide 246
Functional Versus Non-functional Requirements 247
Use Cases and Non-functional Requirements 248
The Non-functional Requirements Types 249
Look and Feel Requirements: Type 10 250
Usability and Humanity Requirements: Type 11 253
## Contents

Performance Requirements: Type 12 257
Operational and Environmental Requirements: Type 13 259
Maintainability and Support Requirements: Type 14 261
Security Requirements: Type 15 262
  - Access 263
  - Privacy 263
  - Integrity 264
  - Auditing 265
  - . . . And No More 265
Cultural Requirements: Type 16 266
Legal Requirements: Type 17 268
  - Sarbanes-Oxley Act 269
  - Other Legal Obligations 270
  - Standards 271
Finding the Non-functional Requirements 271
  - Blogging the Requirements 271
  - Use Cases 272
  - The Template 274
  - Prototypes and Non-functional Requirements 274
  - The Client 275
Don’t Write a Solution 276
Summary 277

12 Fit Criteria and Rationale 279
  
in which we show how measuring requirements makes them unambiguous, understandable, communicable, and testable

Formality Guide 280
Why Does Fit Need a Criterion? 280
The Rationale for the Rationale 282
Deriving Fit Criteria 284
Scale of Measurement 285
Fit Criteria for Non-functional Requirements 286
  - Product Failure 288
  - Subjective Tests 289
  - Standards 289
  - Look and Feel Requirements 290
  - Usability and Humanity Requirements 291
  - Performance Requirements 292
  - Operational Requirements 293
  - Maintainability Requirements 294
  - Security Requirements 294
  - Cultural Requirements 294
  - Legal Requirements 295
Fit Criteria for Functional Requirements 295
  - Test Cases 296
Forms of Fit Criteria 296
  - Defining the Data 297
  - Graphic Fit Criteria 297
  - Decision Tables 297
  - Graphs 298
13 The Quality Gateway
in which we prevent unsuitable requirements from becoming part of the specification

- Formality Guide
- Requirements Quality
- Using the Quality Gateway
- Within Scope?
  - Relevancy
- Testing Completeness
  - Are There Any Missing Attributes?
  - Meaningful to Stakeholders?
- Testing the Fit Criterion
- Consistent Terminology
- Viable within Constraints?
- Requirement or Solution?
- Requirement Value
- Gold Plating
- Requirements Creep
- Implementing the Quality Gateway
  - Alternative Quality Gateways
- Summary

14 Requirements and Iterative Development
in which we look at how to discover and implement requirements in an iterative development environment

- The Need for Iterative Development
- An Iterative Requirements Process
  - The Work
    - Analyze Business Needs
    - Write User Stories
    - Develop Product
- Business Value Analysis and Prioritization
- How to Write a Good User Story
  - Questions to Ask
  - Formalizing Your User Stories
  - Flesching out the Story
- Iterative Requirements Roles
  - Business Knowledge
  - Analytical and Communication Knowledge
  - Technical Knowledge
- Summary
15 Reusing Requirements

15 Reusing Requirements

in which we look for requirements that have already been written and explore ways to make use of them

What Is Reusing Requirements? 338
Sources of Reusable Requirements 341
Requirements Patterns 342
Christopher Alexander’s Patterns 343
A Business Event Pattern 344
Context of Event Response 344
Processing for Event Response 345
Data for Event Response 345
Forming Patterns by Abstracting 346
Patterns for Specific Domains 348
Patterns Across Domains 349
Domain Analysis 351
Summary 351

16 Communicating the Requirements

in which we turn the requirements into communicable form

Formality Guide 353
Turning Potential Requirements into Written Requirements 354
Knowledge Versus Specification 354
The Volere Requirements Specification Template 357
Template Table of Contents 357
Template Divisions 358
Discovering Atomic Requirements 359
Snow Cards 359
Attributes of Atomic Requirements 361
Requirement Number 361
Requirement Type 361
Event/BUC/PUC # 361
Description 362
Rationale 362
Originator 363
Fit Criterion 363
Customer Satisfaction and Customer Dissatisfaction 363
Priority 364
Conflicts 364
Supporting Materials 365
History 365
Assembling the Specification 365
Automated Requirements Tools 366
Functional Requirements 367
Non-functional Requirements 368
Project Issues 369
Summary 369
17 Requirements Completeness

in which we decide whether our specification is complete, and set the priorities of the requirements

Formality Guide
Reviewing the Specification
Inspections
Find Missing Requirements
Have All Business Use Cases Been Discovered?
  1. Define the Scope
  2. Identify Business Events and Non-events
     Non-events
  3. Model the Business Use Case
  4. Define the Business Data
  5. CRUD Check
  6. Check for Custodial Processes
     Repeat Until Done
Prioritizing the Requirements
  Prioritization Factors
  When to Prioritize
  Requirement Priority Grading
  Prioritization Spreadsheet
Conflicting Requirements
Ambiguous Specifications
Risk Assessment
  Project Drivers
  Project Constraints
  Functional Requirements
Measure the Required Cost
Summary

Appendix A Volere Requirements Specification Template

a guide for writing a rigorous and complete requirements specification

Contents
  Project Drivers
  Project Constraints
  Functional Requirements
  Non-functional Requirements
  Project Issues
Use of This Template
Volere
Requirements Types
Testing Requirements
Atomic Requirements Shell
  1. The Purpose of the Project
     1a. The User Business or Background of the Project Effort
     1b. Goals of the Project
  2. The Stakeholders
     2a. The Client
12d. Reliability and Availability Requirements 444
12e. Robustness or Fault-Tolerance Requirements 445
12f. Capacity Requirements 445
12g. Scalability or Extensibility Requirements 446
12h. Longevity Requirements 446
13. Operational and Environmental Requirements 447
  13a. Expected Physical Environment 447
  13b. Requirements for Interfacing with Adjacent Systems 447
  13c. Productization Requirements 448
  13d. Release Requirements 449
14. Maintainability and Support Requirements 449
  14a. Maintenance Requirements 449
  14b. Supportability Requirements 450
  14c. Adaptability Requirements 450
15. Security Requirements 451
  15a. Access Requirements 451
  15b. Integrity Requirements 452
  15c. Privacy Requirements 453
  15d. Audit Requirements 454
  15e. Immunity Requirements 454
16. Cultural Requirements 454
  16a. Cultural Requirements 454
17. Legal Requirements 455
  17a. Compliance Requirements 455
  17b. Standards Requirements 456
Project Issues 457
18. Open Issues 457
19. Off-the-Shelf Solutions 458
  19a. Ready-Made Products 458
  19b. Reusable Components 459
  19c. Products That Can Be Copied 459
20. New Problems 460
  20a. Effects on the Current Environment 460
  20b. Effects on the Installed Systems 460
  20c. Potential User Problems 461
  20d. Limitations in the Anticipated Implementation Environment That May Inhibit the New Product 461
  20e. Follow-Up Problems 462
21. Tasks 462
  21a. Project Planning 462
  21b. Planning of the Development Phases 463
22. Migration to the New Product 463
  22a. Requirements for Migration to the New Product 464
  22b. Data That Must Be Modified or Translated for the New System 465
23. Risks 465
24. Costs 467
25. User Documentation and Training 468
  25a. User Documentation Requirements 468
  25b. Training Requirements 469
26. Waiting Room 470
27. Ideas for Solutions 471
Appendix B Stakeholder Management Templates 473
  Stakeholder Map 473
  Stakeholder Template 475

Appendix C Function Point Counting: A Simplified Introduction 479
  in which we look at a way to accurately measure the size or functionality of the work area, with a view toward using the measurement to estimate the requirements effort
  Measuring the Work 479
  A Quick Primer on Counting Function Points 481
    Scope of the Work 481
    Data Stored by the Work 482
    Business Use Cases 483
  Counting Function Points for Business Use Cases 484
    Counting Input Business Use Cases 484
    Counting Output Business Use Cases 485
    Counting Time-Triggered Business Use Cases 487
  Counting the Stored Data 489
    Internal Stored Data 489
    Externally Stored Data 490
  Adjust for What You Don’t Know 492
  Now That I Have Counted Function Points, What’s Next? 492

Appendix D Volere Requirements Knowledge Model 495
  Definitions of Requirements Knowledge Classes and Associations 495
    Knowledge Classes 496
    Associations 505
  Knowledge Model Annotated with Template Section Numbers 508

Glossary 511

Bibliography 517

Index 523
Preface to the Third Edition

Why a third edition of *Mastering the Requirements Process*? Because we need it. Much water has passed under the bridge since the last edition of this book was published, and much has happened in the requirements and development world. We have applied the Volere requirements techniques described in this book to many projects; we have received feedback from our projects and those of clients and other practitioners of the Volere techniques; and armed with that knowledge we felt it was time to update our book to reflect the current state of requirements practice. Today’s systems, software, products, and services have to be more attractive and more appropriate if they are to be noticed, bought, used and valued. More than ever, we need to be assured that we are solving the real problem. More than ever, we need to be doing a better job with requirements discovery.

New techniques for software development—most noticeably the rise of agile techniques—have changed the role of the requirements discoverer: not the underlying truth of the requirements activity, but the way in which requirements are discovered. Business analysts working with agile teams perform their task differently. Combinations of iterative, incremental, and spiral development techniques require the business analyst to go about the requirements task in a different way.

Outsourcing has increased enormously, which, rather than lessening the requirements burden, means that there is an even greater need to produce accurate, and unambiguous, requirements. If you are planning to send your specification to the far side of the world, you would like to think that your outsourcer will understand it and know exactly what to build.

Despite all these changes in the way in which we develop and deliver our products and services, one underlying fact is still there, and it is this: *If we are to build some software or a product or a service, then it must provide the optimal value for its owner.*

You will see the theme of optimal value developed in this edition, and what it comes down to is that it does not matter how you develop your software, but rather what that software does for its owner that matters. You can
finish a project on time and on budget, but if the delivered software brings little benefit to the owning organization, it is a waste of money. Alternatively, you can overspend and be late, but if the delivered product brings several million dollars of value, then it is more beneficial than its cheaper counterpart.

The task of the business analyst is to discover the real business that the software is supposed to improve. This cannot be done at the keyboard simply because software is a solution, and to provide a valuable solution you first have to understand the problem—the real problem—that it is meant to solve. In this edition we have written about thinking above the line. The line in this case comes from the Brown Cow Model (you’ll have to read the book to find out what it is) and represents the division between the technological implementations and the abstract, essential world where you discover the real needs. We have written about innovation as a way of finding better, more appropriate needs and solutions.

This, then, is the task of the requirements discoverer, and indeed of this edition: to delve more deeply into how we understand our client organizations, and how we find better solutions by discovering and communicating a better understanding of the problem.

London, June 2012

For college instructors who adopt this book for their courses, some of the graphics used herein are available in the Pearson Instructor Resource Center (www.pearsonhighered.com) for your use in preparing course materials.
Foreword to the First Edition

It is almost ten years now since Don Gause and I published Exploring Requirements: Quality Before Design. Our book is indeed an exploration, a survey of human processes that can be used in gathering complete, correct, and communicable requirements for a software system, or any other kind of product.

The operative word in this description is “can,” for over this decade the most frequent question my clients have asked is, “How can I assemble these diverse processes into a comprehensive requirements process for our information systems?”

At long last, James and Suzanne Robertson have provided an answer I can conscientiously give to my clients. Mastering the Requirements Process shows, step by step, template by template, example by example, one well-tested way to assemble a complete, comprehensive requirements process.

One watchword of their process is “reasonableness.” In other words, every part of the process makes sense, even to people who are not very experienced with requirements work. When introducing this kind of structure to an organization, reasonableness translates into easier acceptance—an essential attribute when so many complicated processes are tried and rejected.

