Architecting Complex-Event Processing Solutions with TIBCO®
TIBCO® Press provides books to help users of TIBCO technology design and build real-world solutions. The initial books – the architecture series – provide practical guidance for building solutions by combining components from TIBCO’s diverse product suite. Each book in the architecture series covers an application area from three perspectives: a conceptual overview, a survey of applicable TIBCO products, and an exploration of common design challenges and TIBCO-specific design patterns for addressing them. The first book in the series, TIBCO® Architecture Fundamentals, addresses the basics of SOA and event-driven architectures. Each of the advanced books addresses a particular architecture style, including composite applications and services, complex event processing, business process management, and data-centric solutions.

The series emphasizes the unification of business process and system design in an approach known as total architecture. A technology-neutral description of this approach to distributed systems architecture is described in Implementing SOA: Total Architecture in Practice. Techniques for addressing the related organizational and management issues are described in Succeeding with SOA: Realizing Business Value through Total Architecture.

Visit informit.com/tibcopress for a complete list of available publications.
Architecting Complex-Event Processing Solutions with TIBCO®

Paul C. Brown
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.

TIBCO, TIBCO ActiveMatrix Adapter for Database, TIBCO ActiveMatrix BusinessWorks, TIBCO ActiveMatrix BPM, TIBCO ActiveMatrix Service Bus, TIBCO ActiveMatrix Service Grid, TIBCO ActiveSpaces, TIBCO Adapter for Files, TIBCO Administrator, TIBCO BusinessEvents, TIBCO BusinessEvents Data Modeling, TIBCO BusinessEvents Decision Manager, TIBCO BusinessEvents Event Stream Processing, TIBCO BusinessEvents Process Orchestration, TIBCO BusinessEvents Views, TIBCO Enterprise Message Service, TIBCO Hawk, TIBCO Rendezvous, TIBCO Runtime Agent are either registered trademarks or trademarks of TIBCO Software Inc. and/or its affiliates in the United States and/or other countries.

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

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@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Brown, Paul C.
Architecting complex-event processing solutions with TIBCO / Paul C. Brown.
pages cm
Includes index.
1. Business logistics—Data processing. 2. Event processing (Computer science)
3. TIBCO Software Inc. I. Title.
HD38.5.B76 2014
658.50285'53—dc23 2013026369

Copyright © 2014 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 RR Donnelley in Crawfordsville, Indiana.
First printing, September 2013
To Mugs and Willie
This page intentionally left blank
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex-Event Processing (CEP)</td>
<td>17</td>
</tr>
<tr>
<td>Event Correlation</td>
<td>20</td>
</tr>
<tr>
<td>Context</td>
<td>21</td>
</tr>
<tr>
<td>Constants</td>
<td>22</td>
</tr>
<tr>
<td>Data</td>
<td>22</td>
</tr>
<tr>
<td>Metadata</td>
<td>23</td>
</tr>
<tr>
<td>Analysis Requires Context</td>
<td>23</td>
</tr>
<tr>
<td>Selecting an Analytical Approach</td>
<td>25</td>
</tr>
<tr>
<td>Responding to Events</td>
<td>26</td>
</tr>
<tr>
<td>Event-Driven Processes</td>
<td>28</td>
</tr>
<tr>
<td>Event-Enabled Enterprise Capabilities</td>
<td>31</td>
</tr>
<tr>
<td>Summary</td>
<td>32</td>
</tr>
<tr>
<td><strong>Chapter 3: CEP Solution Design Patterns</strong></td>
<td>35</td>
</tr>
<tr>
<td>Objectives</td>
<td>35</td>
</tr>
<tr>
<td>Variability in CEP Architectures</td>
<td>36</td>
</tr>
<tr>
<td>Handling Reference Data</td>
<td>36</td>
</tr>
<tr>
<td>Partitioning Functionality</td>
<td>37</td>
</tr>
<tr>
<td>Condition Detection</td>
<td>39</td>
</tr>
<tr>
<td>Situation Recognition</td>
<td>41</td>
</tr>
<tr>
<td>Track and Trace</td>
<td>42</td>
</tr>
<tr>
<td>Business Process Timeliness Monitor</td>
<td>44</td>
</tr>
<tr>
<td>Situational Response</td>
<td>45</td>
</tr>
<tr>
<td>Decision as a Service</td>
<td>46</td>
</tr>
<tr>
<td>Orchestrated Response</td>
<td>48</td>
</tr>
<tr>
<td>Pioneering Solutions</td>
<td>50</td>
</tr>
<tr>
<td>Summary</td>
<td>51</td>
</tr>
<tr>
<td><strong>Part II: Technology</strong></td>
<td>53</td>
</tr>
<tr>
<td><strong>Chapter 4: TIBCO BusinessEvents®</strong></td>
<td>55</td>
</tr>
<tr>
<td>Objectives</td>
<td>55</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Product Suite</td>
<td>55</td>
</tr>
<tr>
<td>TIBCO BusinessEvents®</td>
<td>56</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIBCO BusinessEvents® Data Modeling</td>
<td>57</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Decision Manager</td>
<td>58</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Event Stream Processing</td>
<td>60</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Process Orchestration</td>
<td>61</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Views</td>
<td>61</td>
</tr>
<tr>
<td>TIBCO BusinessEvents® Solution Deployment</td>
<td>62</td>
</tr>
<tr>
<td>BusinessEvents Solution Life Cycle</td>
<td>65</td>
</tr>
<tr>
<td>Summary</td>
<td>67</td>
</tr>
</tbody>
</table>

## Chapter 5: Inference Agents

### Objectives

69

### Inference Agent Overview

70

### Events, Concepts, and Scorecards

**Events**

71

**Concepts**

73

**Scorecards**

75

### Rules

77

**Attributes**

77

**Declarations**

77

**Conditions**

78

**Actions**

78

### Run-to-Completion (RTC) Behavior

79

### Rule Conditions and Rete Network Efficiency

83

### Completing the Inference Agent: Preprocessing and Postprocessing

87

**Channels**

88

**Destinations**

89

**Preprocessor Functions**

90

**Directing Events**

90

### Preprocessing Behavior

91

### Postprocessing Behavior

93

### State Models

98

**State Transitions**

98

### Timeouts

99
Starting and Stopping State Machines 99
Summary 100

Chapter 6: Cache Agents 103
Objectives 103
The Need for a Cache 103
The Cache and Cache Agents 104
Object Management Modes 104

  Cache Only 105
  Memory Only 105
  Cache + Memory 106
Object Locking 109
Cache Object Replication 110
Object Persistence 111

  Shared-All Option 111
  Shared-Nothing Option 113
Summary 113

Chapter 7: Query Agents 115
Objectives 115
Snapshot Queries 115

  Snapshot Query Execution 115
  Snapshot Query Life Cycle 117
Continuous Queries 121
Buffer Management 122

  Continuous Query Life Cycle 123
Summary 126

Chapter 8: Process Agents 127
Objectives 127
Intended Utilization 127
Processes 130
Behavior 130
Deployment 132
Summary 133
Chapter 9: Dashboard Agents 135

Objectives 135
Dashboard Configuration 135
Behavior 136
Metrics 136
Dashboard 137
Deployment 139
Summary 139

Part III: Design Patterns 141

Chapter 10: Solution Basics 143

Objectives 143
Recognizing a Situation Change 143
Reference-Data Comparison Pattern 144
Systems of Record for Reference Data 145
   TIBCO BusinessEvents® as Reference-Data System of Record 145
   Database as Reference-Data System of Record 146
   External System as Reference-Data System of Record 146
Reference-Data Change Coordination Patterns 147
State Machine Change Recognition Pattern 149
Continuous Query Change Recognition Pattern 151
Handling Duplicate Events 151
Enabling Run-Time Rule Changes 154
   Rule Templates 154
   Decision Tables 155
   Rule Management Server (RMS) 156
Sequential and Conditional Action Performance 157
   Orchestration Implemented in the Action Section of a Single Rule 157
   Having a Separate Rule for Each Action 157
   Sequencing the Execution of Rules 158
   Orchestration Implemented in an Explicit Orchestration Component 159
| Contents |
|------------------|-------|
| Database Query | 181   |
| Database Update and Delete | 182   |
| Database Interaction Using TIBCO ActiveMatrix® Adapter for Database | 182   |
| Inference Agent Publication | 183   |
| Inference Agent Request-Reply | 183   |
| Inference Agent Subscription | 184   |
| Database Interaction Using TIBCO ActiveMatrix BusinessWorks™ | 185   |
| Summary | 185   |

**Chapter 13: Solution Modularization Patterns** 187

<table>
<thead>
<tr>
<th>Objectives</th>
<th>187</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning Situation Recognition from Action</td>
<td>188</td>
</tr>
<tr>
<td>Partitioning Filtering and Enhancement from Rule Processing</td>
<td>190</td>
</tr>
<tr>
<td>Using TIBCO ActiveMatrix BusinessWorks™ for Filtering and Enrichment</td>
<td>191</td>
</tr>
<tr>
<td>Partitioning Advantages and Disadvantages</td>
<td>192</td>
</tr>
<tr>
<td>Partitioning Rules of Thumb</td>
<td>192</td>
</tr>
<tr>
<td>Summary</td>
<td>193</td>
</tr>
</tbody>
</table>

