Interchange of ECA Rules with Xpath expressions

Yuri Boglaev
Ameritrade, yboglaev@ameritrade.com

Introduction

Event-condition-action (ECA) is the basic pattern for business rules implementation in the vast majority of companies dealing with complex set of either industry regulations or internal business processes and policies. For example, financial institutions have common federal and/or state rules of conducting business as well as specific rules, e.g. inside brokerage or insurance industry. The standards for publishing and interchanging of ECA rules are needed to close the gaps between various technologies used in this area and finally improve bottom line. The topic is especially important during the merger/acquisitions. This position paper is based in part on the readers feedback to the article http://www.ftponline.com/javapro/2003_08/online/yboglaev_08_08_03/

1. ECA rule specification components

- XML spec. of the event data sources
- Spec. of condition-action relations (logic imposed on data and actions)
- XML spec. of action interfaces

There are no extra efforts to the XML data specification. It is just XML image of data sources that under ECA monitoring that could trigger event notifications to the event listeners. It should follow the schema or DTD from the subject domain in question.

There is no unified XML spec for action interfaces, although something similar to WSDL could be the 1st iteration on this way. The next iteration should include at least XML image for IDL (interface definition language) from OMG.

There is no standard approach right now how to specify condition-action relationship. The main problem is that the rule engine vendors are seeing this part through their proprietary implementations.

One of the possible way to use common denominator for such spec would be an acceptance of the XPath expressions as a way to define the ECA rules. XPath expressions are passed to the conditions and actions as operands in rule definitions.

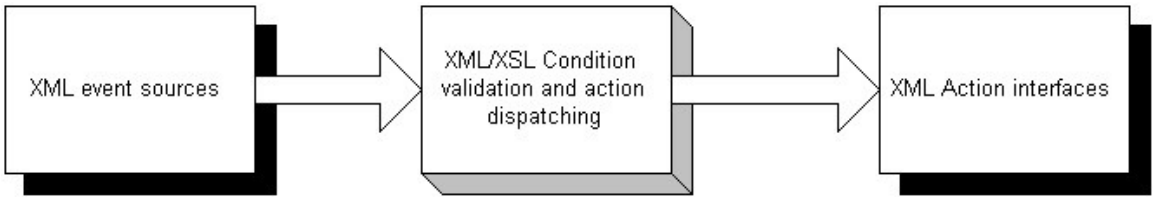
The one of the benchmark tests for the language standard could be the simplicity/complexity/feasibility of the code generation for an implementation language of choice from ECA rule spec. For example, the IDL based spec would be a good

candidate for simplicity mark in such test, but not WSDL.

2. What ECA rule specification should not contain

- Implementation language specific structures (class instances, object fields etc.)
- Actions references to classes or objects (class, object methods)
- Inference type (condition validation strategy)

It is worth to emphasize that if we reduce the level of a rule engine exposure in ECA spec to the event listener and dispatcher of execution, then all proprietary language specifics will be eliminated.



3. Example

The following is the simple example showing usage of XPath expressions with references to the event source and interface documents.

```
<rule_set>

<!-- variables used in xpath expressions -->
<variables>
  <var name="customer_state">/shipment/customer/address/state</var>
  <var name="warehouse_state">/shipment/warehouse/address/state</var>
  <var name="items_number">/shipment/items/item[@quantity]</var>
</variables>

<!-- tax rule -->
<rule name="TAX">
<if>
      <condition>$customer_state != $warehouse_state</condition>
      <action>interface:message("Out of state tax is applied")</action>
</if>
<else>
      <action>interface:message("In state tax is applied")</action>
</else>
</rule>

</rule_set>
```

Conclusion

Standard for ECA rule specification could be created today by fusing several very well known and accepted technologies at hand: XML, XSL/XPath and modified WSDL/IDL.