The process they describe is the Volere approach, which they developed as an outcome of many years helping clients to improve their requirements. Aside from the Volere approach itself, James and Suzanne contribute their superb teaching skills to the formidable task facing anyone who wishes to develop requirements and do them well.

The Robertsons’ teaching skills are well known to their seminar students as well as to fans of their Complete Systems Analysis books. Mastering the Requirements Process provides a much-requested front end for their analysis books—or for anyone’s analysis books, for that matter.

We can use all the good books on requirements we can get, and this is one of them!

Gerald M. Weinberg
www.geraldmweinberg.com
February 1999
This page intentionally left blank
Writing a book is hard. Without the help and encouragement of others, it would be nearly impossible, at least for these authors. We would like to take a few lines to tell you who helped and encouraged and made it possible.

Andy McDonald of Vaisala was generous with his time, and gave us considerable technical input. We hasten to add that the IceBreaker product in this book is only a distant relation to Vaisala’s IceCast systems. The Vaisala User Group, of which E. M. Kennedy holds the chair, also provided valuable technical input.

Thanks are due to the technical reviewers who gave up their time to wade through some fairly incomprehensible stuff. Mike Russell, Susannah Finzi, Neil Maiden, Tim Lister, and Bashar Nuseibeh all deserve honorable mentions.

We would like to acknowledge our fellow principals at the Atlantic Systems Guild—Tom DeMarco, Peter Hruschka, Tim Lister, Steve McMenamin, and John Palmer—for their help, guidance, and incredulous looks over the years.

The staff at Pearson Education contributed. Sally Mortimore, Alison Birtwell, and Dylan Reisenberger were generous and skillful, and used such persuasive language whenever we spoke about extending the deadline.

For the second edition, Peter Gordon provided guidance and persuasion at exactly the right times. Kim Boedigheimer, John Fuller, and Lara Wysong were invaluable at steering us through the publishing process. Jill Hobbs tamed our faulty grammar and punctuation, and made this text readable. The technical input of Ian Alexander, Earl Beede, Capers Jones, and Tony Wasserman goes far beyond valuable. Thank you, gentlemen, for your insights. And we hasten to add that any remaining technical errors are ours and ours alone.

One would imagine that by the time one got to the third edition, one would not need help. Not so. We gratefully acknowledge the alphabetic trinity of Gary Austin, Earl Beede, and John Capron. Our Volere colleague Stephen Mellor sorted out some of the trickier issues we encountered. Our
other Volere colleagues James Archer and Andrew Kendall have helped over the years with their ideas, experience, and meaningful conversations over a glass of wine.

The Pearson crew of Peter Gordon, Kim Boedigheimer, and Julie Nahil were invaluable. We want to point out the special work done by Alan Clements to design the cover. Once again, Jill Hobbs stepped up to tame our grammatical misdemeanors and semantic transgressions.

And finally we thank the students at our seminars and our consulting clients. Their comments, their insistence on having things clearly explained, their insights, and their feedback have all made some difference, no matter how indirect, to this book.

Thank you, everybody.

James and Suzanne Robertson
London, June 2012
The Requirements Process

in which we present a process for discovering requirements and discuss how you might use it

This book is a distillation of our experience. In it, we describe a requirements process that we have derived from our years of working in the requirements arena—working with clever people who do clever things, and working on projects in wonderfully diverse domains. We have also learned much from the experience of the many people around the world who use various parts of our techniques.

We developed the Volere Requirements Process and its associated specification template from the activities and deliverables that had proved themselves to be most effective in project and consulting assignments with our clients. The result of this experience is a requirements discovery and specification process whose principles can be applied—and indeed have been applied—to almost all kinds of application types in almost all kinds of development environments.

We want to stress from the very beginning that while we are presenting a process, we are using it as a vehicle for discovering requirements; we do not expect you to wave this process around and tell your co-workers that it is “the only way to do things.” However, we have high expectations that you will find many useful things from this process that will, in turn, help you to discover and communicate your requirements more productively and accurately. We have personally seen hundreds of companies adapt the process to their own cultures and organizations, and we know of thousands more that have done so.

Our clients who use the Volere Requirements Process are those who develop their products using RUP, incremental, iterative, spiral, Scrum, or other variations of iterative development; more formalized waterfall processes; and a variety of homebrewed development processes. Over the years,
all of these clients agreed with us: If the right product is to be built, the right requirements have to be discovered. But requirements don’t come about by fortuitous accident. To find the correct and complete requirements, you need some kind of orderly process.

The Volere Requirements Process is shown in Figure 2.1. Each of the activities included in the figure, along with the connections between them, is described in detail in subsequent chapters of this book.

**The Requirements Process in Context**

We need to point out—indeed, we need to stress—that this process is not intended to be a waterfall approach. At various stages throughout this book, we will point out how you might modify the process if you are using some kind of iterative development.
Requirements discovery should be seen as a necessary forerunner of any construction activity, but it should also be viewed as something that can be conducted quite quickly, sometimes quite informally, sometimes overlapping with subsequent design and construction activities, but never ignored.

Let’s look briefly at each of the activities shown in Figure 2.1, which are covered in more detail in subsequent chapters. The intention of this chapter is to give you a gentle introduction to the process, its components, its deliverables, and the ways that they fit together. If you want more detail on any of the activities, feel free to jump ahead to the relevant chapter before completing this overview.

As we go through the process, we describe it as if you were working with a brand-new product—that is, developing something from scratch. We take this approach to avoid, for the moment, becoming entangled in the constraints that are part of all maintenance projects. Later, we will discuss requirements for those situations when the product already exists and changes to it are required.

A Case Study

We will explain the Volere Requirements Process by taking you through a project that uses it.

The IceBreaker project is to develop a product that predicts when and where ice will form on roads, and to schedule trucks to treat the roads with de-icing material. The new product will enable road authorities to more accurately predict ice formation, schedule road treatments more precisely, and thereby make the roads safer. The product will also reduce the amount of de-icing material needed, which will help both the road authority’s finances and the environment.

Project Blastoff

Imagine launching a rocket. 10 – 9 – 8 – 7 – 6 – 5 – 4 – 3 – 2 – 1 – blastoff! If all it needed were the ability to count backward from 10, then even Andorra¹ would have its own space program. The truth of the matter is that before we get to the final 10 seconds of a rocket launch, a lot of preparation has taken place. The rocket has been fueled, and the course plotted—in fact, everything that needs to be done if the rocket is to survive and complete a successful mission.

The key purpose of the project blastoff is to build the foundation for the requirements discovery that is to follow, and to ensure that all the needed components for a successful project are in place. The principal stakeholders—the sponsor, the key users, the lead requirements analyst, technical and business experts, and other people who are crucial to the success of the project—gather together to arrive at a consensus on the crucial project issues.

---

¹ The likelihood of frost or ice forming is determined by the energy receipt and loss at the road surface. This energy flow is controlled by a number of environmental and meteorological factors (such as exposure, altitude, road construction, traffic, cloud cover, and wind speed). These factors cause significant variation in road surface temperature from time to time and from one location to another. Winter night-time road surface temperatures can vary by over 10°C across a road network in a county.

—Vaisala News

Blastoff is also known as “project initiation,” “kickoff,” “charter,” “project launch,” and many other things. We use the term “blastoff” to describe what we are trying to achieve—getting the requirements project launched and flying.

---

FOOTNOTE 1

Andorra is a tiny principality in the Pyrenees mountains between France and Spain. Only since 1993 has it been
The blastoff defines the scope of the business problem and seeks concurrence from the stakeholders that yes, this is the area of the owner’s organization that needs to be improved. The blastoff meeting confirms the functionality to be included in the requirements discovery, and the functionality that is to be specifically excluded.

Defining the scope of the business problem is usually the most convenient way to start. In the IceBreaker project, the lead requirements analyst coordinates the group members’ discussion as they come to a consensus on the scope of the work—that is, the business area to be improved—and how this work relates to the world around it. The meeting participants draw a context diagram on a whiteboard to show which functionality is included in the work, and by extension, which elements they consider to be outside the scope of the ice forecasting business. The diagram defines—precisely defines—the included functionality by showing the connections between the work and the outside world. (More on this in the next chapter.) This use of a context diagram is illustrated in Figure 2.2. Later, as the requirements activity proceeds, the context diagram is used to reveal the optimal product to help with this work.

When they have reached a reasonable agreement on the scope of the business area to be studied, the group identifies the stakeholders. The stakeholders are those people who have an interest in the product, or who have knowledge pertaining to the product—in fact, anyone who has requirements for it. For the IceBreaker project, the people who have an interest are the road engineers, the truck depot supervisor, the weather forecasting people, road safety experts, ice treatment consultants, and so on. These people must be identified, so that the requirements analysts can work with them to find all the requirements. The context diagram, by establishing the extent of the work, helps to identify many of the stakeholders.

Refer to Chapter 3, Scoping the Business Problem, for a detailed discussion of project blastoff.
The blastoff also confirms the goals of the project. The blastoff group comes to an agreement on the business reason for doing the project, and agrees that there is a clear and measurable benefit to be gained by doing the project. The group also agrees that the product is worthwhile for the business to make the investment, and that the organization is capable of building and operating it.

It is sensible project management practice at this stage to produce a preliminary estimate of the costs involved for the requirements part of the project—this can be done by using the information already contained in the context diagram. It is also sensible project management to make an early assessment of the risks that the project is likely to face. Although these risks might seem like depressing news, it is always better to get an idea of the downside of the project (its risk and cost) before being swept away by the euphoria of the benefits that the new product is intended to bring.

The blastoff group members arrive at a consensus on whether the project is worthwhile and viable—that is, they make the “go/no go” decision. It might seem brutal to kill off an embryonic project, but we know from bitter experience that it is better to cancel a project at an early stage than to have it stagger on for months—or years—consuming valuable resources when it has little or no chance of success. The blastoff group carefully considers whether the product is viable, and whether its benefits outweigh its costs and risks.

Alternatively, if too many unknowns remain at this point, the blastoff group might decide to start the requirements investigation with the intention of reviewing the requirements after a short while and reassessing the value of the project.

Trawling for Requirements

Once the blastoff is completed, the business analysts start trawling the work to learn and understand its functionality—“What’s going on with this piece of the business, and what do they want it to do?” For convenience and consistency, they partition the work context diagram into business use cases.

Each business use case is an amount of functionality needed by the work to make the correct response to a business event. (These terms will be fully explained soon.) A requirements analyst is assigned to each of the business use cases—the analysts can work almost independently of one another—for further detailed study. The analysts use trawling techniques such as apprenticing, scenarios, use case workshops, and many others to discover the true nature of the work. These trawling techniques are described in Chapter 5, Investigating the Work.

Trawling means discovering the requirements. The business analysts sit with the IceBreaker technicians as they describe the work they currently do, and their aspirations for work they hope to do. The business analysts

---

**Reading**


Refer to Chapter 4 for a discussion of business events and business use cases, and an exploration of how you might use them.

Refer to Chapter 5, Investigating the Work, for details of the trawling activity.
also consult with other interested stakeholders and subject-matter experts—
experts on usability, security, operations, management, and so on—to dis-
cover other needs for the eventual product. The IceBreaker business analysts
spent a lot of time with the meteorologists and the highway engineers.

Perhaps the most difficult part of requirements investigation is uncover-
ing the essence of the system. Many stakeholders inevitably talk about their
perceived solution to the problem or express their needs in terms of the cur-
rent implementation. The essence, by contrast, is the underlying business
reason for having the product. Alternatively, you can think of it as the policy
of the work, or what the work or the business rule would be if it could exist
without any technology (and that includes people). We will have more to
say about the essence of the system in Chapter 7, Understanding the Real
Problem.

