WS-BPEL Standards Roadmap

Dieter König, IBM Senior Technical Staff Member
dieterkoenig@de.ibm.com

SOA on your terms and our expertise
WS-BPEL Standards Roadmap

Dieter König, IBM Senior Technical Staff Member
dieterkoenig@de.ibm.com

SOA on your terms and our expertise
WS-BPEL Standards Roadmap

WS-BPEL 2.0 – OASIS
- Web Services – Business Process Execution Language
  - BPELJ – IBM + bea
    - WS-BPEL 2.0 Extensions for Java
  - BPEL4People – IBM + SAP
    - WS-BPEL 2.0 Extensions for Human User Interactions
  - BPEL-SPE – IBM + SAP
    - WS-BPEL 2.0 Extensions for Sub-Processes
WS-BPEL 2.0

Web Services
Business Process Execution Language
WS-BPEL 2.0

Basic Activities
- receive
- reply
- throw
- invoke
- throw
- exit
- wait
- empty
- compensate
- validate
- rethrow
- extensionActivity
- compensateScope

Structured Activities
- flow
- pick
- sequence
- forEach
- while
- if-else
- repeatUntil
- scope

Variables
-receive
-reply
 invoke
-throw
-exit
-wait
-empty
-compensate
-validate
-rethrow
-extendActivity
-compensateScope

Properties
- Correlation Sets
  - Property 1
  - Property 2

Partner Links
- Partner Link Type
  - Port Type 1
  - Port Type 2

Handlers
- event handler
- fault handler
- compensation handler
- termination handler

WSDL Message
- XML Schema Type
- XML Schema Element

OASIS
Advancing E-Business Standards Since 1993
www.oasis-open.org
Abstract

Business Processes not only play a key role in Business-to-Business (B2B) and Enterprise Application Integration (EAI) scenarios by exposing the appropriate invocation and interaction patterns but they are the fundamental basis for building heterogeneous and distributed applications (workflow-based applications). Web Services Business Process Execution Language (WS-BPEL) provides the language to specify business processes that are composed of Web services as well as exposed as Web services. Business Processes specified via WS-BPEL are portable; they can be carried out by every WS-BPEL compliant execution environment. This presentation gives an overview of the WS-BPEL language and shows how it can be used to compose Web services. It provides highlights of WS-BPEL, including structured activities, correlation, compensation, and fault handling. Finally, the OASIS WS-BPEL Technical Committee work, the current status of the standard, and an outlook on follow-on activities is presented.
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
Motivation

• Application integration is a key problem facing businesses
  – Intra enterprise integration (Enterprise Application Integration)
  – Integrating with partners (Business Process Integration)

• Web services → move towards service-oriented computing
  – Applications 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?
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

- Aiming for a single approach for both …
  - Executable processes
    - Contain the partner’s business logic behind an external protocol
  - Abstract processes
    - Define the publicly visible behavior of some or all of the services an executable process offers
    - Define a process template embodying domain-specific best practices
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-BPEL

• WS-BPEL enables …
  – Defining business processes as coordinated sets of Web service interactions, recursively into new aggregated Web services
  – Defining both abstract and executable processes
    • Abstract processes for e-commerce specifications
    • Executable processes provide a model to integrating enterprise applications
  – Creating compositions of Web services
    • Composition based on abstract descriptions

• WS-BPEL provides portable, interoperable process models

• WS-BPEL 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
WS-BPEL Specifications

- BPEL4WS 1.0 (7/2002)
  - Original proposal from BEA, IBM, Microsoft
  - Combined ideas from IBM’s WSFL and Microsoft’s XLANG

  - Revised proposal submitted to OASIS
  - With additional contributions from SAP and Siebel

- WS-BPEL 2.0
  - Currently in OASIS undergoing standardization
  - Committee Draft specification available
WS-BPEL in the 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

You are here

OASIS

Advancing E-Business Standards Since 1993

www.oasis-open.org
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
Getting the Players Together

BPEL4WS 1.1

OASIS

(*) BPEL4WS 1.1 authors, May 2003
Charter of the WS-BPEL Technical Committee

- Standardize …
  - Common concepts for a business process execution language for usage patterns including both the process interface descriptions and executable process models
  - Explicitly do not address …
    - Bindings to specific hardware/software platforms and other mechanisms required for a complete runtime environment for process implementation

- > 280 committee members, including observers
  - ~ 30 active voting members, attending weekly calls

Issues Process

- List of all issues available at
  http://www.choreology.com/external/WS_BPEL_issues_list.html
  - Issue discussion
    - Weekly calls, quarterly face to face meetings
WS-BPEL Design Goals

- Business processes defined using an **XML-based language**
- **Web services** are the model for process decomposition and assembly
- **The same orchestration concepts** are used for both the **external** (abstract) and **internal** (executable) views of a business process
- Both **hierarchical and graph-like** control regimes are used, reducing the fragmentation of the process modeling space
- An **identification mechanism for process instances** is provided at the application message level
- The **basic lifecycle mechanism** is in implicit creation and termination of process instances.
- A long-running transaction model is defined to support **failure recovery** for parts of long-running business processes
- Language built on **compatible Web services standards in a composable and modular manner**
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
WS-BPEL Language Structure

• Process
• Partner links
• Data handling
• Properties and correlation
• Basic and structured activities
• Scopes
BPEL and WSDL

- 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 interactions with partners
Recursive Composition

- BPEL processes interact with WSDL services exposed by business partners

Interfaces exposed by the BPEL process

Interfaces consumed by the BPEL process

BPEL processes interact with WSDL services exposed by business partners.
Composition of Web Services

Service A

A’s WSDL

Partner Link Type

Service P

P’s WSDL

Partner Link Type

Service B

Φ

B’s WSDL
• Partner link: 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
Scoped variables typed as WSDL messages or XML Schema elements/types.

