# Contents

<table>
<thead>
<tr>
<th>About the Authors</th>
<th>xii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xiii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 The Touchstones: Lean and Agile</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Lean Architecture and Agile Feature Development</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Agile Production</td>
<td>7</td>
</tr>
<tr>
<td>1.3.1 Agile Builds on Lean</td>
<td>7</td>
</tr>
<tr>
<td>1.3.2 The Scope of Agile Systems</td>
<td>8</td>
</tr>
<tr>
<td>1.3.3 Agile and DCI</td>
<td>9</td>
</tr>
<tr>
<td>1.4 The Book in a Very Small Nutshell</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Lean and Agile: Contrasting and Complementary</td>
<td>11</td>
</tr>
<tr>
<td>1.5.1 The Lean Secret</td>
<td>14</td>
</tr>
<tr>
<td>1.6 Lost Practices</td>
<td>14</td>
</tr>
<tr>
<td>1.6.1 Architecture</td>
<td>15</td>
</tr>
<tr>
<td>1.6.2 Handling Dependencies between Requirements</td>
<td>15</td>
</tr>
<tr>
<td>1.6.3 Foundations for Usability</td>
<td>16</td>
</tr>
<tr>
<td>1.6.4 Documentation</td>
<td>16</td>
</tr>
<tr>
<td>1.6.5 Common Sense, Thinking, and Caring</td>
<td>19</td>
</tr>
<tr>
<td>1.7 What this Book is Not About</td>
<td>21</td>
</tr>
<tr>
<td>1.8 Agile, Lean – Oh, Yeah, and Scrum and Methodologies and Such</td>
<td>22</td>
</tr>
<tr>
<td>1.9 History and Such</td>
<td>24</td>
</tr>
<tr>
<td><strong>2 Agile Production in a Nutshell</strong></td>
<td>27</td>
</tr>
<tr>
<td>2.1 Engage the Stakeholders</td>
<td>27</td>
</tr>
<tr>
<td>2.2 Define the Problem</td>
<td>29</td>
</tr>
<tr>
<td>2.3 Focusing on What the System Is: The Foundations of Form</td>
<td>30</td>
</tr>
<tr>
<td>2.4 Focusing on What the System Does: The System Lifeblood</td>
<td>32</td>
</tr>
<tr>
<td>2.5 Design and Code</td>
<td>33</td>
</tr>
<tr>
<td>2.6 Countdown: 3, 2, 1…</td>
<td>34</td>
</tr>
</tbody>
</table>
## 3 Stakeholder Engagement

### 3.1 The Value Stream
- 3.1.1 End Users and Other Stakeholders as Value Stream Anchors
- 3.1.2 Architecture in the Value Stream
- 3.1.3 The Lean Secret

### 3.2 The Key Stakeholders
- 3.2.1 End Users
  - Psyching Out the End Users
  - Don’t Forget Behavior
  - The End User Landscape
- 3.2.2 The Business
- 3.2.3 Customers
  - … As Contrasted with End Users
  - “Customers” in the Value Stream
- 3.2.4 Domain Experts
  - No Ivory Tower Architects
- 3.2.5 Developers and Testers

### 3.3 Process Elements of Stakeholder Engagement
- 3.3.1 Getting Started
- 3.3.2 Customer Engagement

### 3.4 The Network of Stakeholders: Trimming Wasted Time
- 3.4.1 Stovepipe Versus Swarm
- 3.4.2 The First Thing You Build
- 3.4.3 Keep the Team Together

### 3.5 No Quick Fixes, but Some Hope

## 4 Problem Definition

### 4.1 What’s Agile about Problem Definitions?
### 4.2 What’s Lean about Problem Definitions?
### 4.3 Good and Bad Problem Definitions
### 4.4 Problems and Solutions
### 4.5 The Process Around Problem Definitions
  - 4.5.1 Value the Hunt Over the Prize
  - 4.5.2 Problem Ownership
  - 4.5.3 Creeping Featurism
### 4.6 Problem Definitions, Goals, Charters, Visions, and Objectives
### 4.7 Documentation?

## 5 What the System Is, Part 1: Lean Architecture

### 5.1 Some Surprises about Architecture
- 5.1.1 What’s Lean about This?
- Deliberation and “Pull”
- Failure-Proof Constraints or Poka-Yoke
- The Lean Mantras of Conservation, Consistency, and Focus
- 5.1.2 What’s Agile about Architecture?
- It’s All About Individuals and Interactions
7 What the System Does: System Functionality 165

7.1 What the System Does 166

7.1.1 User Stories: A Beginning 166
7.1.2 Enabling Specifications and Use Cases 167
7.1.3 Helping Developers, Too 169
7.1.4 Your Mileage may Vary 170

7.2 Who is Going to Use Our Software? 171

7.2.1 User Profiles 171
7.2.2 Personas 171
7.2.3 User Profiles or Personas? 172
7.2.4 User Roles and Terminology 173

7.3 What do the Users Want to Use Our Software for? 173

7.3.1 Feature Lists 173
7.3.2 Dataflow Diagrams 174
7.3.3 Personas and Scenarios 174
7.3.4 Narratives 174
7.3.5 Behavior-Driven Development 175
7.3.6 Now that We’re Warmed Up... Prototypes 176
Towards Foundations for Decisions 176
Known and Unknown Unknowns 176
Use Cases as a Decision Framework 177

7.4 Why Does the User Want to Use Our Software? 177

7.5 Consolidation of What the System Does 178

7.5.1 The Helicopter View 181
Habits: The Developer View and the User View 182
Trimming the Scope 185
7.5.2 Setting the Stage 186
7.5.3 Play the Sunny Day Scenario 187
Business Rules 191
7.5.4 Add the Interesting Stuff 193
7.5.5 Use Cases to Roles 200
Roles from the Use Case 201
Bridging the Gap between the Business and the Programmer 202

7.6 Recap 203

7.6.1 Support the User’s Workflow 203
7.6.2 Support Testing Close to Development 203
7.6.3 Support Efficient Decision-Making about Functionality 204
7.6.4 Support Emerging Requirements 204
7.6.5 Support Release Planning 204
7.6.6 Support Sufficient Input to the Architecture 205
7.6.7 Support the Team’s Understanding of What to Develop 205
## Contents

10  Epilog  305

Appendix A  Scala Implementation of the DCI Account Example  307

Appendix B  Account Example in Python  311

Appendix C  Account Example in C#  315

Appendix D  Account Example in Ruby  321

Appendix E  Qi4j  327

Appendix F  Account Example in Squeak  331
  F.1 Testing Perspective  333
  F.2 Data Perspective  333
    F.2.1 BB5Bank  333
    F.2.2 BB5SavingsAccount  334
    F.2.3 BB5CheckingAccount  334
  F.3 Context Perspective  335
    F.3.1 BB5MoneyTransferContext  335
  F.4 Interaction (RoleTrait) Perspective  336
    F.4.1 BB5MoneyTransferContextTransferMoneySource  336
    F.4.2 BB5MoneyTransferContextMyContext  337
    F.4.3 BB5MoneyTransferContextTransferMoneySink  337
  F.5 Support Perspective (Infrastructure Classes)  337
    F.5.1 BB1Context (common superclass for all contexts)  337
    F.5.2 BB1RoleTrait (all RoleTraits are instances of this class)  339

Bibliography  341

Index  351