Once they understand the essence of the work, the analysts get together
with the key stakeholders to decide the best product to improve this work.
That is, they determine how much of the work to automate or change, and
what effect those decisions will have on the work. Once they know the
extent of the product, the requirements analysts write its requirements. We
illustrate this process in Figure 2.3.

The IceBreaker product must not be a simplistic automation of the work as
it is currently done; the best of our automated products are not mere imita-
tions of an existing situation. To deliver a truly useful product, the analyti-
cal team must work with the stakeholders to innovate—that is, to develop
a better way to do the work, and a product that supports this better way of
working. They make use of innovation workshops where the team uses cre-
ative thinking techniques and innovative triggers to generate new and better
ideas for the work and the eventual product.

![Diagram of the Requirements Process]

**Figure 2.3**

The blastoff determines the scope of the work to be improved. The
business use cases are derived from the scope. Each of the business
use cases is studied by the requirements analysts and the
relevant stakeholders to discover the desired
way of working. When this is understood, the
appropriate product can be determined
(the PUC scenario) and
requirements or user
stories written from it.

**READING**


We look at developing innovative products in Chapter 8, Starting the Solution.
Quick and Dirty Modeling

Models can be used at any time in the Volere life cycle; in Figure 2.1, we show this activity as “Prototype the Work.” There are, of course, formal models such as you would find in UML or BPMN, but a lot of the time business analysts can make productive use of quick sketches and diagrams to model the work being investigated. One quick and dirty modeling technique we should mention here is using Post-it notes to model functionality; each note can be used to represent an activity, and the notes can be rapidly rearranged to show different ways the work is done or could be done. We find that stakeholders relate to this way of modeling their business processes, and are always willing to participate with hands-on manipulation of the Post-its to show what they think the work should be. We discuss this kind of modeling more fully in Chapter 5, Investigating the Work.

In Chapter 8, Starting the Solution, we examine how you move into an implementation of the requirements discovered so far. At this point, your models change from being something to explain the current work, to something to explain how the future product will help with that work.

We can now start to refer to this type of model as a prototype—a quick and dirty representation of a potential product using pencil and paper, whiteboards, or some other familiar means, as shown in Figure 2.4. Prototypes used at this stage are intended to present the user with a simulation of the requirements as they might be implemented. The IceBreaker business analysts sketch some proposed interfaces and ways that the needed functionality might be implemented—this visual way of working allows the engineers and other stakeholders to coalesce their ideas for the future product.

Figure 2.4
A quick and dirty prototype built on a whiteboard to provide a rapid visual explanation of how some of the requirements might be implemented, and to clarify misunderstood or missing requirements.
Scenarios

Scenarios are so useful that we have devoted the whole of Chapter 6 to them. Scenarios show the functionality of a business process by breaking it into a series of easily recognizable steps, written in English (or whatever language you use at work) so that they are accessible to all stakeholders. The IceBreaker analysts used scenarios to describe the business processes and present their understanding of the needed functionality. These scenarios were then revised as needed—different stakeholders took an interest in different parts of the scenario, and after a short time, the business analysts were able to have everyone understand and come to a consensus on what the work was to be.

Once they are agreed, the scenarios become the foundation for the requirements.

Writing the Requirements

A major problem in system development is misunderstood requirements. To avoid any misunderstanding, the analysts must write their requirements in an unambiguous and testable manner, and at the same time ensure that the originating stakeholder understands and agrees with the written requirement before it is passed on to the developers. In other words, the analysts write the requirements so as to ensure that parties at either end of the development spectrum are able to have an identical understanding of what is needed.

Although the task of writing down the requirements might seem an onerous burden, we have found it to be the most effective way to ensure that the essence of the requirement has been captured and communicated, and that the delivered product can be tested. (See Figure 2.5.)
The IceBreaker analysts start by writing their requirements using business language so that the nontechnical stakeholders can understand them and verify their correctness. They add a rationale to the requirements—it shows the background reason for the requirement, which removes much of the ambiguity. Further, to ensure complete precision and to confirm that the product designers and developers can build exactly what the stakeholder needs, they write a fit criterion for each requirement. A fit criterion quantifies, or measures, the requirement, which makes it testable, which in turn allows the testers to determine whether an implementation meets—in other words, fits—the requirement.

The rationale and the fit criterion make the requirement more understandable for the business stakeholder, who has on several occasions said, “I am not going to have any requirements that I do not understand, nor will I have any that are not useful or that don’t contribute to my work. I want to understand the contributions that they make. That’s why I want each one to be both justified and measurable.”

The business analyst has a different, but complementary, reason for measuring requirements: “I need to ensure that each requirement is unambiguous; that is, it must have the same meaning to both the stakeholder who originated it and the developer who will build it. I also need to measure the requirement against the stakeholder’s expectations. If I can’t put a measurement to it, then I can never tell if we are building the product the stakeholder really needs.”

The analysts use two devices to make it easier to write their specification. The first device, the requirements specification template, is an outline and guide to writing a requirements specification. The business analysts use it as a checklist of the requirements they should be asking for, and as a consistent way of organizing their requirements documents. The second device is a shell, also known as a snow card. Each atomic (that’s the lowest level) requirement is made up of a number of attributes, and the snow card is a convenient layout for ensuring that each requirement has the correct constituents.

Of course, the writing process is not really a separate activity. In reality, it is integrated with the activities that surround it—trawling, prototyping, and the quality gateway. However, for the purposes of understanding what is involved in putting the correct requirements into a communicable form, we will look at it separately.

Iterative development methods employ user stories as a way of conveying the requirements. The stories are, in fact, placeholders for lower-level requirements; they are augmented during conversations between the developers and the stakeholders to flush out the detailed requirements. In Chapter 14, Requirements and Iterative Development, we look closely at how the business analyst can produce better user stories. Working iteratively does not obviate the need for requirements, but rather seeks to discover and communicate the requirements in a different manner.
The primary reason for wanting written requirements is not to *have* written requirements (although that is often necessary), but rather to *write* them. Writing the requirement, together with its associated rationale and fit criterion, clarifies it in the writer’s mind, and sets it down in an unambiguous and verifiable manner. To put that another way, if the business analyst cannot correctly write the requirement, he has not yet understood it.

**Quality Gateway**

Requirements are the foundation for all that is to follow in the product development cycle. Thus it stands to reason that if the right product is to be built, the requirements must be correct before they are handed over to the builders. To ensure correctness, the quality gateway tests the requirements (Figure 2.6). The IceBreaker team has set up a single point that every requirement must pass through before it can become a part of the specification. This gateway is manned by two people—the lead requirements analyst and a tester—and they are the only people authorized to pass requirements through the gateway. Working together, they check each requirement for completeness, relevance, testability, coherency, traceability, and several other qualities before they allow it to be passed to the developers.

By ensuring that the only way for requirements to be made available for the developers is for those requirements to pass through the quality gateway, the project team is in control of the requirements, and not the other way around.

*Figure 2.6*

The quality gateway ensures that requirements are rigorous by testing each one for completeness, correctness, measurability, absence of ambiguity, and several other attributes, before allowing the requirement to be passed to the developers.
Reusing Requirements

The requirements for any product you build are never completely unique. We suggest that before starting on any new requirements project, you go through the specifications written for previous projects and look for potentially reusable material. Sometimes you may find dozens of requirements that you can reuse without alteration. More often you will find requirements that, although they are not exactly what you want, are suitable as the basis for some of the requirements you will write in the new project.

For example, in the IceBreaker project, the rules for road engineering have not changed much over the years. Thus, the requirements analysts working on various projects do not have to rediscover them, but can simply reuse them. They also know that the business of vehicle scheduling does not change radically over time, so their trawling process can take advantage of some requirements from previous projects.

Similarly, for different projects within your organization, the non-functional requirements are fairly standard, so you can start with a specification from one of the previous projects and use it as a checklist.

The point about reusing requirements is that once a requirement has been successfully specified for a product, and the product itself is successful, the requirement does not have to be reinvented or rediscovered. In Chapter 15, Reusing Requirements, we discuss how you can take advantage of the knowledge that already exists within your organization, and how you can save yourself time by recycling requirements from previous projects.

Reviewing the Requirements

The quality gateway exists to keep bad requirements out of the specification—it does this one requirement at a time. Nevertheless, at the point when you think your requirements specification is complete (or as complete as you need it for the next activity), you should review it. This final review checks that there are no missing requirements, that all the requirements are consistent with one another, and that any conflicts between the requirements have been resolved. In short, the review confirms that the specification is really complete and suitable so that you can move on to the next stage of development.

This review also offers you an opportunity to reassess the costs and risks of the project. Now that you have a complete set of requirements, you know a lot more about the product than you did at the project blastoff. In particular, you have a much more precise knowledge of the scope and functionality of the product, so this is a good time to remeasure its size. From that size, and from your knowledge of the project’s constraints and solution architecture, you can estimate the cost to construct the product.
You also know at this stage which types of requirements are associated with the greatest risks. For example, the users might have asked for an interface that your organization has not built before. Or perhaps they want to use untried technology to build the product. Perhaps the developer might not have the people with the skills needed to build the product as specified? By reassessing the risks at this point, you give yourself a more realistic chance of building the desired product successfully.

**Iterative and Incremental Processes**

One common misconception in the requirements world is that you have to gather all the requirements before moving on to the next step of design and construction. In other words, doing requirements means that you employ a traditional waterfall process. In some circumstances this is necessary, but not always. On the one hand, if you are outsourcing or if the requirements document forms the basis of a contract, then clearly you need to have a complete requirements specification. On the other hand, if the overall architecture is known, then construction and delivery can often begin before all the requirements are discovered. We show these two approaches in Figure 2.7, and suggest you consider which one works best for you when working on your own requirements projects. We also have a lot more to say on various approaches in Chapter 9, Strategies for Today’s Business Analyst.

On the IceBreaker project, the developers are ready to start building the product, so after the blastoff the key stakeholders select three (it could be any low number) of the highest-priority and greatest-value business use cases. The requirements analysts trawl and gather the requirements for only those

---

**Figure 2.7**

Two (of many) variations on development life cycles. At the top of the figure is the traditional waterfall approach, in which the complete requirements document is put together before product development begins. At the bottom of the figure is an iterative process, in which, after a preliminary analysis, the product is developed in small increments. Both approaches achieve the same purpose.
business use cases, putting aside the rest of the work for now. Then, when the first tranche of requirements have successfully passed the quality gateway, the developers start their work. The intention is to implement a small number of use cases as early as possible to get the reaction of the stakeholders—if there are going to be any nasty surprises, the IceBreaker team wants to get them as early as possible. While the developers are building and delivering the first lot of business use cases, the analysts are working on the requirements for the next-highest-priority ones. Soon they have established a rhythm for delivery, with new use cases being implemented and delivered every few weeks.

Requirements Retrospective

You are reading this book about a requirements process, presumably with the intention of improving your own process. Retrospectives, sometimes known as lessons learned, are one of the most effective tools for discovering the good and bad of a process, and suggesting remedial action. Retrospectives for requirements projects consist of a series of interviews with stakeholders and group sessions with the developers. The intention is to canvas all the people involved in the project and ask these questions:

- What did we do right?
- What did we do wrong?
- If we had to do it again, what would we do differently?

By looking for honest answers to these questions, you give yourself the best chance of improving your process. The idea is very simple: Do more of what works and less of what doesn’t.

Keep a record of the lessons learned from your retrospectives. While humans have memory and can learn from their experience to their advantage in future projects, organizations don’t learn—unless you write down the experience. By keeping the lessons learned available in some readily accessible manner, subsequent projects can learn from your accomplishments and mishaps.