**Chapter 14: Common Design Challenges** 195

<table>
<thead>
<tr>
<th>Objectives</th>
<th>195</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Sharing</td>
<td>195</td>
</tr>
<tr>
<td>Using an Event for Information Sharing</td>
<td>196</td>
</tr>
<tr>
<td>Using the Cache for Information Sharing</td>
<td>196</td>
</tr>
<tr>
<td>Locking</td>
<td>198</td>
</tr>
<tr>
<td>Locks</td>
<td>198</td>
</tr>
<tr>
<td>Locking Requires Programming Discipline</td>
<td>199</td>
</tr>
<tr>
<td>Avoiding Deadlocks</td>
<td>199</td>
</tr>
<tr>
<td>Locking and Data Structures</td>
<td>199</td>
</tr>
<tr>
<td>Load Distribution</td>
<td>201</td>
</tr>
<tr>
<td>Using IP Redirectors to Distribute Load</td>
<td>201</td>
</tr>
<tr>
<td>Using JMS Queues to Distribute Load</td>
<td>201</td>
</tr>
<tr>
<td>Using TIBCO BusinessEvents® Load Balancer to Distribute Load</td>
<td>202</td>
</tr>
<tr>
<td>Directing Related Work to a Single Agent</td>
<td>202</td>
</tr>
<tr>
<td>Managing Sequencing</td>
<td>203</td>
</tr>
<tr>
<td>Preserving Sequencing within One Inference Agent</td>
<td>204</td>
</tr>
<tr>
<td>Preserving Sequencing across Multiple Inference Agents</td>
<td>205</td>
</tr>
<tr>
<td>Recovering Temporal Sequencing (Reordering)</td>
<td>205</td>
</tr>
<tr>
<td>Handling Duplicate Events</td>
<td>206</td>
</tr>
<tr>
<td>Summary</td>
<td>207</td>
</tr>
</tbody>
</table>

**Part IV: Deployment**

**Chapter 15: Case Study: Nouveau Health Care**

| Objectives | 211 |
| Nouveau Health Care Solution Architecture | 212 |
| Nouveau Health Care Business Processes | 212 |
| Nouveau Health Care Architecture Pattern | 213 |
| Nouveau Health Care in Context | 214 |
| Processing Claims from Providers | 215 |
| Claim Tracker | 217 |
| Claim Status Concept | 218 |
| Claim Track Interface | 219 |
| Claim Tracker Processes | 221 |
| Monitor Claim Processing | 222 |
| Obtain Claim Status | 224 |
| Summary | 224 |

**Chapter 16: Performance**

| Objectives | 225 |
| TIBCO BusinessEvents® Profiler | 225 |
| Design Choices and Agent Performance | 226 |
| Structuring Rule Conditions | 227 |
| Organizing Decision Tables | 228 |
| Accessing Large XML Event Payloads | 228 |
Site Disaster Recovery 256
Summary 257

Chapter 19: Best Practices 259
Objectives 259
Architecture Planning 259
Designing Data Models for Concepts 260
Object Management Modes, Threading, and Locking 261
Designing Rules 261
Testing Best Practices 262
Summary 262

Index 265
Preface

Complex-Event Processing

Complex-event processing is a nontraditional style of building solutions. This style makes it possible to address problems that do not yield well to traditional approaches such as real-time situation analysis. More broadly, complex-event processing enables the enterprise to sense, analyze, and respond to its business situations in new and innovative ways—ways that provide extreme value and competitive advantage.

In complex-event processing solutions, the word complex comes into play in two very different ways. The first refers to sensing, analyzing, and responding to what is going on. It’s not just, “Oh, this event occurred, therefore I need to do <some activity>.” It’s more complex than that: It requires correlating that event with other events and with contextual information in order to understand whether a situation of business importance exists, and then deciding what, if anything, needs to be done. Complexity in sensing, complexity in analyzing, complexity in responding.

The other way that complexity applies is that complex-event processing involves a wide variety of computational techniques. There is no single approach to sensing, analyzing, and responding that is suitable for all types of situations. Each of the approaches has its own strengths and weaknesses, all of which need to be understood in order for you to craft your solution.

About This Book

This book provides an introduction to the complex-event processing space and the computational approaches enabled by TIBCO BusinessEvents®. It is divided into four parts: Getting Started, Technology, Design Patterns, and Deployment.

Part I, Getting Started, provides a conceptual overview of the complex-event processing space. It discusses how complex-event
processing can be employed in a business context to provide competitive differentiation, covers the terminology of complex-event processing, and explores the ways in which complex-event processing is different from traditional computing. It also explores a number of business applications for complex-event processing.

Part II, Technology, covers the capabilities of the TIBCO Business Events® product suite. It covers the TIBCO Business Events suite of products and presents a life-cycle overview of solutions based on these products. The TIBCO Business Events executable, a Java virtual machine (JVM), can be configured with combinations of five functional components: inference agents, cache agents, query agents, process agents, and dashboard agents. Inference agents process rules, and cache agents provide the information-sharing mechanism within TIBCO BusinessEvents. Query agents provide both snapshot and continuous queries of cached information. Process agents provide orchestration capabilities, while dashboard agents provide real-time visualization capabilities. The architecture and functionality of each type of agent are explored.

Part III, Design Patterns, explores the building-block design patterns used in constructing complex-event processing solutions with TIBCO BusinessEvents. Patterns for recognizing situation changes, comparisons and changes to reference data, systems of record, handling duplicate inputs, run-time rule changes, and orchestrating actions are explored. Patterns for pattern recognition, integration, solution modularization, information sharing, locking, load distribution, and sequencing are covered.

Part IV, Deployment, covers the architecturally significant aspects of putting a solution into production. The Nouveau Health Care case study is a realistic design problem that illustrates many of the issues an architect needs to address. It is used as an example to explore performance, modularization for deployment, managing the cache and backing store, defining deployment patterns, and monitoring. Design patterns for solution fault tolerance, high availability, and site disaster recovery are discussed, along with best practices for the conduct of complex-event processing projects.

The organization of the book is shown in Figure P-1.

---

**Online Examples**

Many of the examples in this book are taken from actual TIBCO BusinessEvents projects that are available online. All of these projects begin with the prefix ACEPST and can be found at informit.com/title/9780321801982.
TIBCO Architecture Book Series

Architecting Complex-Event Processing Solutions with TIBCO® is the third book in a series on architecting solutions with TIBCO products (Figure P-2). It builds upon the material covered in TIBCO® Architecture Fundamentals, which provides material common to all TIBCO-based designs. Each of the more advanced books, including this one, explores a different style of solution, all based on TIBCO technology. Each explores the additional TIBCO products that are relevant to that style of solution. Each defines larger and more specialized architecture patterns relevant to the style, all built on top of the foundational set of design patterns presented in TIBCO® Architecture Fundamentals.
Intended Audience

*Project architects* are the intended primary audience for this book. These are the individuals responsible for defining an overall complex-event processing solution and specifying the components and services required to support that solution. Experienced architects will find much of interest, but no specific prior knowledge of architecture is assumed in the writing. This is to ensure that the material is also accessible to novice architects and advanced designers. For this latter audience, however, a reading of *TIBCO® Architecture Fundamentals*¹ and *Architecting Composite Applications and Services with TIBCO®*² is highly recommended. These books explore integration and services along with the broader topics of solution architecture specification and documentation.

TIBCO specialists in a complex-event processing center of excellence will find material of interest, including background on TIBCO BusinessEvents product suite and related best-practice design patterns. The material on performance and tuning lays the foundation for building high-performance applications based on the product suite.

---

Enterprise architects will find content of interest as well. The collection of design patterns, in conjunction with those presented in TIBCO® Architecture Fundamentals, provides the basis for a baseline set of standard design patterns for the enterprise.

Detailed Learning Objectives

After reading this book, you will be able to

• Describe the characteristics of an event-enabled enterprise
• Explain the concepts related to complex-event processing
• List examples of complex-event processing solutions
• Describe the TIBCO BusinessEvents product suite
• Explain the operation and tuning of TIBCO BusinessEvents agents
• Explain how situations and changes in situations can be recognized
• Describe how rules can be changed at runtime
• Explain how activities can be orchestrated
• Describe how patterns of events can be recognized
• Modularize complex-event processing solutions to facilitate maintainability and scalability
• Describe how to share information among distributed components of a complex-event processing solution
• Select and apply appropriate patterns for load distribution, fault tolerance, high availability, and site disaster recovery
• Explain how design choices impact agent performance
• Define deployment patterns for complex-event processing solutions
• Describe the best practices for conducting complex-event processing projects
This page intentionally left blank
Acknowledgments

I must begin by acknowledging the extraordinary contribution of Rodrigo Abreu in formulating the content of this book and the accompanying TIBCO Education course. Not only was he instrumental in helping me fine-tune the scope of material to be covered, but he also clarified (often by experiment) many questions about actual product behavior. This was all in addition to his “day job,” and often on his own time. It is fair to say that without his assistance, neither this book nor the accompanying course would exist. I am in his debt.

This book and accompanying course started out as a conversation with Paul Vincent in September 2010. The outline we put together at that time has stood the test of time and it can still be clearly recognized in the finished product. Wenyan Ma made many valuable contributions in defining the scope of material to be covered, and Michael Roeschter made significant contributions to the content.

Many others have contributed to the technical content: Pranab Dhar, Sergio Gonik, Ryan Hollom, Fatih Ildiz, Ali Nikkhah, Mitul Patel, Nicolas Prade, Patrick Sapinski, Rajarsi Sarkar, Rajesh Senapathy, Piotr Smolinski, Suresh Subramani, Piotr Szuszkiewicz, and Yueming Xu. I am grateful for their contributions.

My deepest thanks to the TIBCO Education team who worked with me on this project: Alan Brown, Mike Fallon, Michelle Jackson, and Madan Mashalkar. In more ways than I can mention, they made it all happen. Special thanks to Jan Plutzer, who strongly supported this effort from its inception.

Without the strong support of Eugene Coleman, Paul Asmar, and Murat Sonmez, I would not have been able to dedicate the time necessary for this effort. You have my deepest gratitude.

