JavaOne℠

Real-World Processes with WS-BPEL

Murali Pottlapelli
Ron Ten-Hove
Sun Microsystems, Inc
Application Platform Software Engineering

Sun microsystems

WS-BPEL Is Incomplete

Real-world applications need more capabilities than WS-BPEL 2.0 delivers

About Murali Pottlapelli

- Senior Staff Engineer in SOA/BI product group
- 5 years implementing and refining BPEL engines, started with BPEL 1.0 in 2004
- 15 years in delivering software products and solutions

About Ron Ten-Hove

- Senior Staff Engineer in IdM product group
- Many years in SOA + integration product engineering
  - Workflow engine product engineering
  - JSR 208 Co-spec Lead
  - Member of the OASIS WS-BPEL technical committee
- Murky past in industrial automation, oil & gas
- http://blogs.sun.com/rtenhove/

Not the Agenda

- Variety of real-world problems
  - Hard to do in plain WS-BPEL 2.0
- One or more solutions
  - Extension of BPEL
  - BPEL engine implementation feature
  - Under-utilized BPEL language feature (that is actually useful)
- Not a BPEL tutorial
- Extensions & implementation for BPELSE

BPEL feature - Correlation

- Declaration is very tedious
  - Multiple artifacts to define/edit
    - Property
    - PropertyAlias
    - CorrelationSet
    - Associate with messaging activity
  - Multiple files to edit

BPEL feature - Correlation
Solution

Demo of Correlation Wizard

Implementation feature – Reliability
Problem

- Business processes can span long time periods
  - Insurance claims take days and weeks!
- Maintenance downtime
- Hardware or software failure leading to downtime
- Crashes should:
  - Be short
  - Not lose process instances or data

Implementation feature – Reliability
Solution

- Business process instance execution state is persisted
  - Business process instance is a state machine – state changes on each activity
  - Some activities are idempotent – no need to persist
  - Some activities are not – persist state
- Recover instance state from the persisted state on restart
- Continue execution from the last persisted state

Implementation feature – High Availability

- Business needs processes to be highly available
  - GlassFish ESB supports GlassFish cluster
  - All the nodes in the cluster are homogeneous
  - Available nodes continue to service the requests
  - Business process instances on the failed nodes migrate to available nodes
    - Leasing algorithm is used for detecting failed nodes
    - Instances owned by failed nodes are acquired by available nodes.
  - BPEL-SE needs to be configured with high availability database

Implementation feature – Scalability
Problem

- Business processes instance life spans a few hours to several days
  - Most of the time waiting for an event or timeout
- Business process instance state is consuming memory even when there is no activity
- It limits concurrent instances executed for a given memory
  - BPEL variable foot print – message size - 10KB to 10 MB
  - Business process instance itself consumes memory

Implementation feature – Scalability
Solution

- Dehydrate business process instance state when waiting on an event
- Load business process instance state into memory when waiting event arrives
- Dehydrate and hydrate is expensive – Only do when needed and configurable
  - Threshold 1 – available to total memory falls below this ratio – Dehydrate variables for waiting instances
  - Threshold 2 – available to total memory falls below this ratio – Dehydrate waiting instances

Implementation feature – Retry
Problem

- Services aren't always available, and <invoke> doesn't support retries.
- BPEL logic to do retries is UGLY, and hides the real business logic

Implementation feature – Retry
Solution

- Retry on failed <Invoke> message delivery
  - Configurable number of retries
  - Configurable delay between retries
- Outcome is success - continue with execution
- Outcome is error - Configurable actions
  - Route message to dead letter queue
  - Suspend BP instance
  - Propagate error to process – either terminate or handle it in Fault Handler

<Demo>

Implementation feature – Throttling
Problem

- Services have limit on the concurrent consumers
  - Due to resource constraint
  - SLAs
  - Bugs or malice
- BPEL has no mechanism to specify limit on concurrent invocations for <invoke>
- No way to limit <receive> *et al*.

