

contents

preface xv
acknowledgments xvii
about this book xix

PART I HISTORY AND PRINCIPLES 1

1 SOA essentials 3

- 1.1 Brief history of distributed computing 4
 - Problems related to RPC-based solutions* 6
 - *Understanding SOAP's messaging styles* 6
 - *Advent of SOA* 7
- 1.2 The promise of web services for delivering SOA 9
- 1.3 Understanding the core characteristics of SOA 10
 - Service interface/contract* 10
 - *Service transparency* 11
 - Service loose coupling and statelessness* 13
 - *Service composition* 14
 - *Service registry and publication* 15
- 1.4 Technologies of a SOA platform 16
 - Business process management* 16
 - *Enterprise decision management* 17
 - *Enterprise service bus* 19
 - Event stream processor* 21
 - *Java Message Service* 22
 - Registry* 22
 - *Service components and compositions* 23
 - Web service mediation* 25

- 1.5 Introducing a SOA maturity model 25
- 1.6 Summary 27

2 *Defining the Open SOA Platform* 28

- 2.1 Evaluating open source products 30
- 2.2 Choosing a BPM solution 30
 - BPM product evaluation criteria* 31
 - *Open source BPM products* 32
 - *Selecting a BPM solution* 34
 - *Introducing JBoss jBPM* 34
- 2.3 Choosing an enterprise decision management solution 35
 - EDM product evaluation criteria* 37
 - *Open source EDM products* 37
 - *Selecting an EDM* 38
 - *Introducing JBoss Rules (Drools)* 39
- 2.4 Choosing an ESB 39
 - ESB product evaluation criteria* 40
 - *Open source ESB products* 42
 - *Selecting an ESB* 43
 - *Introducing Synapse as a lightweight ESB* 44
- 2.5 Choosing an ESP solution 45
 - What is event stream processing?* 46
 - *Introducing Esper* 47
- 2.6 Choosing a registry 47
 - Registry evaluation criteria* 49
 - *Open source registry products* 49
 - *Selecting a registry* 50
 - *Introducing WSO2 Registry* 51
- 2.7 Choosing a service components and composites framework 52
 - Examining the Service Component Architecture* 53
 - *Introducing Apache Tuscany* 54
- 2.8 Choosing a web services mediation solution 55
- 2.9 Summary 56

PART II ASSEMBLING COMPONENTS AND SERVICES 59

3 *Creating services using Apache Tuscany* 61

- 3.1 What are service components and compositions? 62
- 3.2 The SCA assembly model 64
 - Introducing the composite file* 66
 - *Configuring components* 70
 - Defining services* 74
 - *Working with properties* 76

Implementation options 79 ▪ *Using references for dependency injection* 84 ▪ *Defining available bindings* 87

3.3 Summary 93

4 *Advanced SCA* 94

4.1 Configuration using component types 95

4.2 SCA interaction models 96

Using conversations 96 ▪ *Understanding callbacks* 99

4.3 Scripting language support 104

Creating a Ruby component 105 ▪ *Creating a Java interface using the Ruby method signature* 105 ▪ *Modifying the service implementation class* 106 ▪ *Modifying the composition assembly* 106

4.4 Advanced Tuscany/SCA 108

Production deployment 108 ▪ *Introducing Service Data Objects (SDOs)* 113 ▪ *Advanced SDO features* 119

4.5 Summary 121

PART III BUSINESS PROCESS MANAGEMENT 123

5 *Introducing jBPM* 125

5.1 BPM: the “secret sauce” of SOA 127

5.2 History and overview of JBoss jBPM 129

Development lifecycle of a jBPM process 130 ▪ *Graph-oriented programming and jBPM* 136

5.3 Understanding nodes 137

Node nodetype 137 ▪ *Task-node nodetype* 139 ▪ *State nodetype* 139 ▪ *Mail-node nodetype* 140 ▪ *Decision nodetype* 142 ▪ *Fork and join nodetypes* 142

5.4 Using transitions 144

5.5 Extending using actions 145

Action class property instantiation 148 ▪ *Using action expressions* 149

5.6 Using events for capturing lifecycle changes in a process 151

5.7 Managing context using variables 153

5.8 Summary 155

6 *jBPM tasks* 157

- 6.1 What are tasks? 158
 - Task management using the jBPM Console* 159 ▪ *task element configuration* 160
- 6.2 Task user management 161
 - Actors and assignments* 162 ▪ *Understanding swimlanes* 164
- 6.3 Using timers 165
- 6.4 Task controllers 168
- 6.5 Developing with the task API 169
 - Identifying processes within a jBPM instance* 170 ▪ *Identifying running process instances for a given process* 172 ▪ *Finding open tasks within a process instance* 174 ▪ *Finding all tasks assigned to a user* 176 ▪ *Finding all pooled tasks for an actor* 176
 - Completing a task* 177
- 6.6 Summary 179

7 *Advanced jBPM capabilities* 180

- 7.1 Important enterprise features of jBPM 181
 - Superstates for grouping* 181 ▪ *Using subprocesses to manage complexity* 183 ▪ *Managing exceptions* 185 ▪ *Scripting with BeanShell* 187 ▪ *Audit logging* 190 ▪ *Understanding asynchronous continuations* 192
- 7.2 Integration with SCA/SDO 195
 - Using SCA client components for service integration* 196 ▪ *Service enabling jBPM* 201 ▪ *Developing the ListProcesses service operation* 203 ▪ *Developing the CreateProcessInstance service operation* 210
- 7.3 Summary 212