I would like to acknowledge those who took the time to review the manuscript: Abby Brown, Antonio Bruno, Benjamin Dorman, Jose Estefania, Lloyd Fischer, Alexandre Jeong, James Keegan, Lee Kleir, Edward Rayl, Michael Roeschter, and Mark Shelton. Your feedback has significantly improved the book.
I would like to thank the folks at Addison-Wesley for their continued support. Peter Gordon, my editor, has been a thoughtful guide through five books. Kim Boedigheimer continues to work behind-the-scenes magic to make things happen. The production team Julie Nahil, Megan Guiney, and Diane Freed did their usual fine work in making this book a reality.

Finally, I would like to thank my wife, Maria, and my children, Jessica and Philip, for their love and support.
About the Author

Dr. Paul C. Brown is a Principal Software Architect at TIBCO Software Inc. His work centers on enterprise and large-scale solution architectures, the roles of architects, and the organizational and management issues surrounding these roles. His total architecture approach, the concurrent design of both business processes and information systems, can reduce project duration by 25 percent. He has architected tools for designing distributed control systems, process control interfaces, internal combustion engines, and NASA satellite missions. Dr. Brown is the author of Succeeding with SOA: Realizing Business Value Through Total Architecture (2007), Implementing SOA: Total Architecture In Practice (2008), TIBCO® Architecture Fundamentals (2011), Architecting Composite Applications and Services with TIBCO® (2012), and Architecting Complex-Event Processing Solutions with TIBCO® (2014), all from Addison-Wesley, and he is a coauthor of the SOA Manifesto (soa-manifesto.org). He received his Ph.D. in computer science from Rensselaer Polytechnic Institute and his BSEE from Union College. He is a member of IEEE and ACM.
This page intentionally left blank
This page intentionally left blank
Chapter 3

CEP Solution Design Patterns

Objectives

There are many different architectural patterns that arise in complex-event processing (CEP) solutions. While all add one or more sense-analyze-respond processes to the enterprise, the manner in which they do so varies widely. This chapter identifies the kinds of variation you can expect and presents a number of well-understood patterns, each of which addresses a common business challenge.

After reading this chapter you will be able to explain the variability in CEP architectures and describe the following patterns:

• Condition Detection
• Situation Recognition
• Track and Trace
• Business Process Timeliness Monitor
• Decision as a Service
• Situational Response
• Orchestrated Response

You will also be able to explain the challenges associated with pioneering projects that develop new solution patterns.
Variability in CEP Architectures

The core CEP process (Figure 3-1) is always the same: Some event is sensed, it is analyzed in the context of some reference data to determine whether something of business interest has occurred, and some decision is made about what the nature of the response ought to be. Yet despite the fact that the core process is always the same, there are many different architectures for complex-event processing. Why?

There are two dominant reasons for the variability in CEP architectures: the handling of reference data and the partitioning of functionality.

Handling Reference Data

The first area of variability centers around the relationship between the reference data and the events being analyzed: Does the stream of events alter the reference data that is used to interpret subsequent events? Applications in which the stream of events does not alter the reference data are relatively straightforward. The primary challenge in these applications is obtaining access to the reference data, which almost always originates elsewhere, and making access to the data efficient during the analysis and response activities.

On the other hand, applications in which the stream of events modifies the reference data are much more complicated. The portion of the reference data that represents the history of prior events does not have a system of record, at least without additional design. If it is unacceptable to lose this historical data when systems are shut down or fail, then the CEP solution must now include a system of record for the historical data. The system of record requires careful and clever design to ensure that it can handle the stream of data changes efficiently and robustly—and still make the data efficiently accessible to the analysis and response activities (Figure 3-2).

Figure 3-1: Core CEP Process
Partitioning Functionality

The other area of variability lies in the many ways in which the CEP functionality can be partitioned and assigned to different components. The basic partitioning found in CEP solutions is shown in Figure 3-3.

Generally, the events driving the process are the observable actions of a participant (human or system) in some business process. Most of these participants do not announce their activities, at least to components not engaged in that business process. For this reason, CEP solutions generally have one or more components dedicated to sensing these actions and announcing their observations.

The techniques used for these observations are the same ones traditionally used in application integration. These techniques, and the products that support them, are detailed in TIBCO™ Architecture
Fundamentals.1 The relevant observation here, however, is that the products used for sensing are, for the most part, not the products used for CEP analysis and response. Thus the participant that does the sensing is generally not the participant doing the analysis and response.

As a side note, one of the hallmarks of the event-enabled enterprise is that its architecture includes the types of components necessary to sense and announce actions and the types of components necessary to analyze and respond to those announcements.

In many cases, the volume of events handled by many CEP solutions makes it impractical to have one component handle all of the events and perform all of the analysis and response processing. Once this point is reached, there are a variety of ways in which performance can be increased. One is to simply deploy multiple instances of the component performing the analysis and response. This is a straightforward approach if the reference data is not updated when events occur. But when the reference data is updated by events, sharing the history across multiple instances of the analysis and response components requires additional design. The design patterns for this are discussed in Chapter 14.

Another approach to scalability is to begin to partition the functionality across additional components. Figure 3-4 shows one possible partitioning in which the analysis that leads up to situation recognition is performed by one component and the determination of the required responses is performed by another. Partitioning patterns also become more complex when the analysis and response computations also

---

update reference data. Chapter 13 discusses this and other partitionings as well as the tradeoffs that need to be considered.

As should be obvious by now, there are many possible functional partitionings for CEP solutions. Some lead to simple and straightforward implementations. Others require clear architectural thinking to achieve the desired behavior in a robust and highly scalable fashion.

The following sections discuss a number of CEP solution design patterns, each focused on providing a commonly required business capability. For the most part, the patterns are arranged somewhat in order of increasing complexity. The chapter concludes with a brief discussion of problems for which there may not be well-established design patterns.

For simplicity, the sensing component is not shown in these design patterns: It is assumed to be always present.

---

**Condition Detection**

The simplest solution pattern you will encounter in complex-event processing is threshold detection (Figure 3-5). In this pattern, a component takes an action that can be observed and results in a technical event. The condition detector is listening for this event, whose arrival serves as the trigger for analysis. The analysis compares a value conveyed by the event to a threshold value and, if the event value exceeds the threshold value, generates a business event announcing this condition. Completing the pattern, another component is listening for these announcements and taking appropriate actions.

**Figure 3-5: Threshold Detection Pattern**
In using this pattern the location of the threshold value must be considered. One option is to permanently fix the threshold value in the analysis logic. Another option is to make it a piece of contextual information that is looked up by the condition detector, either when it starts or each time an event is analyzed. Yet another option is to use infrastructure that makes it possible to change the value at runtime. TIBCO BusinessEvents® rule templates provide this capability, as described in Chapter 10.

The more general form of this pattern is the Condition Detection pattern (Figure 3-6). In this pattern the detected condition is defined by a number of values that define the boundaries of the condition being recognized. The information considered in the analysis is generally a combination of event and contextual data. If the condition is detected, then a business event is generated announcing the existence of the condition.

When using this pattern the sources of the parameters defining the boundary conditions and the contextual data required to detect the condition must be considered, along with the possible need to change some of these values at runtime. The design effort required to provide access to information originating in other systems and make it efficiently available is often a major part of a CEP project.

In the Condition Detection pattern, the reference data that is used is not modified by the processing of events: It does not reflect prior history. The only state information being used is that conveyed by the

![Figure 3-6: Condition Detection Pattern](image-url)
triggering event. This makes the condition detector stateless, and therefore easy to scale and make highly available.

**Situation Recognition**

The Situation Recognition pattern (Figure 3-7), on the surface, looks a lot like the Condition Detection pattern. However, there is a major difference: In the Situation Recognition pattern, the context data used to recognize a situation when the triggering event arrives contains historical information. Many of the triggering events that arrive do not result in a business event, but their occurrence results in the modification of the context data. The updated context data then provides the context for evaluating the next event that arrives.

Since the context data in this pattern contains historical information, the ability of the pattern to recognize a situation may be compromised if the historical data is lost. Such a loss would occur if the situation recognition component is holding context data in memory and the component is restarted. For this reason, the use of this pattern almost always requires persisting the historical information and recovering this information when the component restarts. The object persistence discussion in Chapter 6 discusses techniques for doing this.

There are many variations on this pattern both in the manner in which the context data keeps track of prior history and the manner in which the historical information is used to interpret a current event. Chapter 10 discusses a number of design patterns that can be used for this purpose.

![Figure 3-7: Situation Recognition Pattern](image-url)
Track and Trace

The Track-and-Trace pattern (Figure 3-8) is a special case of the Situation Recognition pattern. This pattern involves two contextual elements: a model of the expected process and the state of an existing instance of that process. If the triggering event marks the beginning of a new process execution, an initial process state is created. For other events, information in the event is used to locate the state of the process already being executed (there may be many instances of the process being executed at any given point in time). Once the current state has been identified, the process model is then used to interpret the triggering event in the context of that state.

This simplified example omits a common challenge: the handling of out-of-sequence events. In many real-world situations, events may arrive out of sequence. In some cases, the first event that arrives may not be the initial event in the process. In a full solution, additional logic must be added to handle these situations. Chapter 14 discusses some of the design considerations.

The state machine approach provides for a rich and varied interpretation of the process execution. If the triggering event corresponds to

![Figure 3-8: Track-and-Trace Pattern](image)
an expected transition in the state machine (given the current state), the conclusion is that the process is executing in an expected manner—at least at this time. The analysis can be designed to announce business events when particular states have been achieved (i.e., announce that a milestone has been reached).

If the triggering event does not correspond to an expected transition, something unexpected has happened. Again, the analysis can be designed to emit business events announcing this unexpected situation.

This type of analysis is appropriate for monitoring any type of unmanaged process. Tracking of a package from initial pickup to final delivery is one example. Tracking your luggage from the time you drop it off at the departure airport ticket counter until the time you pick it up at the baggage carousel at your final destination is another.