Your retrospective can be very informal: a coffee-time meeting with the project group, or the project leader collecting e-mail messages from the participants. Alternatively, if the stakes are higher, this process can be formalized to the point where it is run by an outside facilitator who canvases the participants, both individually and as a group, and publishes a retrospective report.

The most notable feature of retrospectives is this: Companies that regularly conduct retrospectives consistently report significant improvements in their processes. In short, retrospectives are probably the cheapest investment you can make in improving your own process.

“If we did the project again tomorrow, what would we do differently?”
Chapter 2  The Requirements Process

Evolution of Requirements

You start a project with little more than a vision—and sometimes a fairly blurred vision—of the desired future state of your owner’s work. (As we have done elsewhere in this book, we use the term “work” to refer to the area of the owner’s organization where improvements are to be made, usually by automating or re-automating part of it.)

During the early stages of requirements discovery, analysts deploy models of varying degrees of formality to help them and the stakeholders to learn what the work is, and what it is to be. From this investigation of the work, everyone arrives at the same level of understanding such that the stakeholders find improvements that will be truly beneficial.

It helps enormously when coming to an understanding of the work if the analysts and stakeholders can see the essence of the work. The essence is an abstraction of the work that sees the underlying policy of the work without the technology that clouds our vision of what the work actually is. This “thinking above the line,” as we call it in Chapter 7, Understanding the Real Problem, is important if the requirements are not to merely replicate whatever it is that exists at the moment, and if “technological fossils” and inappropriate process are not to be inadvertently reimplemented.

The understanding of the work evolves and matures, and at some point it is possible for the stakeholders, guided by the business analysts and the systems architects, to determine the optimal product to improve that work. When this stage is reached, the business analysts determine the detailed functionality for the product (keep in mind that not all of the work’s functionality would be included in the product) and to write its requirements. The non-functional requirements are derived at roughly the same time and written along with those constraints that are not already recorded. At this point, the requirements are written in a technologically neutral manner—they specify what the product has to do for the work, but not how the technology will do it.

You can think of these requirements as “business requirements,” meaning that they specify the product needed to support the business. Once they are adequately understood, they are released to the designer, who adds the product’s technological requirements before producing the final specification for the builders. This process is illustrated in Figure 2.8.

We have said that the requirements evolve, but this process should not be thought of as an inexorable progression toward some known destination. As Earl Beede points out, every time you think of a solution, it causes some new problems that require you to backtrack and revisit some of your earlier work. When we are talking about a requirements process, keep in mind that the process, if it is to be useful, must allow you to move backward as well as forward. Naturally, you would like to spend most of your time moving
forward, but don’t be too disappointed if you have to return to some things you thought you had put behind you.

The Template

It is easier to write requirements, and far more convenient, if you have a guide to writing them. Appendix A of this book provides The Volere Requirements Specification Template, which is a complete blueprint for describing your product’s functionality and capabilities. This template, which is a distillation of literally hundreds of requirements specifications, is in use by thousands of organizations all over the world.

It is convenient to categorize requirements into several types—each of the template’s sections describes a type of requirement and its variations. Thus, as you discover the requirements with your stakeholders, you add them to your specification, using the template as a guide to necessary content.

The template is designed to serve as a sophisticated checklist, providing you with a list of what to write about, and suggestions on how to write about them. The table of contents for the template is reproduced here, and we will discuss each section in detail later in the book.

Our associate, Stephen Mellor, suggests using the template by going directly to the most pressing sections—the ones that seem to you to be most
useful—and then revisiting the template as needed. You will probably use most of it, but it is not—really not—a template that you fill by starting on page one and working through to the bitter end. Like any good tool, when used wisely the template provides a significant advantage to your requirements discovery.

Here, then, is the content of the template.

**Project Drivers**—reasons and motivators for the project

1. **The Purpose of the Project**—the reason for making the investment in building the product and the business advantage that you want to achieve by doing so
2. **The Client, the Customer, and Other Stakeholders**—the people with an interest in or an influence on the product
3. **Users of the Product**—the intended end users, and how they affect the product’s usability

**Project Constraints**—the restrictions on the project and the product

4. **Requirements Constraints**—the limitations on the project, and the restrictions on the design of the product
5. **Naming Conventions and Definitions**—the vocabulary of the project
6. **Relevant Facts and Assumptions**—outside influences that make some difference to this product, or assumptions that the developers are making

**Functional Requirements**—the functionality of the product

7. **The Scope of the Work**—the business area or domain under study
8. **The Scope of the Product**—a definition of the intended product boundaries and the product’s connections to adjacent systems
9. **Functional and Data Requirements**—the things the product must do and the data manipulated by the functions

**Non-functional Requirements**—the product’s qualities

10. **Look and Feel Requirements**—the intended appearance
11. **Usability and Humanity Requirements**—what the product has to be if it is to be successfully used by its intended audience
12. **Performance Requirements**—how fast, big, accurate, safe, reliable, robust, scalable, and long-lasting, and what capacity
13 **Operational and Environmental Requirements**—the product’s intended operating environment
14 **Maintainability and Support Requirements**—how changeable the product must be and what support is needed
15 **Security Requirements**—the security, confidentiality, and integrity of the product
16 **Cultural Requirements**—human and sociological factors
17 **Legal Requirements**—conformance to applicable laws

**Project Issues**—issues relevant to the project that builds the product
18 **Open Issues**—as yet unresolved issues with a possible bearing on the success of the product
19 **Off-the-Shelf Solutions**—ready-made components that might be used instead of building something from scratch
20 **New Problems**—problems caused by the introduction of the new product
21 **Tasks**—things to be done to bring the product into production
22 **Migration to the New Product**—tasks to convert from existing systems
23 **Risks**—the risks that the project is most likely to incur
24 **Costs**—early estimates of the cost or effort needed to build the product
25 **User Documentation**—the plan for building the user instructions and documentation
26 **Waiting Room**—requirements that might be included in future releases of the product
27 **Ideas for Solutions**—design ideas that we do not want to lose

Browse through the template in Appendix A before you go too much further in this book. You will find a lot of information about writing requirements, plus much food for thought about the kinds of requirements you are looking for.

Throughout this book, we will refer to requirements by their type—that is, one of the types as shown in the template’s table of contents.

**The Snow Card**

Whereas the template is a guide to *what* to write about, the snow card is a guide to *how* to write it. Individual requirements have a structure—a set of attributes, where each attribute contributes something to your understanding
of the requirement, and to the precision of the requirement, and thereby to the accuracy of the product’s development.

Before we go any further, we must point out that although we call this device a card, and we use cards in our courses, and this book is sprinkled with diagrams featuring this card, we are not advocating writing all your requirements on cards. Some good things can be realized by using cards when interviewing stakeholders and quickly scribbling requirements as they come to light. Later, these requirements are recorded in some electronic form; at that time, their component information is filled in. Thus any reference to “card” should be taken to mean (probably) a computerized version.

At first glance, the card might seem rather bureaucratic. (See Figure 2.9.) We are not seeking to add to your requirements burden, but rather to provide a way of accurately and conveniently gathering the needed information—each of the attributes of the snow card makes a contribution. We shall explain these as we work our way through this book.

Figure 2.9
The requirements shell or snow card, consisting of a 5-inch by 8-inch card, printed with the requirement’s attributes, that is used for our initial requirements gathering. Each of the attributes contributes to the understanding and testability of the requirement. Although a copyright notice appears on the card, we have no objections to any reader making use of it for his or her requirements work, provided the source is acknowledged.
Your Own Requirements Process

The itinerant peddler of quack potions, Doctor Dulcamara, sings the praises of his elixir—it is guaranteed to cure toothache, make you potent, eliminate wrinkles and give you smooth beautiful skin, destroy mice and bugs, and make the object of your affections fall in love with you. This rather fanciful libretto from Donizetti’s opera *L’elisir d’amore* points out something that, although very obvious, is often disregarded: There is no such thing as the universal cure.

We really would like to be able to present you with a requirements process that has all the attributes of Doctor Dulcamara’s elixir—a process that suits all projects for all applications in all organizations. We can’t. We know from experience that every project has different needs. However, we also know that some fundamental principles hold good for any project. So instead of attempting to provide you with a one-size-fits-all magic potion, we have distilled our experiences from a wide variety of projects to provide you with a set of foundation activities and deliverables that apply to any project.

The process described in this book is made up of the things you have to do to successfully discover the requirements. Likewise, the deliverables presented here are the foundation for any kind of requirements activity. Our intention is not to say that there is only one true path to requirements Nirvana, but rather to give you the components you need for successful requirements projects.

As you read this book, think about how you can use these components within the constraints of your own culture, your own environment, your own organizational structure, and your own chosen way of product development.

To adapt this process, you should understand the deliverables it produces—the rest of this book will discuss these items in detail. Once you understand the content and purpose of each deliverable, ask how each one (provided it is relevant) would best be produced within your project environment using your resources:

- What is the deliverable called within your environment? Use the definitions of the terms used in the generic process model and identify the equivalent deliverable in your organization.
- Is this deliverable relevant for this project?
- How much do you already know about this deliverable? Do you know enough to be able to avoid devoting additional time to it?
- Who produces the deliverable? Understand which parts of the deliverable are produced by whom. Also, when several people are involved, you need to define the interfaces between them.
- When is the deliverable produced? Map your project phases to the generic process.

---

**Reading**

Where is the deliverable produced? A generic deliverable is often the result of fragments that are produced in a number of geographical locations. Define the interfaces between the different locations and specify how they will work.

Who needs to review the deliverable? Look for existing cultural checkpoints within your organization. Do you have recognized stages or phases in your projects at which peers, users, or managers must review your specification?

The generic model describes deliverables and procedures for producing them; our intention is that you decide how you use them.

We also point you to Chapter 9 of this book, entitled Strategies for Today’s Business Analyst. This chapter considers how you might approach your requirements projects. We suggest that before you become too involved in the mechanics of requirements discovery, you think about the strategy that is most suitable for you.

Formality Guide

There is every reason to make your requirements discovery and communication as informal as possible. We say “as possible” because it is not so much what you would like as what your situation demands—often the degree of formality will be dictated by factors beyond your control. For example, you may be developing software using contracted outsourced development. In this case, there is a clear need for a complete written requirements specification. In other cases, the way you communicate your requirements can be informal to the point that a portion of the requirements are not written, or partially written, and communicated verbally.

We have included a formality guide to suggest where you might take a more relaxed approach to recording requirements, as well as those times when you should rightly be more systematic with your requirements discovery and communication. These are the conventions you will encounter as you move through this book.

Rabbit—small, fast, and short-lived. Rabbit projects are typically smaller projects with shorter lifetimes, where close stakeholder participation is possible. Rabbit projects usually include a lesser number of stakeholders.

Rabbit projects are usually iterative. They discover requirements in small units (probably one business use case at a time) and then implement a small increment to the working functionality, using whatever has been implemented to solicit feedback from the stakeholders.

Rabbit projects do not spend a great deal of time writing the requirements, but use conversations with the stakeholders as a way to elaborate
the requirements written on story cards. Rabbit projects almost always co-locate the business knowledge stakeholders with the business analysts and the developers.

**Horse—fast, strong, and dependable.** Horse projects are probably the most common corporate projects—they are the “halfway house” of formality. Horse projects need some formality—it is likely that there is a need for written requirements so that they can be handed from one department to another. Horse projects have medium longevity and involve more than a dozen stakeholders, often in several locations, factors that necessitate consistently written documentation.

If you cannot categorize your own project, think of it as a horse.