PART IV **EVENT STREAM PROCESSING, INTEGRATION, AND MEDIATION** 215

8 *Complex events using Esper* 217

- 8.1 Business events in the enterprise 218
- 8.2 Understanding events 219
 - BAM and ESP—what’s the difference?* 220 ▪ *Event-Driven Architecture and SOA* 221

- 8.3 What is Esper? 221
- 8.4 Getting started with Esper 224
 - What are event objects?* 224
 - *Defining and registering query statements* 225
 - *Specifying listeners or subscribers* 226
 - Configuration options* 226
- 8.5 EPL basics 227
 - Querying events* 227
 - *Using variables* 231
 - *Understanding views* 233
 - *Creating new event streams with named windows* 235
- 8.6 Advanced Esper 237
 - Extending with functions* 237
 - *Applying event patterns* 241
 - *Using JDBC for remote connectivity* 244
- 8.7 Service enabling Esper 245
 - Creating a framework and components* 246
 - *Esper service and session manager* 247
 - *SCA composite file* 248
 - Testing with soapUI* 250
- 8.8 Summary 250

9 *Enterprise integration and ESBs* 252

- 9.1 The relationship between ESB and SOA 253
- 9.2 Historical foundations of ESB 254
 - Core ESB capabilities* 256
 - *Appropriate uses of an ESB* 263
 - Inappropriate uses of an ESB* 265
- 9.3 Introducing Apache Synapse 268
 - Protocol adapters* 270
 - *Message-oriented middleware* 271
 - XML-based messaging* 272
 - *Intelligent routing and distribution* 272
 - *Message transformation* 272
 - Tasks/timers* 273
 - *Quality of service/web mediation* 273
 - Monitoring and administration* 273
 - *Extendable API* 273
- 9.4 Basic Apache Synapse message and service mediation 274
 - Simple message mediation example* 275
 - *Simple service mediation example* 279
- 9.5 Summary 282

10 *ESB implementation with Apache Synapse* 283

- 10.1 Learning Synapse through a case study 284
 - Phase I: typical web service mediation using error handling, routing, and transport switching* 284

- Phase 2: protocol/transport bridging and event propagation* 285 ▪ *Phase 3: using tasks, scripting, and database integration* 285 ▪ *Phase 4: quality of service mediation* 286
- 10.2 Phase 1: simple web service mediation 286
Sales order initiation 288 ▪ *Configuring the service mediation proxy and using validation mediation* 289 ▪ *Configuring XSLT mediation* 291 ▪ *Transport switching from HTTP to JMS* 292
Transport switching from JMS to HTTP 295
- 10.3 Phase 2: VFS, CSV, email, and message wiretap 299
Using the VFS transport 299 ▪ *Working with CSV files* 301
Exception handling and SMTP transport 303 ▪ *Using the wiretap message pattern* 304
- 10.4 Phase 3: tasks, DB mediator, and iterator 308
Configuring Synapse tasks 309 ▪ *Using the iterator mediator to split messages* 311 ▪ *Using the DB mediator* 312
- 10.5 Phase 4: QoS using Synapse 314
Implementing WS-Security 315 ▪ *Using Synapse throttling mediator* 317
- 10.6 Summary 321

PART V ENTERPRISE DECISION MANAGEMENT 323

11 *Business rules using JBoss Drools* 325

- 11.1 Understanding business rules 326
Benefits and drivers of the business rule approach 328
Relationship to SOA 329 ▪ *Characteristics of a rules engine* 330
Business rules management systems 332
- 11.2 Introducing Drools 333
Hello World, Drools! 334 ▪ *Running Hello World, Drools!* 338
- 11.3 Drools Rule Language (DRL) overview 339
- 11.4 Drools header elements 340
package 340 ▪ *import* 340 ▪ *expander* 341 ▪ *global* 341
function 341
- 11.5 Defining rules in Drools 342
Modifying rule behavior with attributes 342 ▪ *Conditional part of rule statement (when part)* 346 ▪ *Consequence part of rule statement (then part)* 354

- 11.6 Querying facts in Drools 356
- 11.7 Drools RuleFlow for rule orchestration 356
- 11.8 Alternatives to using Drools Rule Language 358
 - Using DSLs for business user authoring* 359
 - *Defining rules using decision tables* 362
- 11.9 Summary 363

12 *Implementing Drools* 364

- 12.1 Case study overview 365
 - Defining the DRL rules* 367
 - *Running as an embedded engine* 371
 - *User-friendly rules using a DSL* 377
- 12.2 Rules management using Drools Guvnor 379
 - Guvnor functionality overview* 379
 - *Rule authoring using Guvnor* 386
- 12.3 Developing decision services 390
 - What are decision services?* 390
 - *Designing the decision service* 392
 - *Implementing the decision service using Tuscany and Drools* 397
 - *Testing* 403
- 12.4 Summary 404
 - resources* 406
 - index* 409