

# Event processing in action – Table of contents

## Preface

## Part I – Introducing the book and the event programming principles

### 1. Introduction

#### 1.1 Event-driven behavior in daily life

1.1.1 The notion of event

1.1.2 Some examples of event-driven computing

1.1.3 What are the reasons using event-driven computing systems?

#### 1.2. The main concepts of event-driven computing

1.2.1 Event Driven Architecture

1.2.2 Events and event distribution

1.2.3 Event processing networks

1.2.4 Types of intermediary event processing

1.2.5 Streams and stateful event processing

1.2.6 Event processing and its relationship to the real world

#### 1.3 The business value of event processing software

1.3.1 Effectiveness issues

1.3.2 Efficiency issues

#### 1.4 Fast Flower Delivery: an example that accompanies this book

1.4.1 General description

1.4.2 Skeleton specification

#### 1.5 How can you utilize the book's website

1.5.1 The languages-based part of the website

1.5.2 The topic-based part of the website

#### 1.6. Summary

## **2. Event Programming Principles**

- 2.1. The background: request-response interactions
- 2.2. Event-driven interactions and the principle of decoupling
- 2.3. Further event distribution patterns
- 2.4. Benefits of using the event-driven approach
- 2.5. Event processing and its connection to related concepts
  - 2.5.1 Event-driven Architecture and Service Oriented Architecture
  - 2.5.2 Event-driven Business Process Management
  - 2.5.3 Business Activity Monitoring (BAM)
  - 2.5.4 Business Intelligence (BI)
  - 2.5.5 Business Rule Management Systems (BRMS)
  - 2.5.6 Network and Systems Management
  - 2.5.7 Message Oriented Middleware (MOM)
  - 2.5.8 Stream Computing
- 2.6. The event processing building blocks
  - 2.6.1 What is a building block?
  - 2.5.2 What information is contained in a building block?
  - 2.6.3 The event processing agent building blocks
- 2.7 Event processing networks
- 2.8 Summary

## **Part II - Deep dive into event processing – step by step**

### **3. Defining the Events**

- 3.1. Event type
  - 3.1.1 The logical structure of an event
  - 3.1.2 The event type definition element
- 3.2. Header attributes
  - 3.2.1 Event type description attributes
  - 3.2.2 Event instance attributes
- 3.3. Payload attributes
  - 3.3.1 Data types
  - 3.3.2 Attributes with semantic roles
- 3.4. Event to event relations
- 3.5. Event types in the Fast Flower Delivery example
- 3.6. Event representation in practice

