Pushing Reactive Services to XML Repositories using Active Rules

Angela Bonifati, Stefano Ceri, Stefano Paraboschi
Politecnico di Milano (Italy)

Speaker: Angela Bonifati
bonifati@elet.polimi.it
Outline

- Problem Definition
  - The need of prompt e-services over the Internet
- Solution
  - Distributed reactive services enabled by XML technology
  - Rules are installed on remote systems and promptly monitor the changes
- Architectural framework
- Implementation
- Conclusions
Problem Definition

- Push technology is a convenient solution to deliver updated information to end-users
- So far “Pushing logic” has been broadly applied in IS but only with local mechanisms
- Internet services offer an infrastructure for the implementation of many different architectures
- XML and XML-based protocols are a central component of Internet services
- XML active rules permit the construction of distributed push technology
Our solution in a nutshell

- ECA paradigm on XML documents to bear notification services
- Rules located close to the data (remote installation)
- A simple subscription protocol for rule negotiation and a suitable application scenario
- The broadening XML technologies to cook up our framework

- XML, XQuery, DOM Level 2, SOAP
The renewed ECA Paradigm

Active rules for Notification Services follow the ECA paradigm:

- Events: DOM mutating events generated whenever the XML docs are modified;
- Conditions: queries on the document base, using a suitable XML query language (e.g.: XQuery);
- Actions: invocations of SOAP methods implementing calls to a message delivery system
Active Rules at work

- The event specification is expressed by the DOM Level 2 API and generates the data bindings to be passed to the condition-action part.

- In the Condition, predefined variables `new` and `old` are declared in a similar way to transition variables in databases.

- In the Action, SOAP methods implement the transfer of information to recipients.

- The Action is “simplicistic” and receives parameters from conditions; to prevent problems related to rule termination, updates of XML data are avoided.
A simple rule

<event>insert(//cd)</event>
<condition> FOR $a IN //cd
    WHERE $a=$new AND
    $a/price < 20 AND
    contains($a/author,"Milli Vanilli")
    RETURN $a
</condition>
<action>
    <soap-header>
        <uri>/notification</uri>
        <host>131.175.16.105</host>
        <soap-action>notify</soap-action>
    </soap-header>
    <SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/encoding">
        <SOAP-ENV:Body>
            <m:Notify xmlns:m="http://131.175.16.105/methods">
                <cdfound>
                    $a//*
                </cdfound>
            </m:Notify>
        </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
</action>
Architectural framework

Rule Subscription Protocol

Service Broker
Rule Broker

Service Suppliers
XML Repository
Rule Engine

Rule Broker

Service Reseller
XML Repository
Rule Engine

Rule Repository

Web & Mail Interfaces

Requests

Messages

Clients
Actors

● Service Reseller
  ■ Focus: interaction with the final user
  ■ Recipient of messages produced by rule actions

● Service Broker
  ■ Focus: Construction of rules satisfying the user needs
  ■ Intermediary between resellers and suppliers

● Service Supplier
  ■ Focus: Management of XML information
  ■ Internet site equipped with a rule execution engine
An example: the real estate agency (1)

- Consider a real estate agency application:
  - **example of request**: a furnished four-bedroom (or more), two-bathrooms (or more) Victorian house, which costs $1,500,000 or less, located in the Marina area in San Francisco.

- No matching is found on the house agency Web sites!
An example:
the real estate agency (2)

- A reactive service is invoked:
  - A request is sent from the Service Reseller to the Service Broker
  - A set of rules are agreed by defining their contract and by setting the proper authorization through the Rule Subscription Protocol
  - The rules are installed into several house agency XML Servers (Service Suppliers)
  - When the targeted house appears on the market, the rules produce the response messages, which are delivered from the XML Server to the Service Reseller
<event>
  update($a) OR insert($a)
</event>

<condition>
  FOR $a IN document( )//housestobuy/house,
  WHERE $a//cost < 1500000 AND
      contains($a//style,"victorian") AND
      contains($a//description,"furnished") AND
      $a//nr_of_bedrooms >= 4 AND
      $a//nr_of_bathrooms >= 2 AND
      $a//city="San Francisco" AND
      $a//area="Marina" AND
      empty($a//sold_to)
  RETURN $a
</condition>

<action>
  <SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
    <SOAP-ENV:Body>
      <m:DeliverHouseNews xmlns:m="http://housemediator.com/soap/methods">
        <foundthehouse>
          $a//*
        </foundthehouse>
        <server>
          www.expensivehousesinCalifornia.com
        </server>
        <localHouseld>
          $a/@ld
        </localHouseld>
      </m:DeliverHouseNews>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
</action>
The B2B interface

- The Rule Broker and the Service Supplier communicate through a B2B protocol, which is both technical and business-oriented.
- This protocol is based upon four SOAP primitives, that are invoked by the Service Broker and supported by the Service Supplier.
- The Service Broker is required to know well how to write rules and how to submit them to the Service Supplier.
- The Service Reseller can be the final user of the notification server or a mediator which interacts with the final user through a friendly interface.
The SOAP primitives (1)

- **Connect:**
  - Instead of using a stateless connection a-la HTTP, this SOAP primitive warrants a connection to manage more than a single request-response:

    ```
    ConnectionId Connect (in AuthenticatedUser user, 
                         in ServerProfile requestedProfile)
    ```

- **Subscribe:**
  - This primitive permits the submission of a rule to a server with a specified contract:

    ```
    SubmissionId Subscribe (in ConnectionId openCon, 
                           in Rule ruleToSub, in Contract conProp)
    ```
The SOAP primitives (2)

- **Unsubscribe:**
  - This primitive is invoked when a submitted rule must be removed from the Service Supplier:

    ```
    void Unsubscribe (in ConnectionId openCon, in SubmissionId SubId)
    ```

- **Disconnect:**
  - This primitive closes the connection created by the Connect primitive and frees the resources that were allocated for the connection:

    ```
    void Disconnect (in ConnectionId openCon)
    ```
Rule Packaging

● The Subscribe primitive presents a *ProposedContract* parameter, which specifies the contract to be agreed by each part in the transaction.

● The contract should include the remuneration information and the guarantees of each rule.

● The contract is the “sine qua non” of a B2B initiative.

● In our solution, the contract is application-dependent and has to be defined rule per rule.

From the real estate ex.:

```xml
<contractProposed>
  <cost>0</cost>
  <guarantee>none</guarantee>
</contractProposed>
```
Implementation

- Needed components: an XML system supporting the DOM Event Model, an XQuery engine, a SOAP implementation:
  - The DOM permits the definition of event listeners, that are attached to each node to be monitored
  - When the event is captured, the XQuery processor evaluates the XQuery condition, possibly returning an XML fragment
  - The SOAP call is built from the query result and executed by the server

- Much simpler rule engine than in database trigger systems
DOM Event detection

- Two main strategies:
  - *centralized*: a single event listener associated with the root of the document
  - *fragmented*: a set of event listeners, associated with every node instance on which events need to be monitored

- Impact on system performance
Conclusions

- Active rules for pushing reactive services has a great potential
- The reuse of current Web standards makes the implementation relatively easy
- Some obstacles remain:
  - pre-knowledge of the schema of XML resources
  - risks of external rule execution
  - scalability issues