In general, this approach is well suited for monitoring any process in which there is a hand-off of responsibility from one participant to another. You give your luggage to the counter agent—one hand-off of responsibility. The counter agent places the bag on the conveyor as a means of handing off responsibility to the baggage handlers. The process continues until the final hand-off, which begins when the baggage handler at your final destination places the bag on the conveyor leading to the baggage carousel and ends when you pick up your luggage.

The events being monitored in track-and-trace situations are the evidence that individual hand-offs have been successful. The challenge in most situations is finding the evidence. In the days before security requirements mandated scanning and tracking luggage on airplanes, the evidence was scanty: You got your receipt for your bag when you dropped it off (that is, when you handed it off to the airline) and you (hopefully) picked up your bag at its destination. There was little evidence available for any intermediate progress.

The security requirement that luggage not travel on a plane unless the associated passenger is also on board has resulted in better tracking—better evidence—of your luggage’s progress. The luggage tracking tag is scanned when the luggage is loaded on the plane or placed in a bin that will subsequently be loaded on the plane. It is scanned again when it comes off. These scans provide intermediate evidence of progress.

Your challenge in designing a Track-and-Trace solution is going to be finding appropriate evidence of progress. It is not uncommon that the full set of evidence you would like to have is simply not available. When this occurs, you may want to implement the degree of tracking that is supported by the currently available evidence and
independently begin an initiative that will eventually provide more detailed evidence of progress. This is exactly what happened in the telecommunications case study described back in Chapter 2.

**Business Process Timeliness Monitor**

The Business Process Timeliness Monitor (Figure 3-9) is an extension of the Track-and-Trace pattern. State machine models can be extended so that the absence of an expected event within some period of time can be recognized. While, of course, you can apply this approach to recognizing that an overall process did not complete on time, the greatest benefit comes from recognizing that some intermediate event did not occur on time, and thus the overall process is in jeopardy of being late. The recognition can be used to trigger an action that will correct the course of the overall process and get it back on track for an on-time completion. The telecommunications case study discussed back in Chapter 2 is an example of this pattern in action.

Detecting the absence of an event requires the establishment of a service-level agreement specifying the maximum amount of time it should take for the process to complete or remain in each intermediate state. When the state machine monitoring the process is started or a particular intermediate state is entered, a timer is started. When the overall process completes, or the intermediate state is exited, the corresponding timer is stopped. However, if the timer expires before the

![Figure 3-9: Business Process Timeliness Monitor](image-url)
process completes or the intermediate state is exited, a timeout event is generated. This is an indication that some expected event did not occur.

In recognizing this situation, it is the expiration of the timer that serves as the trigger for the analysis. Some introspection of the state machine may be required to identify which events did not occur, but the larger design requirement is to determine which parties should be notified when this situation arises and what actions those parties are going to take to get the overall process back on track.

**Situational Response**

All the patterns in this chapter up to this point have had one characteristic in common: They simply recognize that some condition exists and announce that fact with an event. Other independent participants receive these notifications and decide what action to take.

In some situations there is an additional challenge in determining what the appropriate response ought to be (Figure 3-10). Further analysis is required, generally to focus the actions on achieving specific business objectives. Reference data, often containing historical information, is required for the analysis. The result of the analysis is generally one or more directives to actually perform the identified actions.

Consider the case in which there is some form of perishable commodity being sold: fresh produce and meat, seats on a plane, or hotel rooms—anything that becomes worthless if not sold by some point in

---

**Figure 3-10: Situational Response Pattern**
time. The desired business strategy is to dynamically set the price of the commodity based on the remaining inventory and the time remaining before the commodity becomes worthless. The situation being responded to in these cases is the presence of a potential consumer for the perishable commodity.

The simplistic approach to pricing the commodity is to fix a point in time at which it will be put on sale. The idea is that this will raise demand and ensure that the commodity does not go to waste. The problem with this approach is that it neither maximizes revenue nor minimizes the likelihood that the commodity will go to waste. If the commodity is selling well and will likely sell out, putting it on sale will result in lost revenue. On the other hand, if the commodity is selling very poorly, lowering the price by a set amount at a fixed point in time might not ensure that the commodity actually sells out.

A more sophisticated approach is to track the rate at which the commodity is selling versus the price of the commodity. With this approach, the offering price for the commodity can be adjusted dynamically. This approach is often applied to online product sales. It requires complex-event processing to do the dynamic price adjustments as consumers shop and as commodity inventories change. Note that the rate of sales and the current inventory become part of the reference data—a dynamic part whose currency must be maintained in a timely manner—most likely via events!

**Decision as a Service**

In the Decision-as-a-Service pattern (Figure 3-11), the logic necessary to make a decision is factored into a separate component. The service consumer gathers all relevant current-state input data for the decision and passes it to the service. This is typically a synchronous request-reply interaction, but it may be asynchronous. In either case, the decision service computes output data from the input data, using static reference data as appropriate. The output data reflects the decision results.

The value of this pattern is that it encapsulates the logic of the decision as a service. This simplifies the maintenance of both the service consumer and the decision service. In particular, it allows the implementation of the service (that is, the business rules) to be updated without requiring a modification to the service consumer.
To make this possible, however, both the input and output data structures have to remain fixed.

Let’s consider an example from the banking world. A bank needs to evaluate applications for credit cards to determine whether a credit card should be issued and what the credit limit should be on the account. In this case, the same data structure is used for both the input and output, with the difference being that some of the field values are computed by the Credit Card Decision service. Figure 3-12 shows this data structure. The input data includes the applicant’s age, credit score,
a flag indicating whether or not the applicant has a driver's license, another flag indicating whether they are married, and the applicant’s income. The computed output values comprise a Boolean indicating whether the applicant is eligible, a field indicating the current status, and another field indicating the credit limit should the status be accepted.

A decision table describing the logic for this service is shown in Figure 3-13. This example is developed using the TIBCO BusinessEvents® Decision Manager, which is described in Chapter 10. Each line of the table defines a set of conditions for the input values (the Condition Area) and the corresponding computed output values (the Action Area).

The Decision-as-a-Service pattern is useful when the business rules change frequently but the data used to drive the decision and the outputs of the decision can be fixed.

Orchestrated Response

While process orchestration is not a traditional focus of complex-event processing, the need to orchestrate portions of CEP solution activity is increasing in importance (Figure 3-14). In this relatively common
pattern, process orchestration is used to coordinate multiple participants in responding to a situation. The reason for the orchestration is twofold: to control the order in which the actions are performed and to confirm the successful completion of the actions. Less common is a situation in which process coordination is required for situation recognition.

This pattern is a hybrid of event-driven and request-driven interactions. All of the interactions up to the receipt of the situation recognition announcement are event driven. The response orchestration component, however, uses request-driven interactions to not only request that each participant perform its work but also to confirm the successful completion of that work.

When this pattern is used, a choice must be made regarding the type of technology to be used for the response orchestration. Traditionally, this would be a component designed specifically for process orchestration, such as TIBCO ActiveMatrix BusinessWorks™ or TIBCO ActiveMatrix® BPM. With this approach, if rule-based reasoning is required in the orchestration, the Decision-as-a-Service pattern is used. The service returns values that then guide the subsequent process execution.
However, separating process orchestration from complex-event processing may become a performance barrier, particularly if a significant amount of repetitive information must be passed to the decision service on each invocation. In such cases, it is better to have the process orchestration performed directly by a CEP component. This is the purpose of the TIBCO BusinessEvents® Process Orchestration product. It adds process orchestration capabilities to TIBCO BusinessEvents®.

### Pioneering Solutions

We close this chapter on a cautionary note. Early explorers drew maps of the territories they became familiar with and drew dragons in the unexplored corners of these maps, warning those later map readers to beware of those unexplored spaces. Even worse, many explorers never even reached their goals: Columbus was seeking Asia when he found the Americas, and numerous explorers sought unsuccessfully for the Northwest Passage that would provide a North American route from the Atlantic to the Pacific.

The relevance here is that there are many types of applications for complex-event processing that have been well explored. If you are working in one of these areas, the problem is well defined, and implementing your solution will be a straightforward engineering exercise. If, however, you are working in an area that is not well defined, one in which the analytical approach for either situation recognition or action determination has not yet been established, proceed with caution. Some (but not all) of these areas are true research topics—you need to invest a little time in determining whether or not your particular problem is well defined before you commit to building a solution. Remember, it took more than 400 years to find the Northwest Passage!

How can you tell when you are on safe ground? Ask yourself the following questions:

- Is the information related to the problem understood well enough to create a quality information model (including relevant state information)?
- Is there a well-defined (i.e., measurable) set of criteria that defines the situation that needs to be recognized?
- Are there well-defined triggers that identify the points in time at which the situation recognition analysis will be performed?
• Is the information necessary for this recognition analysis readily accessible?
• Is there a clearly articulated approach for using the available information to recognize the situation?
• Is there a well-defined (i.e., measurable) approach for responding to the situation once it has been recognized?
• Is the reference information needed for determining the response readily accessible?
• Does the business value of the resulting situation recognition and response capabilities warrant the investment in the solution?

If you answered yes to all of these questions, you are on solid ground. If you answered no to any of them, you may be plowing new ground. You need to eliminate this uncertainty before you commit to producing a solution. Focus your initial efforts on developing the answers to these questions, with particular attention to the last one: Is the result worth the effort? Then, and only then, should you commit to building a solution.

The riskiest question in the list is the first: What is it that you are trying to recognize? Define your goals based on solid analytical results and beware of open-ended criteria. For example, you are never going to recognize all forms of financial fraud: The bad guys are constantly inventing new ways to scam the financial system and circumvent the checks currently in place. Identifying fraud, in general, is not an achievable goal.

