Event Driven Architecture and Complex Event Processing

Paul Fremantle
CTO and co-Founder
WSO2 Inc
Overall Presentation Goals

Understand Event Driven Architecture
Look at how you can implement Event based models
Quick overview of Apache Synapse
How CEP extends EDA
Examine Open Source CEP and EDA approaches
Look at some code and XML
Speaker’s qualifications

- Paul Fremantle, Chief Technology Officer and Co-Founder, WSO2
- [http://wso2.com](http://wso2.com) Open Source SOA Middleware
- OASIS WSRX TC Chair
- Apache Synapse Chair
- InfoWorld CTO 25 2008
- [http://pzf.fremantle.org](http://pzf.fremantle.org)
Event Architecture

What is an Event?
In everyday usage
Event Organizers
What should be Google’s #1 Event organizer!
In Philosophy

Object(s) [x], a property [P] and time or a temporal interval [t]

Jaegwon Kim
In Event Architecture

- A set of information (properties) about an object (or objects)
- At a given time
- Usually encapsulated as a message
Event based models are everywhere

nntp://

twitter

irc://

SPAM
# of mailing lists > # of employees
Sensors and Actuators
Sensors and Actuators
Publish Subscribe

• A distribution model for Events
• Events are tied to some logical model
  • “Event Streams” /
  • “Topic” is the most used model
• Topics are a tree-based model or namespace that makes it easy to organize your Events
An example

Sensor → Event → Actuator
An example

Sensor

Event

Actuator

Actuator
The overall integration/data model is organized into a *Topic Space*. Wiring is devolved: each system subscribes to the topics it can understand.
Why is it attractive?

• Incredibly loosely coupled
  • The sensor doesn’t need to be aware of the actuator
  • The actuator doesn’t need to be aware of the sensor

• Events can be re-distributed
  • The topology can change

• New actuators or sensors can be added seamlessly

• Topic or Event Streams give the ability to self-organize
Scaling to new participants

System A

ESB

System B

Registry

A

B
Scaling to new participants

System A

ESB

System B

Registry
A
B
C

System C
Scaling to new Participants

- In a “point-to-point” model
  - You need to wire in the new participants

- In EDA it becomes the responsibility of the user to subscribe and publish to the right topics

- Analogy:
  - The difference between
    - asking you to include me in the “To:” list
    - Subscribing to a mailing list
Self Contained Messages

- A self-contained message is one where the data is fully realised
- Can be archived and reviewed in the future
- No references to other data that may no longer be available or may be out of sync
- I’m ambivalent about this
- One option is permanent URLs / versions

Problem - echo / feedback loop

How do you stop feedback?

- Timing?
  - Doesn’t work!
- Looking at the data
  - Requires an understanding of the schema and “identity”
Solving the feedback by adding Master Data
Wire transports for Events

- (JMS)
- Stomp/.../...
- AMQP
- XMPP
- WS-Eventing / SOAP
- WS-Notification / BrokeredNotification / Topics
- Atom / RSS
WS-Eventing

- Publish-Subscribe in Web services/XML
- A very simple model for Event subscription over SOAP
- Three main concepts:
  - An Event Source
  - A Subscriber endpoint
  - A Filter
- Create, Delete, Renew and expire subscriptions
- Subscriptions may have XPath-based filters
  - Specify how event messages should be delivered
WS-Eventing

Subscriber

Subscribe with URL of subscriber

Respond with URL of SubsMgr

Event Notifications (any message)

getStatus

renew

unsubscribe

Event Source

SubsMgr
Synapse

- A lightweight ESB from Apache
- Highly performant
  - Based on a stream-based parsing model
    - Apache AXIOM
  - And highly tuned Non-Blocking HTTP
    - Apache HTTPCore
- Simple XML-based configuration
- UI-based config and administration from WSO2 ESB
- [http://synapse.apache.org](http://synapse.apache.org)
Flows in Synapse