### 3.7. Summary

## 4. Producing the events

### 4.1 Event producer: concept and definition element

#### 4.1.1 The event producer definition element

#### 4.1.2 Event producer details

#### 4.1.3 Output terminal details

#### 4.1.4 Producer relations

### 4.2 The various different kinds of event producer

#### 4.2.1 Hardware event producers

#### 4.2.2. Software event producers

#### 4.2.3. Human interaction

### 4.3. Interfacing with an event producer

#### 4.3.1 Interaction patterns

#### 4.3.2 Queriable event producers

#### 4.3.3 Interfacing mechanisms

### 4.4 Producers in the Fast Flower Delivery example

### 4.5 Summary

## 5. Consuming the events

### 5.1. Event consumer: concept and definition element

#### 5.1.1. Event consumer definition elements

#### 5.1.2. Event consumer details

#### 5.1.3. Input terminal details

#### 5.1.4. Consumer relationships

### 5.2. The various different kinds of event consumer

#### 5.2.1 Hardware event consumers

#### 5.2.2 Human interaction

#### 5.2.3 Software event consumers

### 5.3 Interfacing with event consumers

#### 5.3.1 Interaction patterns

#### 5.3.2 Interfacing mechanisms

### 5.4 Consumers in the Fast Flower Delivery example

### 5.5 Summary

## **6 The event processing network**

### 6.1 event processing networks

6.1.1. The event processing network and its notation

6.1.2. Recursive event processing networks

6.1.3 Implementation perspective

6.1.4. Benefits of an explicit EPN representation

### 6.2. Event Processing Agents

6.2.1 The functions of an EPA

6.2.2 EPA types

6.2.3 The filter EPA

6.2.4. The transform EPA

6.2.5 The pattern detection EPA

6.2.6 The EPA definition element

6.2.7 Event processing agents in the Fast Flower Delivery example

### 6.3 Event Channels

6.3.1 The event channel notion

6.3.2 Routing schemes

6.3.3 Channels in the Fast Flower Delivery example

### 6.4 Global State

6.5 EPN in practice

6.5 Summary

## **7 Putting events in contexts**

7.1 The notion of context and its definition element

7.2 Temporal context

7.2.1 Fixed interval

7.2.2. Event interval;

7.2.3. Sliding fixed interval

7.2.4. Sliding event interval

7.3. Spatial context

7.3.1. Fixed location

7.3.2. Entity distance location

7.3.3. Event distance location

7.4. State oriented context

7.5. Segmentation oriented context

7.6. Event synonyms policies

7.7. Composite contexts

7.8. Contexts in the fast flower delivery

7.9. Context definition in practice

7.10. Summary

## **8. Filtering and transforming**

### 8.1. Filtering in the event processing network

- 8.1.1 Filtering on an input terminal
- 8.1.2 Filtering in an event processing agent
- 8.1.3 Filtering and event processing contexts

### 8.2. Transformation in depth

- 8.2.1. Project, translate and enrich
- 8.2.2 Split
- 8.2.3 Aggregate
- 8.2.4 Compose
- 8.2.5 Header attributes and validation

### **8.3 Examples in the Fast Flower Delivery application**

### **8.4 Filtering and transformation in practice**

### **8.5 Summary**

## **9. Detecting event patterns**

### 9.1. Introduction to event patterns

- 9.1.1. The pattern matching process
- 9.1.2. Threshold patterns
- 9.1.3. Event patterns categories and types

### 9.2. Basic event patterns

- 9.2.1. Logical operators patterns.
- 9.2.2. Threshold patterns
- 9.2.3. Relative patterns
- 9.2.4. Modal patterns

### 9.3. Dimensional patterns

- 9.3.1. The sequence pattern
- 9.3.2 Trends pattern
- 9.3.3 Spatio-temporal patterns

### 9.4. Pattern policies

- 9.4.1. Repeated type policies
- 9.4.2. Order policies
- 9.4.3. Cardinality policies
- 9.4.4. Consumption policies

### 9.5. Pattern reference table

### 9.6. The fast flower delivery patterns

### 9.7. Pattern detection in practice

## 9.8. Summary

# **Part III – Additional Topics**

## **10. Engineering and implementation considerations**

### 10.1. Event processing programming in practice

#### 10.1.1 Stream oriented programming style

#### 10.1.2 Rule oriented languages

#### 10.1.3 Development environments

### 10.2. Non-functional properties

#### 10.2.1 Scalability

#### 10.2.2 Availability

#### 10.2.3 Security

### 10.3. Performance objectives

### 10.4. Optimization types

#### 10.4.1 EPA assignment optimizations

#### 10.4.2 EPA code optimizations

#### 10.4.3 Execution optimizations

### 10.5. Summary

## **11. Focal point on major challenging topics**

### **11.1. The temporal semantics of event processing**

#### 11.1.1 Occurrence time: time point vs. interval

#### 11.1.2 Putting derived events in order

#### 11.1.3 Event order and out-of-order semantics

### 11.2 inexact processing

#### 11.2.1 Uncertain events and inexact event content

#### 11.2.2 Inexact matching between events and situations

#### 11.2.3 Handling inexact event processing

### 11.3. Retraction and causality

#### 11.3.1 Event retraction

#### 11.3.2 Event causality

### 11.4. Summary

## **12. Emerging directions of event processing**

### **12.1 Event processing trends**

- 12.1.1 Going from narrow to wide
- 12.1.2 Going from monolithic to diversified
- 12.1.3 Going from proprietary to standard-based
- 12.1.4 Going from programming-centered to semi-technical-centered
- 12.1.5 Going from stand-alone to embedded
- 12.1.6 Going from reactive to proactive

### **12.2 Future directions in event processing technology**

- 12.2.1 Event processing virtual platforms
- 12.2.2 Event processing optimization
- 12.2.3 Event processing software engineering
- 12.2.4 Intelligent event processing

Appendix A: A dictionary of definitions.

Appendix B: The complete specification of the "Fast Flower Delivery" use case

Appendix C: Short Survey of event processing products and link to executable implementations of the use case.