On the other hand, there are specific behavior patterns that fairly reliably indicate that there might be fraud in progress. An analysis of login patterns might identify these behavior patterns, and the recognition of these patterns as they occur is definitely a well-defined and measurable goal.

If you find yourself waving your hands as you attempt to get specific about defining your recognition goals—stop! You are treading on thin ice. Do your analytical homework and convince yourself that you can be precise about what is to be recognized.

Summary

There are two factors that contribute to the variability in complex-event processing architectures. One is the handling of reference data and the extent to which the stream of events modifies the reference data used to
interpret subsequent events. The other is the myriad ways in which the necessary sense, analyze, and respond activities can be partitioned and assigned to components. There is no one-size-fits-all architecture for complex event processing.

The simplest architectures are those in which the reference data is not impacted by the stream of events. The Threshold Detection and Condition Detection patterns are examples of these.

When the event stream can alter the reference data, the architecture gets a bit more complicated. The reference data now contains some historical information. If this information is essential for analysis, the solution must now become a system of record for this information. This requires persisting the information.

The Situation Recognition pattern uses historical data in its analysis. Some of the events that arrive simply result in updates to the historical data. Others, when analyzed, signify the recognition of a business-significant condition that must be announced. Track-and-Trace is a specialization of this pattern that does milestone-level tracking of a process. The Business Process Timeliness Monitor extends Track-and-Trace to determine whether the milestones are achieved on time.

Some applications require more than simply announcing that a condition exists. The Situational Response pattern applies contextual analysis to determine the actions that are required in a specific situation. The Decision-as-a-Service pattern makes these analytical capabilities available to non-CEP components. Sometimes the requirement extends beyond simply identifying the required actions to include the management of their execution. The result is the Orchestrated Response pattern.

Building a solution in which the situations to be recognized, the desired responses, and the analytical techniques to be used are all well defined is a straightforward (though sometimes complex) engineering exercise. Building a solution when any of these is not well defined has a significant degree of uncertainty. In these situations, before a commitment is made to produce a solution, preliminary work should be undertaken to clarify the approach to recognition and response. Once this preliminary work has been completed, an estimate of the effort required to implement the solution should be made to ensure that it is warranted by the expected business benefit.
This page intentionally left blank
Index

A
Account Change Recognition, 151–152
Actions
  partitioning situation recognition from response, 188–189
  rule agenda, 81
  rule clauses of inference agents, 78–79
  sequential and conditional, 157–160
  synchronous I/O calls in rule actions, 231
Active sensing, 6–7
ActiveMatrix
  BPM, 49–50, 159
  BusinessWorks. See BusinessWorks database adapter. See Adapter for Database
  integration with, 172
  process orchestration and, 49–50, 159
  resource information for, 171
  Service Grid, 204
ActiveMatrix BPM, 49–50, 159
ActiveSpaces
  channel types, 88
  Data Grid and, 160
Adapter for Database
  database interaction with, 182–183
  Inference Agent Publication pattern, 183
  Inference Agent Request-Reply pattern, 183
  Inference Agent Subscription pattern, 184–185
  updating reference data on external system, 147
Administrator, TIBCO, 56, 68
Agents
  cache agents. See cache agents
  dashboard agents. See dashboard agents
deadlocks and, 199
in deployment, 241–242
directing related work to single agent, 202–203
inference agents. See inference agents
inter-agent communications, 256
modularization and, 240–241
options for turning on profiler, 226
partitioning and, 188–189
process agents. See process agents
processing units and, 243
query agents. See query agents
TIBCO BusinessEvents, 56
types of, 63–64
Alerts, display options, 137
Analysis. See also sense-analyze-respond pattern
  approaches to event analysis, 25–26
capabilities of event-enabled enterprises, 31–32
CEP (complex-event processing) and, 17–19
  complex events requiring, 16–17
correlation in, 21
event-enabled enterprises and, 11
inference agents and, 69
innovations in, 8
interpreting, 236
  of performance, 234–236
rule-based, 19–20
Track-and-Trace pattern and, 43
  triggers for, 261
Architecting Composite Applications and Services with TIBCO (Brown), xx
Architecture
  best practices for planning, 259–260
  Nouveau Health Care case study, 213–214
  variability of CEP architectures, 36
Asynchronous Service Consumer pattern interaction with BusinessWorks, 185
  TIBCO BusinessEvents as service consumer, 175–178
attribute keyword, 77
Attributes, in rule clauses, 77
Automation, of sensing, 7
Availability. See also fault tolerance
    load distribution and, 201
    of solutions, 253

B
Backing store
    configuring for fault tolerance, 254
    configuring object management, 245–246
Backup Copies, replicating cache objects, 110
Behavior models, event correlation and, 21
BEMM (BusinessEvents Monitoring and Management)
    managing logging, 160
    overview of, 56
    solution life cycle and, 66–68
    turning on profiler, 226
Berkeley DB, 111
Best practices
    architecture planning, 259–260
    designing data models by concepts, 260–261
    for object management, 261
    overview of, 259
    for rule design, 261
    summary, 262–263
    testing, 262
BPM, 49–50, 159
BPMN (Business Process Modeling Notation)
    defining processes, 130
    managing process execution, 23
    metadata models for process description, 23
    plugin for TIBCO Business Events Studio, 62
Browser Interface, Decision Manager, 58
Buffers
    continuous queries operating on, 121
    managing, 122–123
Business Process Modeling Notation. See BPMN (Business Process Modeling Notation)
Business Process Timeliness Monitor pattern, CEP design patterns, 44–45
Business processes. See processes
Business rules. See rules
BusinessEvents
    as asynchronous service consumer, 175–178
    BusinessWorks Receive Event as process starter, 173–174
    core features, 56–57
    Data Modeling, 57–58
    Decision Manager. See Decision Manager
    deploying solutions, 63–65
Event Stream Processing. See Event Stream Processing
Load Balancer, 202
managing logging, 160
metadata and, 23
overview of, 55, 56
Process Orchestration. See Process Orchestration
product suite, 55–56
Profiler, 225–226
reference data change coordination patterns, 147–148
request-reply interaction with ActiveMatrix BusinessWorks, 173–174
rule templates, 22, 40, 154–155
as service provider, 174–175
solution life cycle, 65–68
solution life cycle and, 66–68
Studio. See Studio
summary, 68–69
support for computational specializations, 19
as synchronous service consumer, 178–180
as system of record for reference data, 145
turning on profiler, 226
variety in feature set, 17
Views. See Views
BusinessEvents Monitoring and Management. See BEMM (BusinessEvents Monitoring and Management)
BusinessWorks
    database interaction with, 185
    for filtering and enrichment, 191
    integration with, 172
    process orchestration and, 49–50, 159
    Receive Event activity, 173–174
    resource information for, 171
    Send Event activity, 172
    Wait for Event activity, 173

C
Cache
    cache agents and, 104
    example of use of, 106–109
    factors contributing to need for, 103
    fault tolerance and, 254
    for information sharing, 196–198
    interpreting performance analysis, 236
    recognizing changes to cached objects, 148–149
Cache Agent Quorum, 110
Cache agents
    analysis of Claim Tracker, 235
    cache and, 104
    Cache + Memory mode, 106
    Cache Only mode, 105
Index 267

in core of TIBCO BusinessEvents, 56
deploying process agents and, 132
element of use of, 106–109
factors contributing to need for cache, 103
functional data modeling extensions
and, 57
Memory Only mode, 105–106
modularization and, 242
object locking, 109–110
object management modes, 104–105
object persistence, 111
object replication, 110
processing units and, 243
Shared-All persistence option, 111–113
Shared-Nothing persistence option, 113
sizing rules of thumb, 237
snapshot query execution and, 115
solution deployment and, 63–64
summary, 113–114
supplying buffer of continuous queries, 121
timeout notification and, 236
Cache-Aside behavior, 112
configuring backing store, 246
update strategy, 111–112
Cache + Memory mode
database concept configuration, 181
object management modes, 106
Cache Only mode
configuring object management, 244–246
database concept configuration, 181
information sharing, 196–198
object management, 105
process management, 131
Call Activity, 130
Caller threads, threading models for inference agents, 230
Catalog functions, turning on profiler, 226
CDD files
logging and exception reporting, 160
object management modes and, 104–105
solution life cycle and, 65–66
CEP (complex-event processing)
correlation in, 21
feasibility and, 25–26
functional partitioning, 37–39
handling reference data, 36–37
metadata, 23
overview of, 17–20
responding to complex events, 26–28
variability of architectures, 36
CEP design patterns
Business Process Timeliness Monitor, 44–45
Condition Detection, 39–41
creating new solutions, 50–51
Decision-as-a-Service pattern, 46–48
functional partitioning, 37–39
handling reference data, 36–37
overview of, 35
Response Orchestration, 48–50
Situation Recognition, 41
Situational Response, 45–46
summary, 51–52
Track-and-Trace pattern, 42–44
variability of CEP architectures, 36
Change management
coordinating changes to reference data, 147–148
recognizing changes to cached objects, 148–149
recognizing changes to continuous queries, 151
recognizing changes to state machines, 149–150
runtime rule changes, 154
Channels
communication between clusters, 243
destinations, 89
snapshot queries and, 115
supplying buffer of continuous queries, 121
threads associated with channel types, 92
for transport of services, 174–175
types of, 88–89
Charts, display options for metric data, 137
Checkpoints, process agent behavior and, 131
Claim processing, in Nouveau Health Care case study
analysis, 234–236
Claim Status Notification, 217
Claim Tracker, 217–218
Claim Tracker Interface, 219–221
Monitor Claim Processing process, 219–221
Obtain Claim Status process, 223–224
Claim Status Request, triggering claim processing events, 234
Claim Status Timeout, 234
Claim Tracker
agent modularization and, 242
configuring object management, 246
deployment patterns and, 247
monitoring and, 249–250
overview of, 217–218
triggering events in, 233–234
Claim Tracker Interface, 219–221
Clusters
deployment and, 243–244, 248
modularization and, 240
Communication
inter-agent, 256
telecommunication service restoration case, 13–14
Competitive advantage
event-enabled enterprises and, 9
by improving business processes, 5
Complex events
complex event processing. See CEP (complex-event processing)
defined, 14
deploying, 239
design challenges, 195
event categories, 15
feasibility of analysis, 25–26
integration and, 171
overview of, 16–17
processing generally, xvii
processing with TIBCO BusinessEvents, 55
recognizing situations that require action, 143
responding to, 26–28
Computation
in analytic approach, 25
metrics and, 136
Concept data structures, in inference agents
actions and, 78
designing data models by, 260–261
examples, 74–75
integration using database concepts, 181
join conditions and, 261
Condition Detection pattern, CEP design patterns, 39–41
Conditions
action performance, 157–160
avoiding complex, 261
checkpoints and, 131
filter conditions, 115–117
Rete network efficiency, 83–87
rule clauses of inference agents, 78
rule conditions for accessing XML event payloads, 228–229
structuring rule conditions for performance, 227–228
types of rule conditions, 227
Configuration
of backing store, 254
of dashboard agents, 135–136
database concept, 181
object management, 244–246
run-time configuration requirements, 248–249
Constants, as thresholds for decision making, 22
Content-aware load balancing, 203
Context
Condition Detection pattern and, 40
event analysis requiring, 23–24
of events, 21–23
Situation Detection pattern and, 41
Track-and-Trace pattern and, 42
Continuous queries
buffer management, 122–123
Continuous Query Change Recognition pattern, 151
event analysis requiring, 23–24
information sharing, 197
life cycle, 123–124
overview of, 121–122
Coordination patterns, fault tolerance and, 254–256
Correlation, of events, 20–21
Credit-card fraud, CEP example, 18–19
Custom functions, defining, 180
Customers, extreme value converting into fans, 4
D
Dashboard agents
behavior of, 136
configuring, 135–136
deploying, 139
display options, 137
information sharing, 197
metrics, 136–137
overview of, 135
solution deployment and, 63–64
summary, 139–140
TIBCO BusinessEvents Views, 63
TickerTracker example, 138–139
Data