Proxy

inflow sequence
- log
- xslt
- send

outflow sequence
- send
- xslt
- log

Endpoint

HTTP 8080
HTTPS 8443
JMS
SMTP
How we are adding Eventing into Synapse

- Core Event library
  - “EventSource” - roughly equivalent to a Topic
  - SubscriptionManager
    - Memory, Database, Registry - based
- Event Publisher
  - Mediator that emits events to all subscribers
WSO2 Commons Eventing
Apache License library

- Event
- EventFilter
- EventSink
- EventSource
- EventingConstants
- FilteredSink
- NotificationManager
- Subscription
- SubscriptionData
- SubscriptionManager
Adding an Event Source to Synapse

<eventSource name="Paul">
  <subscriptionManager
      class="org.apache.synapse.eventing.managers.DefaultInMemorySubscriptionManager">
    <property name="topicHeaderName" value="Topic"/>
    <property name="topicHeaderNS" value="http://apache.org/aip"/>
  </subscriptionManager>
</eventSource>
Publishing messages

<proxy .....>

...<sequence>
  <eventPublisher eventSourceName="eventsourc1"/> </sequence>
</proxy>
Simple Event Processing

- XPath-based filtering
  - (only send on certain messages)
- XSLT / XQuery / E4X transformations
- Also
  - CSV/EDI/COBOL
  - Smooks
How we are using this

- Adapters publish “native” events
- The ESB acts as a smart event broker
  - Transforms native events into standard schema events
  - Everyone who needs to can subscribe
What is Complex Event Processing?

- **Simple Event Processing**
  - Acting on single events
  - e.g. a `<filter>` in the ESB
  - Is this a gold or platinum customer?
- **Event Stream Processing**
  - Looking across multiple events
  - Finding patterns - e.g. the CPU utilization has been more than 90% for the last 10 minutes
- **Complex Event Processing**
  - Looking across multiple event streams
  - e.g. There has been a significant increase in overall trading activity AND the average price of commodities has fallen 2% in the last 4 hours
Example uses of Complex Event Processing

- Trading systems
- Fraud Detection
- RFID processing
- Systems Management
- Spying on you
Little Brother

http://craphound.com/littlebrother/download/
Esper is a component for CEP and ESP applications, available for Java as Esper, and for .NET as NEsper. Esper and NEsper enable rapid development of applications that process large volumes of incoming messages or events. Esper and NEsper filter and analyze events in various ways, and respond to conditions of interest in real-time.

Technology Introduction

Complex Event Processing, or CEP, is technology to process events and discover complex patterns among multiple streams of event data. CEP stands for Event Stream Processing and deals with the task of processing multiple streams of event data with the goal of identifying the meaningful events within those streams, and deriving meaningful information from them. Real-time OLAP (online analytical processing) is also a term used frequently for
Esper

- [http://esper.codehaus.org](http://esper.codehaus.org)
- A Java library that can be integrated into multiple systems
- A GPLv2 project
  - (with a commercial edition available)
- Supports multiple query models
  - Based on a SQL-like language
  - Grouping, aggregation, sorting, filtering and merging of event streams
  - Windows based on time, length, sorted, and others
- Events can be XML, Map, Object
Esper in Java

- Events can be simple POJOs, Maps or XMLs
- e.g.

```java
package org.myapp.event;

public class OrderEvent {
    private String itemName;
    private double price;

    public OrderEvent(String itemName, double price) {
        this.itemName = itemName;
        this.price = price;
    }

    public String getItemName() {
        return itemName;
    }

    public double getPrice() {
        return price;
    }
}
```
A statement

```java
EPServiceProvider epService = EPServiceProviderManager.getDefaultProvider();
String expression = "select avg(price) from org.myapp.event.OrderEvent.win:time(30 sec)";
EPStatement statement = epService.getEPAdministrator().createEPL(expression);
```
## What can you do with Esper statements

