Business Process Execution Language

Valery Che Ngah
Overview

- Motivation
- WS-BPEL
- Main Concept
- Examples
- Conclusion
Motivation

- Application integration is a key problem facing businesses
  - Intra enterprise integration (Entreprise Application Integration)
  - Integration with partners (Business Process Integration)
- Web Service -> move towards service-oriented computing
  - Application are viewed as "services"
  - Loosely coupled, dynamic interactions
  - Heterogeneous platforms
  - No single party has complete control
- Service composition
  - How do you compose services in this domain?
WS-BPEL

Definition[8]

Business Process Execution Language for Web Services (BPEL or BPEL4WS) is a language used for the definition and execution of business processes using Web services. BPEL enables the top-down realization of Service Oriented Architecture (SOA) through composition, orchestration, and coordination of Web services.
WS-BPEL (Cont.)

- **BPEL enables:**
  - Defining business processes as coordinated sets of Web service interactions.
  - Define both abstract and executable processes.
    - Abstract processes are for e-commerce specifications.
    - Executable processes provide a model to integrating enterprise applications.
  - BPEL enables the creation of compositions of Web services
    - Composition based on abstract descriptions

- **Where it comes from:**
  - Strong roots in traditional flow models.
  - Plus many concepts from structured programming languages.
  - All laid on top of WSDL and core XML specifications.
  - Merges WSFL and XLANG concepts.
Two-level Programming Model

- **Programming in the large**
  - Non-programmers implementing flows
    - Flow logic deals with combining functions in order to solve a more complex problem (such as processing an order)

- **Programming in the small**
  - Programmers implementing functions
    - Function logic deals with a discrete fine-grained task (such as retrieving an order document or updating a customer record)
Process Usage Patterns

- 2 different ways of describing business processes:
  - Executable processes
    - Contain the partner’s business logic behind an external protocol
    - They follow the orchestration paradigm (shown below) and can be executed by an orchestration engine
Abstract processes

- Define the publicly visible behavior of some or all of the services an executable process offers
- Define a process template embodying domainspecific best practices
- Follow the choreography paradigm
Process Model Requirements

- Portability and Interoperability
- Flexible Integration
  - Rich, and easily adaptable to changes in the services it is interacting with
- Recursive, type-based composition, enables
  - third-party composition of existing services
  - providing different views on a composition to different parties
  - inter-workflow interaction
  - increased scalability and reuse
- Separation and composability of concerns
  - Decoupled from the supporting mechanisms (quality of service, messaging frameworks)
- Stateful conversations and lifecycle management
  - Can carry multiple stateful long-running conversations
- Recoverability
  - Business processes, and in particular long running ones, need a way to build-in fault handling and compensation mechanisms to handle and recover from errors
WS STACK

WS-BPEL

WSDL, Policy, UDDI, Inspection

Security

Reliable Messaging

Transactions

Coordination

SOAP (Logical Messaging)

Other protocols

XML, Encoding

Other services

Business Processes

Description

Quality Of Service

Transport and Encoding

Source:[8]
BPEL Language Structure

- Process
- Partner links
- Data handling
- Properties and correlation
- Basic and structured activities
- Scopes
<process>
  <!-- Definition and roles of process participants -->
  <partnerLinks> ... </partnerLinks>
  <!-- Data/state used within the process -->
  <variables> ... </variables>
  <!-- Properties that enable conversations -->
  <correlationSets> ... </correlationSets>
  <!-- Exception handling -->
  <faultHandlers> ... </faultHandlers>
  <!-- Error recovery – undoing actions -->
  <compensationHandlers> ... </compensationHandlers>
  <!-- Concurrent events with process itself -->
  <eventHandlers> ... </eventHandlers>
  <!-- Business process flow -->
  (activities)*
</process>
BPEL Language Structure (Cont.)

- Process
  - BPEL processes are exposed as WSDL services
    - Message exchanges map to WSDL operations
    - WSDL can be derived from partner definitions and the role played by the process in interaction with partners
    - BPEL processes interact with WSDL services exposed by business partners
Partner Links

- Instance of typed connector
  - Partner link type specifies required and/or provided portTypes
  - Channel along which a peer-to-peer conversation with a partner takes place

Source:[1]
BPEL Language Structure (Cont.)