Implementation feature – Throttling
Solution

- Configure concurrent invocation limit on Partner Link
  - Queue requests over limit
  - As requests complete, queue is drained FIFO
- Multiple invokes sharing Partner Link share limit

<Demo>

BPEL Extension – Protocol Headers
Problem

- Ideally business process should
  - Work with abstract WSDLs,
  - Independent of communication protocols
- Often partners/vendors provide business data in headers
- Real-world is different from theory, to interact with systems in the field, need access to protocol-specific data

BPEL Extension – Protocol Headers
Solution

- Provided access to protocol specific data
  - Both get & set
- Access to headers by extension on <from> and <to>

```
    <copy>
        <from variable="inputVar" sxnmp:nmProperty="org.glassfish.openesb.inbound.address.url"/>
        <to>$outputVar.resultType/ns2:paramA</to>
    </copy>
</assign>
```

- Access to header by extensions on property-alias

```
<vprop:property name="CONTENTTYPE" type="xsd:string"/>
<vprop:propertyAlias messageType="ns:accessCONTENTTYPEOperationRequest" part="part1"
    propertyName="tns:CONTENTTYPE"
    sxnmp:nmProperty="org.glassfish.openesb.inbound.http.headers">
    <vprop:query>content-type</vprop:query>
</vprop:propertyAlias>
```

BPEL Extension – Protocol Headers
Solution

- Demo Netbeans BPEL mapper
  - soap headers
  - http headers
  - jms properties
  - file properties

<Demo>

BPEL Extension – Attachments
Problem

- Documents needs to be routed from one service to other service
  - Pipeline company example
- Some partners handle the documents as attachment
- Other partners handle the document in-line

BPEL Extension – Attachments
Solution

- Added extensions to <copy> element
  - “inlined” - makes the binary attachment to base64 in-lined on target variable

```
<copy xmlns:sxat="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Attachment"
            sxat:binaryCopy="inlined">
   <from variable="FileReadVar" part="part1"/>
   <to>$FileWriteVar.part1/ns0:ContentData</to>
</copy>
```

  - “attachment” - transforms in-lined base64 to binary attachment on target variable
  - Default – copy's attachment from-variable to target-variable

Pop Quiz

- What does this conditional BPEL do?

```
<!-- $booVar is type xsd:boolean -->

<assign><from>"false"</from>
        <to>$boolVar</to></assign>

<if><condition>$boolVar</condition>
  <terminate/>
  <else>
    <invoke .../>
  </else>
</if>
```

BPEL Extension – XPath 1.0 Limitations
Problem

- Why is false true? Given schema & document you expect the condition would return false
- XPath 1.0 returns true, as “boolVar” is non empty node

WS-BPEL Extension – Schema awareness
Solution

- BPEL-SE uses JXPath for evaluating XPath expressions. It is enhanced to be schema aware
  - No false true
  - No .0 for integer numbers

BPEL Extension – XPath 1.0 Limitations
Problem

- XPath 1.0 is very limited in functionality
- It is impossible to set a unique id on pay load
- It is challenging to extract expensive items in a purchase order document, it spans multiple activities

<Demo>

BPEL Extension – Java functions
Solution

- Java calls can be embedded in any XPath expression, syntax is adopted from Xalan
  - < from >
  - < to >
  - < condition >
- It facilitates "write your own XPath extension function"
- Java calls also can be embedded in XSLT style sheets invoked from bpel:doXslTransform

<Demo>

BPEL Extension – Javascript
Another Solution

- Native XML Scripting with E4X makes using Javascript very attractive
- Use case “extract expensive items from PO” in Javascript

<Demo>

BPEL Extension–Fault Handler Extensions
Problem

- Standard faults have no error message
  - Tough to debug
- Standard fault versus non-standard
  - Hard to distinguish
  - Complicates fault handlers no end!
    - Multiple catches for standard faults
    - Carries WSDL 1.1 fault model too far

BPEL Extension–Fault Handler Extensions
Solution

- Errors and fault not defined in WSDL propagated as sxeh:systemFault
- Associated errors wrapped in sxeh:faultMessage
- BPEL standard faults are associated with sxeh:faultMessage
  - carries failure details like activity name
- <catch faultMessageType="sxeh:faultMessage" faultVariable="faultVar"> catches all standard and system fault

BPEL feature – Event Handlers
Problem

- While process is in progress
  - Query the status
  - Cancel the process
  - Modify the process

… all asynchronously

BPEL feature – Event Handlers
Solution

- WS-BPEL provides concurrent message processing to normal flow
  - OnEvents - possible in-bound messages
  - OnAlarm - timer events

BPEL feature – Dynamic Addressing
Problem

- Business process needs to provide address to its partner for call back
- Business process determines service provider's address and invokes the service.
  - It could be simple look up based on callers membership level
  - Caller supplies the address in the payload

BPEL feature – Dynamic Addressing
Solution

- WS-BPEL provides a to-spec variation in Assign to set partner's address
- A from-spec variation to get address from myRole

Q&A

Resources

- GlassFish ESB (aka OpenESB)
  https://openesb.dev.java.net/

JavaOne℠

Thank You

Murali Pottlapelli
Murali.Pottlapelli@Sun.COM

Ron Ten-Hove
Captain.Canuck@Sun.COM

Sun microsystems