- acquisition for analysis, 25
- aggregation, 136
- as context for analysis, 22
- innovations in sensing and data quality, 7–8

Data Grid, ActiveSpaces, 160

Data modeling

- designing models by concepts, 260–261
- state machine modeling, 57–58
- using database as system of record for reference data, 146

Data structures

- Claim Status example, 220
- concepts, 73–75
- events, 71–73
- locks and, 199–200
- overview of, 70–71
- scorecards, 75–76

Databases

- concepts and memory management, 181
- integration using ActiveMatrix Adapter for Database, 182–183
- integration with, 180–181
- object persistence options, 111
- queries, 181
- as system of record for reference data, 146
- updates and deletes, 182

DB2, 111

Deadlocks. See also locks, 199

Decision-as-a-Service pattern

- in decision making, 46–48
- decision tables, 155–156
- process orchestration and, 49

Decision Manager

- Decision-as-a-Service pattern and, 48
- overview of, 58–60
- runtime rule changes, 154

Decision tables

- defining processes and, 130
- organizing for performance, 228
- for virtual rules, 155–156

Decision trees, visualization with TIBCO BusinessEvents Studio, 228

Declaration, in rule clauses, 77–78

declare keyword, 77–78

Dedicated worker pool thread, 230

Delegation with Confirmation, coordination patterns, 159

Deletes, database, 182

Demand analysis, 232–233

Deployment

- agents for, 241–242
- clusters in, 243–244
- configuring object management, 244–246

of dashboard agents, 139

modularization units in, 240–241

monitoring, 249–250

overview of, 239

of patterns, 167, 247–248

of process agents, 132

processing units and, 242–243

requirements for run-time configuration, 248–249

summary, 250–251

TIBCO BusinessEvents solutions, 63–65

Design

- best practices for rule design, 261
- of cache for information sharing, 196–198
- CEP patterns. See CEP design patterns
directing related work to single agent, 202–203

for duplicate event handling, 206–207

for information sharing, 195–196

for load distribution, 201–202

for locking, 198–200

overview of, 195

for performance, 226

for sequencing, 203–206

summary, 207

Destinations

- channels and, 89

preprocessing and, 92

selecting threading models by, 261

sending events to, 172–173

Disaster recovery. See also fault tolerance, 256–257

Display options, dashboard agents, 137

Duplicate events, handling, 151–153, 206–207

E

EAR files

- RMS generating, 156–157

solution life cycle and, 65–66

EMS (Enterprise Message Service)

- analysis of Claim Tracker, 235
- deployment of Claim Tracker, 248
- interpreting performance analysis, 236

resource information for, 171

solution life cycle and, 65

timeout notification and, 236

Enhancement/enrichment

- BusinessWorks used for, 191

partitioning from rule processing, 190–191

Enterprise architects, intended audience for this book, xxi

Enterprise Message Service. See EMS (Enterprise Message Service)

Equivalent joins

- inference agent conditions, 78

- types of rule conditions, 227
Event data structure, inference agent actions and, 78–79
overview of, 70
timeToLive, properties, and payload, 71–73

Event-driven processes
demand analysis and, 232
overview of, 28–31
partitioning and, 188

Event-enabled enterprises
capabilities of, 31–32
delivery of extreme value and, 3–4
innovations in analysis, 8
innovations in response, 9
innovations in sensing, 6–8
overview of, 3
sense-analyze-respond pattern, 5–6
summary, 10
what it is, 9–10

Event pattern recognition
monitoring liveness of components, 168–169
need for, 163–165
overview of, 163
pattern language in Event Stream Processing, 166
putting patterns to use, 166–168
summary, 169

Event Stream Processing
event pattern recognition, 163, 165
overview of, 60–61
pattern language in, 166
query agents and, 115

Events
accessing XML event payloads, 228–229
analysis requires context, 23–24
analytical approaches, 25–26
capabilities of event-enabled enterprises, 31–32
categories of, 14–15
communicating between blocks of rules, 260
complex. See complex events
complex-event processing. See CEP
(context of, 21–23
correlation of, 20–21
defined, 12
directing events created in preprocessor functions, 90–91
event-driven processes, 28–31
handling duplicate, 151–153, 206–207
for information sharing, 196
for intermediate conclusions, 261
missing or failing to recognize, 15–16
modularization and, 241
processing TIBCO BusinessEvents, 56
recognizing, 12–14
responding to, 26–28
rule-based analysis, 19–20
summary, 32–33

Excel spreadsheets, 58

Exception handling
Exception Reporting Interface, 221–222
Resolve Claim Processing Exception, 217, 221
solutions, 160

Expected behavior, CEP metadata, 23

Explicit buffer management, continuous queries, 123

External system, as system of record for reference data, 146–147

Extreme value, successful enterprises delivering, 3–4

F
Fault tolerance
configuring backing store for, 254
coordination patterns and, 254–256
inter-agent communications for, 256
object replication providing, 111
summary, 257

File systems
analysis of Claim Tracker, 235
timeout notification and, 236

Filters/filtering
BusinessWorks for, 191
inference agent conditions, 78
partitioning from rule processing, 190–191
snapshot query execution and, 115–117
structuring rule conditions for performance, 261
types of rule conditions, 227
Fire and Forget coordination pattern, 159
Forgy, Charles L., 80

G
Global locking, performance impact of, 229

Graphs
dashboard agent and, 135
display options for metric data, 137

H
Hawk
channel types, 88
Hawk agent in MM (Monitoring and Management), 66
Hawk agent in monitoring, 250
Hawk agent starting/stoping processing units, 68
managing logging, 160
options for turning on profiler, 226
High availability. See also fault tolerance request-reply coordination patterns in, 254–255
of solutions, 253
Historical data, recent history as context for analysis, 22
Hot Deployment, 249
HTTP
channel types, 88
custom function invocation, 180
IP redirectors for load distribution, 201
multithreaded channels and, 230
sendRequest() for interacting with, 179
TIBCO Business Events as service provider for, 174
HTTP Send Request Invocation, 179
I
I/O calls, synchronous I/O calls used in rule actions, 231
Implicit buffer management, for continuous queries, 123
Inference agents
analysis of Claim Tracker, 235
channels, 88–89
concept data structure and, 73–75
ConditionsSeparate example, 81–84
in core of TIBCO BusinessEvents, 56
data structures used by, 70–71
defining processes, 130
deploying patterns, 167
deploying process agents, 132
destinations, 89
directing events created in preprocessor functions, 90–91
event data structure and, 71–73
functional data modeling extensions and, 57
Inference Agent Publication pattern, 183
Inference Agent Request-Reply pattern, 183
Inference Agent Subscription pattern, 184–185
interpreting performance analysis, 236
lacking fault tolerance, 254
Load Balancer and, 202
LocalChannel example, 94–98
locks and, 109, 198
overview of, 69–70
partitioning and, 190–191, 241
postprocessing behavior, 93–94
preprocessing and postprocessing, 87–88
preprocessing behavior, 91–92
preprocessor functions, 90
preserving sequences across multiple, 205
preserving sequences within single, 204
processing timeouts, 235–236
RTC (run-to-completion) behavior, 79–81
rule actions, 78–79
rule attributes, 77
rule conditions, 78
rule conditions and Rete network efficiency, 83–87
rule declaration, 77–78
scorecard data structure and, 75–76
sizing rules of thumb, 237
solution deployment and, 63–64
starting/stopping state machines, 99
state models, 98
state transitions, 98–99
summary, 100–101
threading model for, 229–231
timeouts, 99
updating reference data on external system, 147
working memory, 103
Information
cache for sharing, 196–198
events for sharing, 196
planning architecture and, 259–260
sharing, 195
Innovations
in analysis, 8
extreme value and, 5
in response, 9
in sensing, 6–8
Integration
with ActiveMatrix Adapter for Database, 182–183
with ActiveMatrix BusinessWorks, 172
BusinessWorks Send Event activity, 172
BusinessWorks Wait for Event activity, 173
database concepts and memory management and, 181
database queries and, 181
database updates and deletes and, 182
with databases, 180–181
Inference Agent Publication pattern, 183
Inference Agent Request-Reply pattern, 183
Inference Agent Subscription pattern, 184–185
invoking rule function for, 174
overview of, 171–172
summary, 183–186
TIBCO BusinessEvents as asynchronous service consumer, 175–178
Integration, continued