- **Data handling**
  - **Variables**
    - Variables are used to define data containers
      - WSDL messages received from or sent to partners
      - Messages that are persisted by the process
      - XML data defining the process state
    - Constitute the „business“ of the process
  - Access to variables can be serialized to some extent
    - Serializable scopes
Properties and correlation

- How to identify stateful instances via stateless Web service interfaces?
- Messages in long-running conversations are correlated to the correct process instance
  - Typed properties defined in WSDL are named and mapped (aliased) to parts of several WSDL messages used by the process
What is correlation?

- BPEL can model many types of interactions:
  - simple stateless interactions
  - Stateful, long running, asynchronous interactions.
- Correlation sets (CSs) provide support for the latter:
  - CSs represent the data that is used to maintain the state of the interaction (a “conversation”).
  - At the process end of the interaction, CSs allow incoming messages to reach the right process instance.
- What is a correlation set?
  - A set of business data fields that capture the state of the interaction (“correlating business data”). For example: a “purchase order number”, a “customer id”, etc.
  - Each set is initialized once
  - Its values do not change in the course of the interaction.
Activities

Receive
- Wait for a message to be received from a partner
- Specifies partner from which message is to be received, as well as the port and operation provided by the process used by the partner to pass the message

Reply
- Synchronous response to a request corresponding to a receive activity
- Combination of Receive/Reply corresponds to request-response operation in WSDL

Invoke
- Issue an asynchronous request
- Synchronously invoke an in-out operation of a web service provided by a partner
Nesting Structured Activities

<sequence>
  <receive .../>
  <flow>
    <sequence>
      <invoke .../>
      <while ...>
        <assign>...</assign>
      </while>
    </sequence>
  </flow>
  <reply .../>
</sequence>
Disadvantages of BPEL

- Static process composition.
- Process participants (partner’s web services) must be defined and bound to the process flow at design time.
- BPEL standard is not about Semantic Web services:
  - Partner discovery and bounding at run time not possible.
  - Message mediation not possible.
Example: Graph-Oriented Authoring Style

- A customer asks for a loan, providing name and amount info
- Two services are involved
  - A risk assessor which can approve the loan if the risk is low
  - A loan approver which checks the name and approves/disapproves the loan
- The reply is returned to the customer

Source:[8]
Example of BPEL Products

- BPEL Editors
  - Oracle BPEL Designer for JDeveloper
  - Oracle BPEL Designer for Eclipse
  - ActiveBPEL Designer
  - IBM BPWS4J Editor
  - Eclipse BPEL Designer

- BPEL Engines
  - Oracle BPEL Process Manager
  - ActiveBPEL Engine
  - IBM BPWS4J Engine
  - Bexee
  - Twister
# BPEL Engines

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Oracle BPEL Process Manager</th>
<th>ActiveBPEL Engine</th>
<th>ActiveBPEL Enterprise</th>
<th>IBM BPWS4J Engine</th>
<th>Bexee</th>
<th>IBM WebSphere Business Integration Server Foundation</th>
<th>Microsoft BizTalk 2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>J2EE</td>
<td>J2EE</td>
<td>J2EE</td>
<td>J2EE</td>
<td>-</td>
<td>J2EE</td>
<td>.NET</td>
</tr>
<tr>
<td>Current status</td>
<td>In development</td>
<td>In development</td>
<td>In development</td>
<td>Discontinued</td>
<td>Discontinued</td>
<td>In development</td>
<td>In development</td>
</tr>
<tr>
<td>Type of license</td>
<td>Commercial</td>
<td>Open Source</td>
<td>Commercial</td>
<td>Commercial</td>
<td>Open Source</td>
<td>Commercial</td>
<td>Commercial</td>
</tr>
<tr>
<td>Price</td>
<td>$7500-$30000 / CPU</td>
<td>Free</td>
<td>Unknown</td>
<td>unknown</td>
<td>Free</td>
<td>$50000 / CPU</td>
<td>$25000 / CPU</td>
</tr>
<tr>
<td>BPEL version supported</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>-</td>
<td>1.1</td>
<td>?</td>
</tr>
</tbody>
</table>
Conclusion

- It is clear that by itself, Web services is a “great technology” without a practical way to implement it in business. Using BPEL we have a common standard on how to publish multiple services, orchestrate them into business processes and audit and manage the results.
References