**Elephant—solid, strong, long life, and a long memory.** An elephant project has a need for a complete requirements specification. If you are outsourcing the work, or if your organizational structure requires complete, written specifications, you’re an elephant. In certain industries, such as pharmaceuticals, aircraft manufacture, or the military, regulators demand not only that full specifications be produced, but also that the process used to produce them be documented and auditable. Elephant projects typically have a long duration, and they involve many stakeholders in distributed locations. There are also a large number of developers, necessitating more formal ways of communicating.

**The Rest of This Book**

We have described—briefly—a process for discovering, communicating, and verifying requirements. The remainder of this book describes the various activities in this process, along with their deliverables, in some detail. Feel free to jump to any chapter that is of immediate concern—we wrote the chapters in more or less the order in which you would do each of the activities, but you don’t have to read them that way.

And please, while you are reading this book, be constantly asking yourself how you will do the things we describe. After all, it is you who has to do them.

We hope find useful ideas, processes and artifacts, in the rest of this book. We also hope you enjoy reading and using it.
This page intentionally left blank
Index

A
Abbreviations, 415–416
Abstraction
   Brown Cow Model, 153–154
   patterns from, 346–351
   problem determination, 147–149
   for requirements, 316, 342
   reusable requirements, 106–107
   in trawling, 99, 125–126
Acceptance, usability for, 253
Access requirements, 263, 451–452
Accessibility requirements
   fit criteria for, 292
   in usability, 256, 441
Accuracy
   patterns for, 342
   requirements, 258, 443–444
Achievable goals, 57
Acronyms, 415–416
Actions
   fit criteria, 297–298
   functional requirements, 295–296
Active adjacent systems, 190–192
Active stakeholders, 134, 144
Activities in strategies, 204
Activity diagrams
   functional requirements, 240
   scenarios, 138–139
Actors
   business use cases, 70, 84
   operational requirements, 260
Adaptability requirements, 450–451
Addiction to connections, 185
Adjacent systems, 190
   active, 190–192
   autonomous, 192–193
   in business use cases, 71
   cooperative, 193–194
   in function point counting, 490
   interfacing with, 447–448
   legal requirements, 269
   in operational requirements, 260, 447–448
   and scope, 43
   as stakeholders, 53–54
Adjectives, 388
Adjustments in function point counting, 492
Adobe Photoshop usability, 254
Adoption, usability for, 256
Advantage in PAM technique, 399–400
Advantages, Limitations, Unique Quantities
   and overcome (ALUo) management technique, 64
Adverbs, 388
Affordable Care Act, 270
Aggregation in systemic thinking, 162
Agile techniques
   iterative development, 323–324
   in problem determination, 153
Air traffic control systems, 172
Airlines
   cargo, 260–261
   check-in agent scenario, 131–140
   Alexander, Christopher, 281, 343–344
   Alexander, Ian, 44, 142
   Alfresco system, 115
   Allowable values, requirements for, 258
Alternatives
   functional requirements, 233–234, 238–241
   Quality Gateways, 320–321
   scenarios for, 139–140, 145
ALUo (Advantages, Limitations, Unique Quantities and overcome) management technique, 64
Amazon
   1-click feature, 233
   convenience, 161
   future of books, 158–159
   non-functional requirements, 246
Ambiguity

- in functional requirements, 234–237
- reviews for, 388

Analysis Artifacts activity, 325
Analysis Backlog activity, 325
Analysts

- apprenticeships for, 98–99
- for scope, 16
- for trawling, 91–92
- writing by, 20

Analytical knowledge in iterative development, 334
Analyze Business Needs activity, 324–325
“And no more” requirements, 265–266
Antagonists in negative scenarios, 142–143
Anticipated environments

- constraints from, 412
- for new products, 461–462
Appearance requirements, 435
Apprentices, 17, 98–99
ART-SCENE scenario presenter, 141
Artifacts

- in apprentices, 99
- in domain patterns, 350
- in functional requirements, 225
- murder books, 120
- in prototyping, 111
- retaining, 124
- for stakeholder interviews, 103

Asimov, Isaac, 258
Assembling specification templates, 365–366
Associations, 495, 505–508
Assumptions

- in blastoff, 37
- constraints as, 169
- in reusing requirements, 339
- risk analysis, 390
- in specification templates, 418–419
- of usability, 253
Atomic Requirement knowledge class, 496–497
Atomic requirements

- attributes, 361–365
- discovering, 359–361
- external strategy, 208–209
- in functional requirements, 238
- prioritizing, 383

“Atending exquisitely,” 106
Attributes

- atomic requirements, 361–365
- business use cases, 378–380, 489
- classes, 483
- completeness tests for, 311–312
- stored data, 489–491
- user categories, 49–50

Auditing requirements, 265, 273, 454
Authority, 295–296
Authorization, 263
Automated tools

- for Quality Gateway, 320
- for scenarios, 141
- specification templates, 366–367
- Autonomous adjacent systems, 192–193
Availability requirements, 258, 263, 444–445

B

Babbage, Charles, 4
Background in specification templates, 397–398
Baker, Jenny, 341
Bang measuring method, 481
Beck, Kent, 280
Beede, Earl, 26, 196, 277
Benchmarks, 61
Benefits in solutions, 194–195
Beyer, Hugh, 98, 113, 131
Blastoff, 15–17, 35–37

- constraints in, 37
- costs, 17
- go/no go decisions, 17, 37
- meetings, 64–65
- naming conventions and definitions, 37
- purpose determination, 36
- risks, 37
- scope, 36
- stakeholders, 37
- for trawling, 92
Blogs

- for non-functional requirements, 271
- for trawling, 122
Book selling, 158–159
Boundaries

- product, 180–181
- scope, 429–431
BPMN (Business Process Modeling Notation), 139
Brainstorming

- overview, 173–174
- videos for, 121
Branding standards

- company colors for, 290
- in look and feel requirements, 251
Breakout conditions

- external strategy, 206–209
- knowledge requirements for, 205–206
Brokers, idea, 219–220
Brooks, Fred, 8
Brown Cow Model, 149–150

- abstraction, 153–154
- essence, 150–153
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>525</td>
</tr>
</tbody>
</table>

- future view, 157–160, 174–175
- for interviews, 105
- overview, 93–97
- solutions, 177–178
- swim lanes, 154–156

**Buddy pairing approach**, 321

**Budgets**
- as constraints, 414
- requirements creep from, 318–319

**Building activity**
- external strategy, 208–209
- iterative strategy, 214

**Business analysts for trawling**, 91–92

**Business boundary association**, 505

**Business data models**
- in risk analysis, 390
- in specification templates, 427

**Business Event knowledge class**, 497–498

**Business events**
- atomic requirements, 361–362
- benefits, 75–78
- business use cases, 73–80
- for cost estimates, 61
- finding, 78–80
- identifying, 377–378
- innovation workshops, 172
- iterative development, 324–325
- origins, 189–190
- patterns, 344–346
- prioritizing, 217
- product use case, 197
- in scenario templates, 144
- time-triggered, 74–75
- video, 121
- in work partitioning, 422–423

**Business knowledge in iterative development**, 333–334

**Business Process Modeling Notation (BPMN)**, 139

**Business process models**, 240–241

**Business relevancy association**, 356, 506

**Business requirements**, 7–8, 26

**Business responding association**, 506

**Business rules**
- business use case workshops, 101–102
- looking for, 218–219
- maintainability requirements, 261
- specification templates, 417–418

**Business tolerances**
- for fit criteria, 284–285
- in subjective tests, 289

**Business tracing association**, 506

**Business Use Case knowledge class**, 498–499

**Business use cases (BUCs)**, 67
- actors, 84
- adjacent systems in, 71
- atomic requirements, 361–362
- benefits, 75–78
- business rules, 219
- completeness reviews for, 376–382
- CRUD check for, 380–381
- custodial processes, 381–382
- data for, 378–380
- events, 73–82, 377–378
- formality guide for, 69
- in function point counting, 483–488
- functional needs, 179
- input, 484–485
- iterative strategy, 211, 327–328, 382
- low-fidelity prototypes for, 112
- modeling, 378–380
- output, 485–487
- outside world in, 72–73
- patterns, 342
- product use cases in, 82–84
- and scenarios, 130–131, 133, 144
- scope, 70–73, 82–83, 375, 377
- time-triggered, 487–488
- in trawling, 17, 92
- user stories, 329–330
- in value, 166
- videos for, 121
- work in, 67–69
- workshops, 99–102, 121

**Business value analysis**, 327–328

**C**

- Cameras, 161

**Capabilities**
- document archeology for, 124
- templates for, 27

**Capacity requirements**, 257, 445–446

**Cargo airlines**, 260–261

**Case study in scoping business problem**, 41–43

**CATWOE (Customers, Actors, Transformation processes, World view, Owners, and Environment) management technique**, 64

**Challenging constraints**, 169–171

**Change**, requirements creep from, 319

**Character of products**, 248

**Check-in agents**, 131–140

**Checklists**
- completeness reviews, 374
- exceptions, 141

**Quality Gateways**, 321
Checklists (continued)
requirement types, 249
specifications, 21
templates as, 27, 247, 274
users, 49–50
Chesterton, G. K., 127
Choices, scenarios for, 139–140
Christensen, Clayton, 159
Class diagrams, 231–232
Classes
attributes, 483
for business use cases, 379–380
Volere Requirements Process Model, 495
Clausing, Don, 180
Clients
non-functional requirements, 275–276
reusing requirements, 339
risk analysis, 389
specification templates, 400
Collaborating systems and applications
constraints from, 410
fit criteria for, 293
in operational requirements, 261
Collections of requirements, 343–344
Color
in branding, 290
measuring, 285
in mind maps, 118
Commercial off-the-shelf software
as constraint, 59, 410–411
functional requirements for, 241–242
in specification templates, 458–460
Communicating requirements, 20–22, 353
formality guide, 353–354
knowledge vs. specification, 353–357
potential requirements, 354
Communication knowledge in iterative
development, 334
Company colors, 290
Completeness requirements, 23–24, 371–372
for ambiguity, 388
business use cases, 375–382
for conflicts, 386–388
cost measurements, 391
formality guide, 372–373
inspections for, 373–374
missing requirements, 374–375
patterns, 342
prioritizing requirements, 382–386
risk assessment, 388–390
specifications, 373
testing, 311–312
Completion of actions, 295
Compliance requirements, 455–456
Conception activity
external strategy, 206–207
iterative strategy, 210, 213
Conditional functional requirements, 234
Conditions in fit criteria, 297–298
Conflicts
atomic requirements, 364–365
completeness reviews for, 386–388
Connections, innovation, 185–186
Connelly, Michael, 119
Consistency in terminology, 313–314
Constraint knowledge class, 499
Constraints, 11, 59–60
blastoff, 37
challenging, 169–171
from environment, 412
fit criteria for, 300
mandated, 339, 390, 407–415
off-the-shelf products, 59, 410–411
project, 60
in reusing requirements, 339
in risk analysis, 390
in scenarios, 135
solutions, 59, 200
in specification templates, 28, 358–359,
407–415
viability within, 314–315
Construction activity
external strategy, 209
iterative strategy, 212
Consultants
for security, 266
as stakeholders, 51
Containing businesses, 45
Content management systems, 115–116
Context
business use cases, 70–72
event responses, 344–345
patterns, 344–345
process in, 14–15
scope, 42–43, 420–421
stakeholder interviews, 103
Context diagrams, 16, 41–43
business events, 78–79
flows, 482
functional requirements, 242
Context flows in Quality Gateways, 307–308
Convenience
innovation, 184–185
paying for, 160–162
Cooper, Alan, 167
Cooperative adjacent systems
in function point counting, 490
overview, 193–194
Copyable products, 459–460
Copyright notices, 269
Core teams
  in stakeholder maps, 45
  as stakeholders, 51–52
Costs
  blastoff, 17, 37
  error repair, 306
  review process, 23
  scoping, 61–62
  solutions, 194–195
  specification templates, 467–468
  value, 165–166
Create step in CRUD checks, 380–381
Creativity in brainstorming, 174
Creep, requirements, 317–319
CRUD checks, 380–381
Cultural issues and requirements
  fit criteria, 294–295
  overview, 266–268
  product use case, 273
  specification templates, 454–455
  stakeholders, 53
Current situation and environment
  implementation environment, 409
  new products, 460
  in scope, 420
  in trawling, 94–97
Custodial processes, 381–382
Customer recognition, company colors for, 290
Customer satisfaction
  atomic requirements, 363–364
  Quality Gateway for, 316–317
Customers
  expression of requirements, 6–7
  reusing requirements, 339
  risk analysis, 389
  specification templates, 401
  as stakeholders, 47–48
  value and satisfaction of, 316–317
Customers, Actors, Transformation processes, World view, Owners, and Environment (CATWOE) management technique, 64
Customs in cultural requirements, 266–268, 454–455