TIBCO BusinessEvents as service provider, 174–175
TIBCO BusinessEvents as synchronous service consumer, 178–180

Interval events, 15
IP redirectors, for load distribution, 201
IPhone, example of extreme value, 3–4

J
Java, defining custom functions, 180
Java virtual machines (JVMs)
solution deployment and, 63
threading models and, 230

JDBC, 182

JMS (Java Message Service)
channel types, 88
coordination patterns in fault tolerance, 255
handling duplicate events, 151–153, 206–207
inter-agent communications, 256
preserving sequences across multiple inference agents, 205
preserving sequences within single inference agent, 204
queues for load distribution, 201
using TIBCO Business Events as service provider, 174

Join conditions
concepts and, 261
inference agent conditions, 78
types of rule conditions, 227

JVMs (Java virtual machines)
solution deployment and, 63
threading models and, 230

L
Life cycle
continuous queries, 123–124
of patterns, 168
snapshot queries, 117–118
TIBCO BusinessEvents, 65–68
Liveness monitoring, 168–169

Load Balancer, TIBCO BusinessEvents, 202
Load distribution
content-aware load balancing, 203
IP redirectors for, 201
JMS queries for, 201
TIBCO BusinessEvents Load Balancer, 202

Local
channel types, 88–89
LocalChannel example, 94–98

Locks
avoiding deadlocks, 199
data structures and, 199–200
establishing locking policy for objects, 261
information sharing and, 197–198
object locking, 109–110
overview of, 198
performance and, 229

Logging, 160

M
Maintenance, modularization for ease of, 260
Manage Payments process, Nouveau Health Care case study, 212–214
Memory management, databases, 181
Memory Only
configuring object management, 244–246
as default memory management mode, 181
overview of, 105–106
transient concepts and, 261
Messages
checkpoints and, 130
defining processes and, 130
Metdata, 25
Metric fields, 136
Metrics
dashboard agents, 136–137
determining for monitoring, 249
display options, 137
planning architecture and, 259
Microsoft Excel spreadsheets, 58
MM (Monitoring and Management)
managing logging, 160
overview of, 56
solution life cycle, 66–68
turning on profiler, 226

Modularization
in deployment, 240–241
ease of maintenance and, 260
Modularization patterns
BusinessWorks used for filtering and enrichment, 191
overview of, 187
partitioning advantages/disadvantages, 192
partitioning filtering and enhancement from rule processing, 190–191
partitioning rules of thumb, 192
partitioning situation recognition from response, 188–189
summary, 193
Monitor Claim Processing process,
Nouveau Health Care case study, 212–213, 217, 222, 242

Monitoring
deployment, 249–250
liveness monitoring, 168–169
Monitor Claim Processing process, 212–213, 217, 222, 242
Track-and-Trace pattern and, 43
Monitoring and Management. See MM
(Monitoring and Management)

N
Naming guidelines, for rules, 160–161
Networks
analysis of Claim Tracker, 235
interpreting performance analysis, 236
Rete networks, 81, 83–87
timeout notification and, 236
Nonequivalent join conditions, 78, 227
Notification events
Claim Status example, 222, 233–235
timeouts, 235–236
Nouveau Health Care case study
architecture pattern, 213–214
business processes, 212–213
claim processing, 215–216
Claim Status Notification, 217
Claim Tracker, 217–218
Claim Tracker Interface, 217–221
Monitor Claim Processing process, 222
Obtain Claim Status process, 223–224
overview of, 211–212
summary, 224

O
Object Data Management Group, 61
Object Query Language (OQL), 61
Objects
best practices for managing, 261
Cache + Memory mode, 106
Cache Only mode, 105
configuring object management, 244–246
locking, 109–110
locking policy for, 261
management modes, 104
Memory Only mode, 105–106
recognizing changes to cached objects, 148–149
replication of cache objects, 110
Shared-All persistence option, 111–113
Shared-Nothing persistence option, 113
Obtain Claim Status process, 217, 223–224, 242
ODBC, 182
Online examples in this book, accessing, xviii–xix
OQL (Object Query Language), 61
Oracle DB, 111
Orchestrated Response pattern. See Response Orchestration pattern
Out-of-sequence events, handling, 42

P
Partitioning. See also modularization patterns
advantages/disadvantages, 192
deployment and, 241
filtering and enhancement from rule processing, 190–191
rules of thumb, 192
situation recognition from response, 38–39, 188–189
variability of CEP architectures and, 37–39
Pattern language
in Event Stream Processing, 166
pattern recognition and, 164–165
putting patterns to use, 166–168
Pattern.IO.toPattern() function, 167
Patterns
Asynchronous Service Consumer pattern, 175–178, 185
CEP. See CEP design patterns
Continuous Query Change Recognition pattern, 149–151
Delegation with Confirmation coordination patterns, 159
deploying, 247–248
Inference Agent Publication pattern, 183
Inference Agent Request-Reply pattern, 183
Inference Agent Subscription pattern, 184–185
match, 60–61
modularization patterns. See modularization patterns
recognizing event patterns. See event pattern recognition
reference data change coordination patterns, 147–148
Reference-Data Comparison pattern, 144
Request-Reply coordination patterns, 254–255
sense-analyze-respond pattern. See sense-analyze-respond pattern
Synchronous Service Consumer pattern, 178–180
Payload
accessing XML event payloads, 228–229
of inference agent event data structure, 72
Performance
analysis of, 234–236
demand analysis, 232–233
Performance, continued
- design choices impacting, 226
- interpreting analysis, 236
- load distribution and, 201
- locking and, 229
- organizing decision tables, 228
- overview of, 225
- profiles, 225–226
- sizing rules of thumb, 237
- structuring rule conditions, 227–228
- summary, 237–238
- synchronous I/O calls used in rule actions, 231
- threading model for inference agents and, 229–231
- triggering events, 233–234
- XML event payloads and, 228–229

Persistence
- backing store providing, 245–246
- overview of object persistence, 111–113
- persistent event pattern for fault tolerance, 256

Point events, 15

Postprocessing
- behavior, 93–94
- RTC cycle and, 87–88

Preprocessing
- behavior, 91–92
- directing events created in preprocessor functions, 90–91
- object locking and, 109
- of patterns, 167
- preprocessor functions, 90
- RTC cycle and, 87–88
- snapshot query example, 119–120

Process agents
- behavior of, 130–132
- defining processes, 130
- deploying, 132
- intended utilization of, 127–129
- lacking fault tolerance, 254
- overview of, 127
- Process Orchestration and, 61
- solution deployment and, 63–64
- summary, 132–133

Process Claim, Nouveau Health Care case study, 212–213

Process Maintains Asynchronous State, 178

Process Orchestration
- adding process orchestration to TIBCO BusinessEvents, 50
- options for orchestration of actions, 159–160
- overview of, 61–62
- Web Service invocation, 179–180

Processes. See also process agents
- BusinessWorks Receive Event as process starter, 173–174
- defining, 130
- event-driven, 28–31, 188, 232
- for improving competitive advantage, 5
- managing execution of, 23
- managing in Cache Only mode, 131
- in Nouveau Health Care case study, 212–213, 221–224
- orchestrating. See Process Orchestration
- request-driven, 28–29
- Timeliness Monitor pattern, 44–45

Processing units
- cache agents and, 104
- in clusters, 240
- deployment and, 242–243
- Hot Deployment and, 249
- sending event to destination in, 172–173
- solution deployment and, 63
- TIBCO BusinessEvents and, 56
- Views and, 135

Product suite, TIBCO BusinessEvents, 55–56

Profiler, TIBCO BusinessEvents, 225–226

Profiles, performance, 225–226

Project architects, intended audience for this book, xx

Properties, of inference agent event data structure, 72

Queries
- CEP metadata and query structure, 23
- continuous. See continuous queries database, 181
- Event Stream Processing and, 60–61
- snapshot. See snapshot queries

Query agents
- buffer management in continuous queries, 122–123
- continuous queries, 121–122
- continuous query example, 124–126
- continuous query life cycle, 123–124
- Event Stream Processing and, 61
- information sharing, 197
- lacking fault tolerance, 254
- overview of, 115
- sizing rules of thumb, 237
- snapshot queries, 115
- snapshot query example, 118–121
- snapshot query execution, 115–117
- snapshot query life cycle, 117–118
- solution deployment and, 63–64
- summary, 126
Index 275

Query.create() function, 117, 123
Query.Statement.close() function, 124
Query.Statement.execute() function, 118, 123
Query.Statement.open() function, 118, 123