<table>
<thead>
<tr>
<th>Activity</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure rate of arrival</td>
<td><code>Select count(*) from Event.win:time_batch(1 second)</code></td>
</tr>
<tr>
<td>Correlate events from 2 or more streams</td>
<td><code>select fraud.accountNumber as acctNum, withdraw.amount as amount from FraudEvent.win:time(30 sec) as fraud, WithdrawEvent.win:time(30 sec) as withdraw where fraud.accountNumber = withdraw.accountNumber</code></td>
</tr>
<tr>
<td>Create new events</td>
<td><code>insert into WarningStream select trainStation, avg(waitTime) as avgWait from MyWindow group by trainStation having avg(waitTime) &gt; 60</code></td>
</tr>
<tr>
<td>React to events</td>
<td><code>on WarningStream delete from MyWindow</code></td>
</tr>
</tbody>
</table>
What happens when an event fires?

```java
public class MyListener implements UpdateListener {
    public void update(EventBean[] newEvents, EventBean[] oldEvents) {
        EventBean event = newEvents[0];
        System.out.println("avg=" + event.get("avg(price)"));
    }
}

MyListener listener = new MyListener();
statement.addListener(listener);
```
Injecting Events into Esper

OrderEvent event = new OrderEvent("shirt", 74.50);
epService.getEPRuntime().sendEvent(event);
Sci-flex

- A plug-in that makes Esper processing available within Synapse/WSO2 ESB
- Allows you to
  - Define Events to Esper
  - Define Esper EPL statements and specify where events are routed
  - Inject events from sequences
  - Supports XML using Axiom
    - Provides significant performance over DOM
    - Adding support for Maps as we speak
Why use Esper with Synapse?

- Synapse provides
  - A unified transport model
    - HTTP, JMS, XMPP, Atom, TCP, Email, etc
  - Transformation
    - Can manipulate events as they are emitted from Esper and reformat to match a third-party system
- Simple wiring
- Some examples:
  - Proxy SOAP/HTTP events and also send interesting events to a third system
  - Take JMS ticker events and publish a 10 min batch summary as an Atom feed
Example:
Looking for system monitoring events

http://ganglia.info
Getting the Axiom mediator working

• Add the following to the LIB path:
  • esper-2.1.0.jar [Main Esper]
  • esperio-2.1.0.jar [Contains AxiomPlugin]
  • antlr-runtime-3.0.1.jar [Used by Esper]
  • cglib-nodep-2.1_3.jar [Used by Esper]
  • synapse-esper-plugin-mediators-0.9-alpha.jar [Mediator code]
A simple example

<filter source="get-property('To')" regex="urn:tick">
  <class name="org.sciflex.plugins.synapse.esper.mediators.AxiomMediator">
    <property name="Configuration">
      <esper-configuration>
        ...
      </esper-configuration>
    </property>
    <property name="statement" value="select * from Ticker.win:length_batch(10)"/>
    <property name="EventToAddress" value="urn:newevent"/>
  </class>
</filter>
<filter source="get-property('To')" regex="urn:newevent">
  <log level="full"/>
</filter>
FIX example

FIX Endpoint (Order generator)

Fix Proxy
<class AxiomEsperMediator.>
<send/>

WSO2 ESB

Esper

New Event

FIX Endpoint (Order acceptor)

JMS Topic
Ongoing work

- Adding EPL support to the Eventing model
  - Allows dynamic add of new CEP queries into a running system
- For example
  - A trader is looking for a particular pattern
  - Use a GUI tool to create the EPL
  - Send a subscribe message
  - Now be notified of new events
Status

• Sci-flex Esper Mediator is released
• Eventing support will be in:
  • Synapse 1.3 and
  • WSO2 ESB 2.0 (January time)
  • Available in nightly builds today
• Esper currently 2.3.0
  • 3.0 planning in progress
Summary

• Event Driven Architecture is a good thing

• Adding Complex Event Processing can significantly add value

• Some open source to help you:
  • Esper
  • Synapse
  • Sci-Flex
Thanks for your attention!

http://synapse.apache.org
http://esper.codehaus.org
http://code.google.com/p/sci-flex/
http://wso2.org/projects/esb
http://soa-eda.blogspot.com
http://code.google.com/p/sci-flex/