D
Data
  business event patterns, 345–346
  business use cases, 379–380
  in functional requirements, 231–233
  models, 231–232
  new systems, 465
  stored in function point counting, 482–483, 489–492
Data definitions in fit criteria, 297
Data dictionaries
  in functional requirements, 232–233
  risk analysis, 390
  specification templates, 416, 427–429
Data element types, 488
Data flows in Quality Gateways, 307–308
Data models
  risk analysis, 390
  specification templates, 425–427
Data Protection Act, 271
Data requirements in specification templates, 433–434
David, Elizabeth, 341
Dead Fish projects, 63–64
Decibels, 285
Decision tables in fit criteria, 297–298
Decisions in activity diagrams, 139
Definitions
  requirements, 211–212
  reusing requirements, 339
  scoping, 60–61
  specification templates, 415–416
DeGrace, Peter, 121
Delete step in CRUD checks, 380–381
Deliverables
  blastoff for, 36
  understanding, 31–32
DeMarco, Tom, 479, 481
Descriptions
  ambiguous, 388
  atomic requirements, 362
  and fit criteria, 283
  in functional requirements, 229–231, 237
  vs. measurements, 279
Design decisions, documenting, 195–196
Designing user experience, 183–184
Detail in functional requirements, 228–229
Development Backlog activity, 326–327
Development phases in planning, 463
Deviations, exception cases for, 140–141
Diagrams
  for business events, 78–79
  class, 231–232
  context. See Context diagrams
  for functional requirements, 240, 242
  mind maps, 116–119
  for scenarios, 138–139
  trawling, 92
  use case, 483
Dictionaries
  in functional requirements, 232–233
  risk analysis, 390
  in specification templates, 415–416, 427–429
Differentiation in solutions, 200
Discretionary money, 160
Discussion forums for trawling, 122
Dissatisfaction rating, 363–364
Divisions in specification templates, 358–359
Document archeology, 123–124
Documentation
  design decisions, 195–196
  murder books, 119–120
  in specification templates, 468–469
Dodd-Frank Wall Street Reform and Consumer Protection Act, 270
Domains, 341
  models, 342
  patterns across, 349–351
  patterns for, 348–349
  in reusing requirements, 351
Downloadable movies, 148
Drivers
  risk analysis, 389–390
  specification templates, 28, 357–359, 395
Drupal system, 115

E
Ease of use requirements, 254, 437–438
Easy to learn products, 254, 291
EEML (Extended Enterprise Modeling Language), 399–400
Effects of Quality Gateway, 304–305
Efficiency
  requirements, 258
  usability for, 253
Effort, estimating, 61–62
Einstein, Albert, 150, 312
Elastic users, 167
Elephant projects, 38
  business use cases, 69
  communicating requirements for, 354
  completeness reviews, 373
  description, 33
  fit criteria, 280
  functional requirements, 225
  non-functional requirements, 247
  problem determination, 149
  Quality Gateway, 305
  scenarios, 130
  trawling, 89
Engineers for prototypes, 114
Enterprise constraints, 414–415
Entities for business use cases, 379
Environment
  constraints from, 409
  requirements from, 259–261, 273, 447–449
  separating work from, 40–41
Error rates, usability for, 253
Errors in software development, 306
Essence
  Brown Cow Model, 150–153
  discovering, 18
  importance, 26
Essential business solutions, 179–180
Estimates, cost, 17, 37, 315, 467–468
Ethnic groups, 182
Ethnography, 182, 184
Eurocontrol, 172
Events. See Business events
Evolution of requirements, 26–27
Exceptions
  completeness reviews, 375
  in functional requirements, 233–234
  scenarios for, 140–141, 145
Existing procedures, 320. See also Current situation and environment
Expectation management, 383
Expected physical environment, 447
Experts
  domain, 351
  as stakeholders, 51, 53
  subject-matter, 51, 333
Extended Enterprise Modeling Language (EEML), 399–400
Extensibility requirements, 446
Extent of products, 180–181
External profiles, 204–205
External strategy, 206–209
External technology in adjacent systems, 190–194
Externally stored data in function point counting, 490–492
Extreme programming
  testing in, 280
  user stories, 326

F
Fact/Assumption knowledge class, 499–500
Facts
  blastoff for, 37
  in reusing requirements, 339
  in risk analysis, 390
  in specification templates, 416–417
Index

Fagan inspections, 373–374
Failure demand, 164
Failures, fit criteria for, 288–289
Family therapy, 125
Fault tolerance requirements, 258, 445
Feasibility studies, 65
Feasible goals, 57
Feature Points, 481
Features
  in functional requirements, 237–238
  unnecessary, 317
Federal Information Security Management Act
  (FISMA), 270
Feedback
  innovation workshops, 172
  iterative development, 327
Feelings in innovation, 187–188
Ferdinandi, Patricia, 143
Financial beneficiaries, 51
Financial constraints, 60
Financial scandals, 269–270
Finding
  business events, 78–80
  fit criteria, 284–285
  functional requirements, 225–228
  non-functional requirements, 271–275
First-cut work context, 42–43
Fit criteria, 279
  ambiguous, 388
  atomic requirements, 363
  finding, 284–285
  formality guide, 280
  forms, 296–299
  for functional requirements, 231, 295–296
  measurement scale for, 285–286
  for non-functional requirements, 286–288
  cultural, 294–295
  legal, 295
  look and feel, 290–291
  maintainability, 294
  operational, 293–294
  performance, 292–293
  product failure, 288–289
  security, 294
  standards, 289–290
  subjective tests, 289
  usability and humanity, 291–292
  for project purpose, 299–300
  purpose of, 21, 280–282
  rationale for, 282–284
  solution constraints, 300
  for testability, 396
  testing, 281, 312–313
  in use cases, 299

Flows
  in business events, 78–80
  context diagrams for, 482
  Quality Gateways, 307–308
  in trawling, 92
Follow-up for new products, 462
Forces in patterns, 344–345
Form in goals, 399–400
Formality guides, 32–33
  business use cases, 69
  communicating requirements, 353–354
  fit criteria, 280
  functional requirements, 224–225
  non-functional requirements, 246–247
  problem determination, 149
  Quality Gateway, 304–305
  reviewing specifications, 372–373
  scenarios, 129–130
  scoping business problem, 38
  trawling, 89
Formality in Quality Gateway, 320
Formalized requirements, 303
Formalizing user stories, 331–332
Function point counting, 479–481
  adjustments in, 492
  business use cases, 483–488
  for cost estimates, 61–62, 467
  help and resources for, 492–494
  overview, 481
  scope, 481–482
  stored data, 482–483, 489–492
Functional beneficiaries, 51
Functional Requirement knowledge class, 500
Functional requirements, 10, 223–224
  alternatives to, 233–234, 238–241
  ambiguity in, 234–237
  conditional, 234
  conflicts in, 387
  data in, 231–233
  descriptions and rationale, 229–231
  in essential business, 179
  exceptions, 233–234
  fit criteria, 295–296
  formality guide, 224–225
  grouping, 237–238
  level of detail, 228–229
  vs. non-functional requirements, 248
  risk analysis, 390
  scope in, 237
  specification templates, 28, 358–359,
    367–368, 433–434
  technological, 237
  uncovering, 225–228
Functionality, 26
Fundamental processes in business use cases, 381
Future-How view in Brown Cow Model, 94, 175, 178–179

G
Geography as trawling consideration, 125
Glossaries, 415–416
Go/no go decisions
blastoff, 17, 37
scoping, 63–64
Goals
aspects, 57–58
blastoff, 17
in domain analysis, 351
measurable, 56–58
overview, 54–55
purpose, 55, 57–58
in specification templates, 398–400
in value, 165
Gold plating, 317
Google, 186
Google Docs, 367
Government as stakeholder, 53
Gramm-Leach-Bliley Act, 270
Granularity
functional requirements, 228–229
user stories, 331
Graphic fit criteria, 297
Graphs in fit criteria, 298
Groups
brainstorming, 173–174
ethnic, 182
functional requirements, 237–238
special-interest, 53
Guard conditions, 139

H
Hands-on users in specification templates, 403–404
Happy case scenarios, 135
Hardware safety requirements, 258
Harmful possibilities, scenarios for, 142–143
Hauser, John, 180
Health Insurance Portability and Accountability Act (HIPAA), 270
Help for function point counting, 492–494
High-fidelity prototypes, 115–116
High-level requirements, 238
HIPAA (Health Insurance Portability and Accountability Act), 270
History in atomic requirements, 365
Holtzblatt, Karen, 98, 113, 131
Homonyms, 235–237
Horse projects, 38
business use cases, 69
communicating requirements, 354
completeness reviews, 373
description, 33
fit criteria, 280
functional requirements, 224
non-functional requirements, 247
problem determination, 149
Quality Gateway, 305
scenarios, 129
trawling, 89
How-Now view in Brown Cow Model, 93–97, 131, 135, 150, 157, 174–175
Humanity requirements, 253–257
accessibility, 441
ease of use, 437–438
fit criteria for, 291–292
learning, 439–440
personalization and internationalization, 438–439
understandability and politeness, 440–441

I
Icons for prototypes, 115
Ideas
brainstorming, 173–174
brokering, 219–220
for solutions, 471
Identifying
business events, 376–378
gold plating, 317
stakeholders, 16
users, 48
Identity requirements, 361
Immunity requirements, 266, 454
Implementation environment, constraints from, 409
Implementing association, 507
Incremental improvements, 6–7
Incremental processes, 24–25
Incubation in innovation workshops, 172
Individual product use cases, 432
Industry standard setters, 52–53
Information
innovation, 186–187
requirements knowledge model, 353–357
Knowledge classes (continued)
  Stakeholder, 503–504
  System Architecture Component, 504
  Technological Requirement, 504
  Test, 504–505
  Work Scope, 505

L
Languages
  cultural requirements, 266–268
  functional requirements, 234–237
  maintainability requirements, 262
Latency requirements, 441–442
Latour, Bruno, 50
Launch. See Blastoff
Laws
  maintainability requirements for, 261
  robotics, 258
Lawyers, 269
Learning requirements, 439–440
Legacy as trawling consideration, 125
Legal experts as stakeholders, 52
Legal goals in specification templates, 399
Legal requirements, 268–271, 274
  compliance, 455–456
  fit criteria standards, 289–290, 295
  government, 269–270
  specification templates, 455–457
  standards, 271, 456–457
Leica cameras, 161
Lessons learned, 25
Level of detail in functional requirements, 228–229
Library domains, 348
Lifelike work situations, prototypes for, 115
Light measurements, 285
Lines in mind maps, 118
Links in mind maps, 116–119
Listening in interviews, 105–106
Lister, Tim, 63
Litigation costs, 268–271
Logical files, 489
Longevity requirements, 446
Look and feel requirements
  appearance, 435
  fit criteria for, 290–291
  overview, 250–253
  style, 436
Loudness measurements, 285
Low-fidelity prototypes, 111–115
Low-level functional requirements, 238