R
Ranadivé, Vivek, 3, 8
RDBMS
memory management and, 181
updates and deletes and, 182
Real time analysis
innovations in analysis, 8
TIBCO BusinessEvents Views, 62–63
Recognition. See sensing
Reference data
cchange coordination patterns, 147–148
as context for analysis, 22
correlating events with, 21
innovations in analysis and, 8
Reference-Data Change pattern, 148
Reference-Data Comparison pattern, 144
Situational Response pattern, 45
system of record, 145–147
variability of CEP architectures, 36–37
Reference-Data Change pattern, 148
Reference-Data Comparison pattern, 144
Relational databases, data modeling and, 57–58
Rendezvous
channel types, 88
solution life cycle and, 65
Replication
of cache objects, 110
configuring object management, 245
Repository. See cache
Request-driven processes
augmenting with event-driven, 30–31
comparing with event-driven, 28–29
Request Reply pattern
coordination patterns, 159–160
in high availability, 254–255
Inference Agent Request-Reply pattern, 183
Resolve Claim Processing Exception, 217, 221
Response. See also sense-analyze-respond pattern
capabilities of event-enabled enterprises, 31–32
decision as a Service pattern, 46–48
event-enabled enterprises and, 11–12
to events, 26–28
innovations in, 9
Response Orchestration pattern, 48–50, 127–129
Situational Response pattern, 45–46
Response Orchestration pattern, 48–50, 127–129
Responsibilities, separating in event-driven processes, 30
Results sets, queries, 118
Rete networks
overview of, 81
rule conditions and efficiency of, 83–87
RMS (Rules Management Server), 56, 156–157, 248
Route Claim, Nouveau Health Care case study, 212–213
RTC (run-to-completion) behavior, 79–81
choosing threading models, 229–231
executing inference agent rules, 70
locks for avoiding conflicts, 109–110
performance information regarding, 225–226
postprocessing behavior, 93–94
pre and postprocessing, 87–88
preprocessing behavior, 92
Rule-based analysis
overview of, 19–20
power and flexibility of, 187
process orchestration and, 49
Rule engines, 56
Rule templates, changing values at runtime, 22, 40, 154–155
Rules
accessing XML event payloads, 228–229
actions, 78–79
agenda, 80–81
attributes, 77
best practices for designing, 261
BPMN script executing, 130
conditions, 78
conditions and Rete network efficiency, 83–87
decision tables for virtual, 155–156
declarations, 77–78
event responses determined by, 27–28
invoking rule functions, 174
locking objects and, 229
naming guidelines, 160–161
orchestrating, 157–160
partitioning, 190–192
reference data change coordination patterns, 147–148
RMS (Rules Management Server), 156–157
time rule changes, 154
sequencing execution of, 158
Rules, continued
  structuring rule conditions for performance, 227–228
  synchronous I/O calls in rule actions, 231
  templates, 40, 154–155
  testing best practices, 262
Rules Management Server (RMS), 56, 156–157, 248
Run-to-completion. See RTC (run-to-completion)
Runtime
  configuration, 248–249
  rule changes, 154
  TIBCO Runtime Agent, 56
S
Scalability, partitioning and, 38
Scorecard data structure
  actions and, 78
  examples, 75–76
  Monitoring Scorecard, 249
  overview of, 71
  for summarizing activity, 75
Scripts
  in deployment, 68
  in rule execution, 130
Send Event activity, BusinessWorks, 172
SendRequest(), HTTP and SOAP
  interactions and, 179
Sense-analyze-respond pattern
  event-enabled enterprises and, 9–10
  innovations in analysis, 8
  innovations in response, 9
  innovations in sensing, 6–8
  overview of, 5–6
Sensing. See also sense-analyze-respond pattern
Business Process Timeliness Monitor pattern, 44–45
capabilities of event-enabled enterprises, 31–32
comparisons against reference data, 144
Condition Detection pattern, 39–41
event-enabled enterprises and, 11
event pattern recognition. See event pattern recognition
  innovations in, 6–8
  missing or failing to recognize events, 15–16
  recognizing changes to cached objects, 148–149
  recognizing events, 12–14
  recognizing situation change, 143–144
  reference data change coordination, 147–148
reference data system of record, 145–147
Situation Recognition pattern, 41
Track-and-Trace pattern, 42–44
Separation of responsibility, in event-driven processes, 30
Sequences
  action performance, 157–160
  managing, 203–204
  preserving across multiple inference agents, 205
  preserving within one inference agent, 204
  recovering temporal sequencing (reordering), 205–206
Service Grid, ActiveMatrix, 204
Service level agreements (SLAs), 13–14
Services
  invoking Web Services, 130, 179–180
  TIBCO BusinessEvents as asynchronous service consumer, 175–178
  TIBCO BusinessEvents as service provider, 174–175
  TIBCO BusinessEvents as synchronous service consumer, 178–180
Shared-All
  cache object persistence, 111–113
  configuring backing store, 245–246
  fault tolerance and, 254
Shared-Nothing
  cache object persistence, 113
  configuring backing store, 245–246
  fault tolerance and, 254
Shared pool threads, choosing threading model, 230
Simple events, 14
Site disaster recovery. See also fault tolerance, 256–257
Situation Recognition pattern, 41
Situational Response pattern
  overview of, 45–46
  partitioning situation recognition from response, 38–39, 188–189
  Track-and-Trace pattern as special case of, 42–44
Sizing, rules of thumb for, 237
SLA Timeout Processing, 242
SLAs (service level agreements), 13–14
Snapshot queries
  example, 118–121
  executing, 115–117
  life cycle of, 117–118
  overview of, 115
SOAP
  IP redirectors for load distribution, 201
  sendRequest() for interacting with, 179
  using TIBCO Business Events as service provider, 174
Solutions
clusters and, 240
comparisons against reference data, 144
Continuous Query Change Recognition pattern, 151
decision tables, 155–156
duplicate event handling, 151–153
logging and exception reporting, 160
modularization patterns. See modularization patterns
naming guidelines, 160–161
overview of, 143
recognizing cached object change, 148–149
recognizing situation change, 143–144
reference data change coordination, 147–148
reference data system of record, 145–147
RMS (Rules Management Server), 156–157
rule templates, 154–155
runtime rule changes, 154
sequential and conditional action performance, 157–160
State Machine Change Recognition pattern, 149–150
summary, 161–162
Spreadsheets, Decision Manager interface and, 58
SQL
Inference Agent Publication pattern, 183
object persistence options, 111
State Machine Change Recognition pattern, 149–150
State Machine Maintains Asynchronous State, 177–178
State machines
change recognition pattern, 149–150
Claim Status example, 220, 223–224
event pattern recognition and, 164–165
maintaining asynchronous state, 177–178
modeling, 57–58
partitioning rules of thumb, 192
starting/stopping, 99
state transitions, 98–99
timeouts, 99
State models
managing state, 192
metadata models for process description, 23
modeling state machines, 57–58
overview of, 98
starting/stopping state machines, 99
state transitions, 98–99
timeouts, 99
transitions and, 98–99
Studio
BPMN (Business Process Modeling Notation) plugin, 62
data modeling and, 57
eclipse-based design environment, 56
solution life cycle and, 66
spreadsheet-style interface, 58
visualization of decision trees with, 228
Sub-processes, executing, 130
Subscription, Inference Agent Subscription pattern, 184–185
Synchronous I/O calls, used in rule actions, 231
Synchronous Service Consumer pattern, 178–180
System of record, for reference data
database as, 146
external system as, 146–147
TIBCO BusinessEvents, 145
T
Tabular data, dashboard agent and, 135
TCP, 230
Technical events, 14
Telecommunication service restoration case, 13–14
Templates, rule templates, 22, 40, 154–155
Temporal sequencing (reordering), recovery, 205–206
Testing, 262
then keyword, action clause of inference agents, 78–79
Threads
associated with channel types, 92
choosing threading model for inference agents, 229–231
postprocessing behavior, 93–94
selecting threading models by destination, 261
Thresholds
Condition Detection pattern and, 39–40
costant use in decision making, 22
transactions and, 106–109
TIBCO
ActiveMatrix. See ActiveMatrix
ActiveSpaces. See ActiveSpaces
Administrator, 56, 68
BusinessEvents. See BusinessEvents
Enterprise Message Service. See EMS (Enterprise Message Service)
Hawk. See Hawk
Rendezvous. See Rendezvous
TIBCO Architecture Fundamentals (Brown), 12–13, 171, xix–xx
TIBCO Runtime Agent (TRA), 56, 65–66
Timeliness, innovations in sensing and, 7–8
Timeouts
  avoiding deadlocks, 199
  Claim Status Timeout example, 233–234
  Claim Tracker example, 223
  notification events for, 235–236
  object locking and, 109
  state modeling and, 99
TimeToLive parameter, of inference agent
  event data structure, 71–72
TRA (TIBCO Runtime Agent), 56, 65–66
Track-and-Trace pattern
  Business Process Timeliness Monitor
    pattern as extension of, 44–45
  CEP design patterns, 42–44
  Claim Tracker example, 217–218
Transactional data, as context for analysis, 22
Transactions
  managing sequencing, 203–206
  thresholds, 106–109
Triggers
  for analysis, 261
  Condition Detection pattern and, 39
  demand analysis and, 232
  event-driven processes, 28
  performance events and, 233–234
  planning architecture and, 260
  Situation Detection pattern and, 41
  Track-and-Trace pattern and, 42–43
Tuning mechanisms. See also
  performance, 225
Two-second advantage (Ranadivé and Maney), 8
U
  UML 1.2 state machine notation, 57
  Updates, database, 182
V
  Validate Membership process,
    Nouveau Health Care case study, 212–213
Views
  be-views.exe, 135
  behavior of, 136–137
  overview of, 62–63
  TickerTracker example, 138–139
Virtual rules, 155–156
Visual alerts, display options for metric
  data, 137
W
  Wait for Event activity, BusinessWorks, 173
  Web Services, invoking, 130, 179–180
  when keyword, 78
  Work, timing of, 8
  Write-Behind behavior
    configuring backing store, 246
    object persistence and, 112
X
  XML event payloads, 228–229