M
Maiden, Neil, 141, 172
Maintainability requirements, 261–262, 273, 294, 449–450
Maintenance operators as stakeholders, 48
Maintenance users in specification templates, 407
Management as stakeholders, 51
Management review in Quality Gateway, 321
Management templates for stakeholders, 473–477
Mandated constraints
  reusing requirements, 339
  risk analysis, 390
  specification templates, 407–415
Maps
  mind, 116–119
  stakeholder, 45, 473–474
Mark II function points, 481
Market forces as stakeholders, 52
Marketing department as stakeholders, 46
Materials for completeness reviews, 374
McBreen, Pete, 147
McMenamin, Steve, 110, 115
Meanings. See also Terms and terminology
  ambiguous, 388
  functional requirements, 234–237
  specification templates, 415–416
Meaningfulness, completeness tests for, 312
Measurements
  completeness reviews for, 391
  effort estimates, 61–62
  and fit criteria, 281–282, 285–286
  function point counting. See Function point counting
  and goals, 399
  in PAM technique, 399–400
  usability, 255
Meetings, blastoff, 64–65
Mellor, Stephen, 27
Merges in activity diagrams, 139
Michalko, Michael, 218
Microsoft SharePoint, 116, 367
Migration to new products, 463–465
Miller, Roxanne, 276
Mind maps, 116–119
Missing attributes, completeness reviews for, 311–312
Missing requirements, completeness reviews for, 374–375
Misuse cases, scenarios for, 142–143
Mobile phones, 161
Models
  apprenticeships with, 98–99
  Brown Cow. See Brown Cow Model
  business use cases, 378
data, 231–232
data dictionaries for, 416
domain, 342
for functional requirements, 240–241
quick and dirty, 107–109
requirements knowledge, 355–356
stakeholder involvement, 103
in trawling, 93–97
Modified data for new systems, 465
MoSCoW approach, 384
Motivation in goals, 398
Movies, downloadable, 148
Multiplicity in Volere Requirements Process Model, 495
Murder books, 119–120
Music media, 153–154, 161

N
Names for patterns, 345
Naming conventions
  blastoff, 37
  reusing requirements, 339
  scoping, 60–61
  specification templates, 415–416
Naming Conventions & Data Dictionary knowledge class, 501
Napoleonic wars, 384
Negative scenarios, 142–143
Negative stakeholders, 52
Netflix, 148, 161
New problems in specification templates, 460–462
Non-events, identifying, 378
Non-functional Requirement knowledge class, 501–502
Non-functional requirements, 10, 245–246
  adaptability, 450–451
  completeness reviews for, 375
  cultural and political, 266–268, 294–295, 454–455
  essential business, 179
  finding, 271–275
  fit criteria for, 286–295
  formality guide for, 246–247
  vs. functional, 247–248
  introduction, 246
  legal, 268–271, 295, 455–457
  look and feel, 250–253, 290–291, 435–436
  maintainability, 261–262, 294, 449–450
  operational and environment, 259–261, 293–294, 447–449
  performance, 257–259, 292–293, 441–446
  product failure, 288–289
  prototypes for, 274
  security, 262–266, 294, 451–454
  vs. solutions, 276–277
  in specification templates, 28–29, 274, 358–359, 368–369, 435–457
  standards, 289–290
  subjective tests, 289
  support, 261–262, 450
  types, 249–250
  usability and humanity, 253–257, 291–292, 437–441
  use cases in, 248–249
Normal cases, 135, 144
Normal operators as stakeholders, 48
Note taking, 119
Numbers in subjective tests, 289

O
Observations
  in trawling, 98–99
  videos for, 120
Off-the-shelf (OTS) products
  as constraint, 59, 410–411
  functional requirements for, 241–242
  in specification templates, 458–460
Onion diagrams, 44
Online book sales, 158–159
Open issues, 457–458
Open questions for interviews, 105
Open source applications, 59
Operational requirements, 259–261, 273, 293–294, 447–449
Operational support, 48, 50
Operational work area, 45
Optimism, problems from, 62
Organization maintainability requirements, 261
Organizing thoughts, mind maps for, 117
Originators in atomic requirements, 363
Origins of business events, 189–190
Osborne, Alex, 174
OTS (off-the-shelf) products
  as constraint, 59, 410–411
  functional requirements for, 241–242
  in specification templates, 458–460
Outcomes
- business use case workshops, 101
- scenarios, 145
- use cases, 299

Output business use cases, 485–487
Output flows in business events, 78–80
Outside world in business use cases, 72–73
Outsourcing requirements, 239
Owning association, 507

P
PAM (Purpose, Advantage, and Measurement) approach, 55–59, 399–400
Panasonic cameras, 161
Partitions
- business events, 75, 345
- business use cases, 69
- scope in innovation workshops, 172
- specification templates, 422–424
- work, 422–424

Partner systems and applications
- constraints from, 410
- fit criteria for, 293
- in operational requirements, 261

Passwords
- in non-functional requirements, 276
- problems, 157

Patterns, 342–344
- from abstraction, 346–351
- business event, 344–346
- collections, 343–344
- across domains, 349–351
- for specific domains, 348–349

Peer review, 321
Pena, William, 360
Penalty in value, 165–166
People in strategies, 204
Perceived solutions vs. system essence, 18
Performance requirements, 257–259, 272
- capacity, 445–446
- fit criteria, 292–293
- longevity, 446
- precision and accuracy, 443–444
- reliability and availability, 444–445
- robustness and fault-tolerance, 445
- safety-critical, 442–443
- scalability and extensibility, 446
- speed and latency, 187, 441–442

Personalization, 256, 438–439
Personas
- constructing, 182–183
- overview, 166–168

specification templates, 404–405
for stakeholders, 49
PESTLE (Political, Economic, Sociological, Technological, Legal, and Environmental) management technique, 64
Pfleeger, Shari Lawrence, 262
Phones
- addiction, 185
- mobile, 161
Photographs, 120–121
Photoshop usability, 254
Physical environment, expected, 447
Pictures for low-fidelity prototypes, 114
Piggybacking in brainstorming, 174
Planning tasks in specification templates, 462–463
Plans for innovation workshops, 172
Pleasure, paying for, 160
Plots in scenarios, 130
Policy as system essence, 18
Politeness requirements, 440–441
Political, Economic, Sociological, Technological, Legal, and Environmental (PESTLE) management technique, 64
Political beneficiaries as stakeholders, 51
Political correctness, 268
Political requirements, 266–268, 454–455
Post-it notes, 107–109
Potential of products, prototypes for, 115
Potential requirements
- communicating requirements from, 354
- formalized, 303
Potential users, 50
Potentially reusable requirements, 340
Precision requirements, 443–444
Preconceptions in problem determination, 153
Preconditions
- business use cases, 134
- scenario templates, 144
Preliminary cost estimates, 17
Prestige, paying for, 160–161
Priorities
- atomic requirements, 364
- business events, 217
- in functional requirements, 229
- iterative development, 211, 327–328
- user stories, 326
- users, 405–406
Prioritizing requirements
- completeness reviews for, 382–386
- factors, 382–383
- grading, 384
Quality Gateways, 22, 303–304
for completeness, 311–312
for consistent terminology, 313–314
effects, 304–305
for fit criteria, 312–313
formality guide, 304–305
for gold plating, 317
implementing, 319–321
quality requirements, 305–306
for requirement value, 316–317
for requirements creep, 317–319
for requirements vs. solutions, 315
scope, 307–311
in specification reviews, 371–372
for viability, 314–315
working with, 306–307
Quantifiable benefits as goals, 399
Questions
  interviews, 104–105
  user stories, 329–331
Quick and dirty modeling, 19, 107–109
Quickness commitment, 323
Rabbit projects, 38
  business use cases, 69
  communicating requirements, 353
  completeness reviews, 372–373
  description, 32–33
  fit criteria, 280
  functional requirements, 224
  non-functional requirements, 247
  problem determination, 149
  Quality Gateway, 304–305
  scenarios for, 129
  trawling, 89
Radiohead band, 169
Ranges
  fit criteria for, 293
  in function point counting, 492
Ratings and rankings by customers, 316–317, 363–364
Rationale
  atomic requirements, 362
  fit criteria, 282–284
  in functional requirements, 229–231
  for requirements, 21
Ready-made products, 458–459
Reasonable goals, 57
Reasoning for requirements, 388
Record elements, 489
Recording innovation workshops, 172
Red zones, 236
Reengineering in trawling, 97
Rees, Judy, 106
Reference step in CRUD checks, 380–381
Related patterns, 344, 346
Relationships in mind maps, 116–119
Releases
  in prioritizing requirements, 384
  requirements for, 449
Relevancy
  Quality Gateways, 309–311
  requirements knowledge model, 356
Relevant facts and assumptions
  blastoff, 37
  reusing requirements, 339
  risk analysis, 390
  specification templates, 416–417
Reliability requirements, 258, 444–445
Religious observances, 268
Renting movies, 148
Repairing errors, cost of, 306
Requirements, 13–14
  blastoff, 15–17
  case study, 15
  context, 14–15
  customizing, 31–32
  evolution, 26–27
  formality guide, 32–33
  functional. See Functional requirements
  issues. See Issues
  iterative and incremental processes, 24–25
  knowledge classes. See Knowledge classes
  known, 5–6
  non-functional. See Non-functional requirements
  overview, 9
  quality, 22, 305–306
  quick and dirty modeling, 19
  retrospective, 25
  reusing, 23, 217–218
  reviewing, 23–24
  scenarios, 20
  snow cards, 29–30
  vs. solutions, 315
templates for. See Volere requirements specification template
trawling for. See Trawling for requirements
truths, 1, 5–6, 9
types of, 249–250, 395–396
writing, 20–22, 353–357
Requirements bait, 110
Requirements creep, 317–319
Requirements Definition activity, 211–212, 214
Requirements knowledge model, 355–356
Index

Requirements profiles, 204–205
Requirements skills
  business rules, 218–219
  ideas brokering, 219–220
  innovation, 218
  strategies, 215–222
  systemic thinking, 220–221
  visualization, 221–222
Resources
  function point counting, 492–494
  requirements for, 258
Responses to events, 189, 344–345
Responsiveness to customers, 188
Retrospectives, 25
Reusable components, 459
Reusable requirements, 106–107
Reusing requirements, 23
  description, 338–341
  domain analysis in, 351
  overview, 337–338
  patterns in. See Patterns
  skills for, 217–218
  sources of, 341–342
Revenue goals in specification templates, 399
Reverse-engineering
  document archeology, 123
  essence, 151
Reviewing requirements specifications. See Completeness requirements
Reward in value, 165–166
Right problem, solving, 156–157
Risks and risk analysis, 63
  in blastoff, 37
  completeness reviews for, 388–390
  constraints, 390
  of damage, 258
  drivers, 389–390
  functional requirements, 390
  reviewing, 23
  in scoping, 62–63
  in solutions, 194–195
  in specification templates, 465–467
Robotics, laws of, 258
Robustness requirements, 258, 445
Roles in iterative development, 333–335
Rules
  business, 218–219
  maintainability requirements for, 261
Sabotage, 315
Safety-critical requirements, 442–443
Safety inspectors as stakeholders, 52
Safety requirements, 258
Saint-Exupery, Antoine de, 153
Sarbanes-Oxley Act (SOX), 269–270
Satellite broadcasting domain, 348–349
Satisfaction, customer
  atomic requirements, 363–364
  Quality Gateway for, 316–317
Scalability requirements, 258, 446
Scale of measurement for fit criteria, 285–286
Scandals, financial, 269–270
Scenarios, 129
  airline check-in agent, 131–140
  alternative cases, 139–140, 145
  business use case workshops, 101
  diagramming, 138–139
  exception cases, 140–141, 145
  formality guide for, 129–130
  functional requirements, 239
  negative, 142–143
  normal case, 135
  in process, 20
  product use cases, 196–199
  templates for, 131, 143–145
  what if?, 142
Schedules as constraints, 413
Scope
  in blastoff, 36
  boundaries, 429–431
  business use cases, 70–73, 82–83, 375–377
  external strategy, 207
  first-cut work context, 42–43
  in function point counting, 481–482
  in functional requirements, 237, 420–425
  innovation workshops, 172
  iterative strategy, 210–211, 213
  lead requirements analysts for, 16
  product, 180–181, 429–432
  Quality Gateways, 307–311
  in reusing requirements, 339
  risk analysis, 390
  specification templates, 420–425
  in systemic thinking, 164
  in trawling, 97
Scoping business problem, 35
  blastoff, 35–37
  blastoff meetings, 64–65
  case study, 41–43
  constraints, 59–60
  costs, 61–62
  external strategy, 207
  formality guide, 38
  go/no go decisions, 63–64
  goals, 54–59
  iterative strategy, 210–211, 213
Scoping business problem (continued)
  naming conventions and definitions, 60–61
  risks, 62–63
  scope setting, 38–41
  stakeholders. See Stakeholders
  trinity, 43–44
Security requirements, 263, 273
  access, 263, 451–452
  “and no more,” 265–266
  auditing, 265, 454
  fit criteria for, 294
  immunity, 454
  integrity, 264, 452–453
  privacy, 263–264, 453
Seddon, John, 162, 164
Self-documentation in legal requirements, 269
Self-referential approach, 167
Seminars for specification templates, 395
Separating work from environment, 40–41
Service goals in specification templates, 399
Service technicians in specification templates, 407
Shared commitment, 323
SharePoint, 116, 367
Shells
  requirements, 359
  for specifications, 21, 396
  “Should,” avoiding, 388
Simulations for subjective tests, 289
Sketches
  interface, 188–189
  overview, 109–115
SMART (Specific, Measurable, Attainable, Relevant and Timebound) management technique, 64
Smartphones addiction, 185
Snow cards
  atomic requirements, 359–361
  iterative development, 327
  for specifications, 21
  user stories, 331–332
  working with, 29–30
Sobel, Dava, 353
Software
  errors in, 306
  look and feel, 251
  off-the-shelf products. See Off-the-shelf (OTS) products
  for prototypes, 115
  safety requirements, 258
  truths, 2–4
Solutions and solution constraints, 59, 177–178
  adjacent systems, 190–194
  conclusion, 199–201
cost information, benefits, and risks, 194–195
document design decisions, 195–196
essential business, 179–180
fit criteria for, 279, 300
innovation, 184–188
iterative development, 179
origins of business events, 189–190
product extent, 180–181
product use case scenarios, 196–199
vs. requirements, 276–277
sketching interface, 188–189
in specification templates, 407–409, 471
user considerations, 181–184
Sorting prioritization categories, 384
Sound measurements, 285
Special-interest groups, 53
Specialized words in functional requirements, 235
Specific, Measurable, Attainable, Relevant and Timebound (SMART) management technique, 64
Specifications, 217
  for functional requirements, 225
  reviewing. See Completeness requirements templates for. See Volere requirements specification template
tools for, 21
Speed requirements, 187, 257, 441–442
Spelling in cultural requirements, 268
Sponsors as stakeholders, 45–47
Spreadsheets, 385–386
Stahl, Leslie Hulet, 121
Stakeholder knowledge class, 503–504
Stakeholders, 44–45
  acceptability of requirements to, 315
  in blastoff, 37
  Brown Cow Model, 159
  completeness tests for, 312
  customers as, 47–48
  finding, 54
  in functional requirements, 233
  identifying, 16
  interviewing, 102–106
  management templates, 473–477
  maps, 473–474
  miscellaneous, 50–54
  prototypes for, 111, 113–115
  in reusing requirements, 339–340
  in risk analysis, 389
  for scenarios, 131, 133–134, 144
  specification templates, 400–407
  sponsors, 45–47
Index

in trawling, 91–92
users as, 48–50
Standard setters as stakeholders, 52–53
Standards
  branding, 251, 290
  fit criteria, 289–290
  legal requirements, 271
  in specification templates, 456–457
Stored data in function point counting, 482–483, 489–492
Stories. See Scenarios
Story cards, 239–240
Strategies, 203
  determining, 215
  external, 206–209
  iterative, 210–212
  knowledge, activities, and people in, 204
  knowledge requirements, 205–206
  progressive, 212–214
  project requirements profiles, 204–205
  requirements skills, 215–222
Strengths, Weaknesses, Opportunities, and Threats (SWOT) management technique, 64
Style requirements, 436
Subject-matter experts
  iterative development, 333
  as stakeholders, 51
Subjective interpretation, 313
Subjective tests, fit criteria for, 289
Subtypes in function point counting, 489–490
Sullivan, Wendy, 106
Support requirements, 261–262, 273, 450
Supporting association, 508
Supporting materials in atomic requirements, 365
Swim lanes, 154–156
SWOT (Strengths, Weaknesses, Opportunities, and Threats) management technique, 64
System Architecture Component knowledge class, 504
Systemic thinking, 162–164, 220–221
Systems
  adjacent. See Adjacent systems
  business events, 76–77
  business use cases, 70
T
Tables of contents in templates, 27, 357–358, 393–394
Tasks in specification templates, 462–463
Team review in Quality Gateway, 321
Technical experts as stakeholders, 53
Technical knowledge in iterative development, 334–335
Technicians in specification templates, 407
Technological fossils, 76
Technological Requirement knowledge class, 504
Technological requirements, 225, 237
Technological skills, 315
Technology
  in problem statements, 151–152
  for wikis, 22
Templates, 27–29
  non-functional requirements, 274
  scenarios, 131, 143–145
  specifications. See Volere requirements specification template
  stakeholders management, 473–477
Terms and terminology
  ambiguous, 388
  blastoff for, 37
  functional requirements, 234–237
  Quality Gateways for, 313–314
  specification templates, 415–416
  stakeholder interviews, 103
Test cases
  functional requirements, 296
  iterative development, 327
Test knowledge class, 504–505
Testability
  fit criteria for, 396
  of goals, 399
  of requirements, 8–9
Testing
  completeness, 311–312
  extreme programming, 280
  fit criteria, 280, 312–313
  Quality Gateways for, 22
  requirements, 396
Testing association, 508
Texting, 185
Thinking, importance of, 8
Thought organization, mind maps for, 117
Three strikes approach, 277
Throughput requirements, 258
Throwaway prototypes, 110
Time constraints in blastoff, 60
Time in product failure measurements, 288
Time-triggered business events, 74–75
Time-triggered business use cases, 487–488
Tolerances
  for fit criteria, 284–285
  in subjective tests, 289
Tower of Babel, 314
Training in specification templates, 469–470
Translated data for new systems, 465
Translators, analysts as, 91
Trawling for requirements, 17–18, 87–88
    analysts for, 91–92
    apprenticeships in, 98–99
    Brown Cow Model, 93–97
    business use case workshops, 99–102
    business use cases, 92
    current situation in, 94–97
    diagrams, 92
    document archeology in, 123–124
    family therapy, 125
    formality guide for, 89
    interviews, 102–106
    for knowledge, 89–90, 126
    mind maps, 116–119
    modeling, 107–109
    murder books, 119–120
    observations, 98–99
    photographs, 120–121
    prototypes and sketches, 109–116
    reusable requirements, 106–107
    techniques, 125–129
    video, 120–121
    wikis, blogs, and discussion forums, 122
Triage in prioritizing requirements, 384
Triggers
    business use cases, 133–134
    innovation, 184
    scenario templates, 144
Trust, 187
Tufte, Edward, 221
Typeface measurements, 285
Types, requirement, 249–250, 361, 395–396

U
Uncertainty range in function point counting, 492
Understandability requirements, 440–441
Unduplicated attributes, 488
Unified Modeling Language (UML)
    activity diagrams, 138, 240
    use case diagrams, 483
Universal cures, 31
Unnecessary features and requirements, 317
Unqualified adjectives and adverbs, 388
Update step in CRUD checks, 380–381
Usability requirements, 49
    accessibility, 441
    ease of use, 437–438
    fit criteria for, 291–292
    learning, 439–440
    overview, 253–257
    personalization and internationalization, 438–439
    understandability and politeness, 440–441
Use cases
    business. See Business use cases (BUCs)
    fit criteria in, 299
    non-functional requirements, 248–249, 272–274
    product. See Product use cases (PUCs)
    in scope, 431–432
    UML use case diagrams, 483
User business in specification templates, 397–398
User documentation in specification templates, 468–469
User experience
    designing, 183–184
    solutions, 201
User-friendliness as requirement, 282–283
User management as stakeholder, 46
User problems for new products, 461
User stories
    fleshing out, 332–333
    formalizing, 331–332
    functional requirements, 239–240
    iterative development, 325–326, 329–333
    questions, 329–331
Users
    priorities, 405–406
    in reusing requirements, 339
    in risk analysis, 389–390
    in solutions, 181–184
    in specification templates, 403–404, 406
    as stakeholders, 48–50
    understanding of requirements by, 9

V
Value
    overview, 165–166
    in solutions, 195
Value demand, 164
Verbs, 106
Version numbers, 383
Viability within constraints, 314–315
Viable goals, 57
Video records, 120–121
Viruses, 266
Visualization, 221–222
Volere Requirements Process Model overview, 11–12
Volere requirements specification template, 357
  assembling, 365–366
  assumptions in, 418–419
  atomic requirements, 359–365
  automated tools, 366–367
  for completeness reviews, 374
  constraints in, 407–415
  data dictionaries, 427–429
  data model, 425–427
  data requirements in, 433–434
  divisions, 358–359
  facts in, 416–417
  functional requirements, 367–368, 433–434
  naming conventions and definitions,
    415–416
  non-functional requirements, 368–369,
    435–457
  product scope, 429–432
  project issues, 369, 457–471
  purpose, 397–400
  requirements types, 395–396
  shell in, 396
  stakeholders, 400–407
  tables of contents, 357–358, 393–394
  testing requirements, 396
  use of, 394
  work scope, 420–425

Weights for prioritizing requirements, 385
What element in Brown Cow Model, 150
What if? scenarios, 142
What-Now view in Brown Cow Model, 93
Whiteboards, 107
Wider environment in stakeholder maps, 45
Wikis
  non-functional requirements, 271
  trawling, 122
Wittenberg, Ethel, 294
Word processors, 366–367
Words. See Terms and terminology
Work
  business use cases, 70–72
  context, 42–43, 92
  in iterative development, 324, 327
  partitioning. See Partitions
  reengineering, 97
  in scope, 39
Work area measurements, 480–481
Work investigation activity
  external strategy, 207–209
  iterative strategy, 210–214
work scope diagrams, 41–43
Work Scope knowledge class, 505
Working models in trawling, 94
Workplace environment, constraints from, 412
workshops
  business use cases, 99–102
  innovation, 171–172
  use case, videos for, 121
Writing requirements, 20–22, 353
  formality guide, 353–354
  knowledge vs. specification in, 353–357
  potential requirements, 354

W
Waist-High Shelf pattern, 343
Waiting room, 470–471
Warning messages, 269
Waterfall process, 324
Web-based products, 252