Activities’ input and output kept in scoped variables.

Assignment activities move data around.
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
Basic Activities

- **receive**: Do a blocking wait for a matching message to arrive
- **throw**: Generate a fault from inside the business process
- **reply**: Send a message in reply to a formerly received message
- **rethrow**: Forward a fault from inside a fault handler
- **invoke**: Invoke a one-way or request-response operation
- **exit**: Immediately terminate execution of a business process instance
- **assign**: Update the values of variables or partner links with new data
- **wait**: Wait for a given time period or until a certain time has passed
- **validate**: Validate XML data stored in variables
- **compensate**: Invoke compensation on all completed child scopes in default order
- **empty**: No-op instruction for a business process
- **compensateScope**: Invoke compensation on one completed child scope
- **extensionActivity**: Wrapper for language extensions
Structured Activities

- **flow**
  - Contained activities are executed in parallel, partially ordered through control links

- **sequence**
  - Contained activities are performed sequentially in lexical order

- **while**
  - Contained activity is repeated while a predicate holds

- **repeatUntil**
  - Contained activity is repeated until a predicate holds

- **pick**
  - Block and wait for a suitable message to arrive (or time out)

- **forEach**
  - Contained activity is performed sequentially or in parallel, controlled by a specified counter variable

- **if-else**
  - Select exactly one branch of activity from a set of choices

- **scope**
  - Associate contained activity with its own local variables, fault handlers, compensation handler, and event handlers
<sequence>
  <receive .../>
</flow>
  <sequence>
    <flow>
      <sequence>
        <invoke .../>
        <while ... >
          <assign>...</assign>
        </while>
      </sequence>
      <sequence>
        <receive .../>
        <invoke ... />
      </sequence>
    </flow>
  </sequence>
  <reply .../>
</sequence>
Scopes and Handlers

• Scope
  – Local variables
  – Local partner links
  – Local correlation sets
  – Set of activities (basic or structured)

• Handlers
  – Event handlers
    • Message events or timer events (deadline or duration)
  – Fault handlers
    • Dealing with different exceptional situations (internal faults)
  – Compensation handler
    • Undoing persisted effects of already completed activities
  – Termination handler
    • Dealing with forced scope termination (external faults)
Process Instance Lifecycle

- Business processes defined in BPEL represent stateful Web services
  - When a process is started, a new instance is created according to the process definition
  - The creation and destruction of BPEL process instances is by design implicit
WS-BPEL 2.0

- Motivation
- OASIS and WS-BPEL
- Main Concepts
- Examples
- Status and support
1. A customer asks for a loan, providing name and amount info
2. Two services are involved:
   a) A risk assessor which can approve the loan if the risk is low
   b) A loan approver which checks the name and approves/disapproves the loan
3. The reply is returned to the customer
if \( \text{buyer} < \text{seller} \)

else

receive (buyer)

receive (seller)

assign "success"

assign "failure"

reply (seller)

reply (buyer)
Parallel Processing

- flow

- parallel forEach

- scope
Dynamic Partner Link Assignment

Process A
- assign EPR
- invoke service
- receive response
- partner link

Process B
- receive request
- assign EPR
- invoke callback
- partner link

Partner Link Type
- A-role
- B-role

PortType-A
PortType-B
Executable Processes View

**Traveler Process**
- Plan trip
- Submit to agent
- Receive confirmation
- Receive tickets

**Agent Process**
- Get itinerary
- Select airline
- Order tickets
- Receive confirmation
- Send confirmation
- Send tickets

**Airline Process**
- Get order
- Reserve seats
- Charge credit card
- Confirm flight
- Send tickets
WS-BPEL 2.0

• Motivation
• OASIS and WS-BPEL
• Main Concepts
• Examples
• Status and support
WS-BPEL

- Portable, interoperable process model for long running business processes
- Flexible integration of Web services
  - WSDL abstract interfaces alone used to define composition
    - Enables two levels of adaptive behavior
      - Abstract partners can be bound to actual services at runtime
      - The process can choose a protocol for communicating with the service at runtime
    - Services whose data definitions do not match can be composed
      - Data transformations can be inlined in process definition
WS-BPEL Adoption: Products

- Active Endpoints ActiveWebflow Designer
- Active Endpoints ActiveWebflow Enterprise Server
- ActiveBPEL Engine (open source)
- BEA WebLogic
- bexe BPEL Execution Engine (open source)
- Cape Clear Orchestrator
- FiveSight Process eXecution Engine (PXE)
- IBM WebSphere Studio Application Developer – Integration Edition
- IBM WebSphere Business Integration – Server Foundation
- IBM WebSphere Integration Developer
- IBM WebSphere Process Server
- Microsoft BizTalk Server
- MidOffice BPEL Editor (open source)
- MidOffice BPEL Engine (open source)
- OpenLink Virtuoso Universal Server
- OpenStorm ChoreoServer
- Oracle BPEL Process Manager
- Parasoft BPEL Maestro
- SAP NetWeaver
- SeeBeyond eInsight BPM
- Twister (open source)
WS-BPEL Application Areas

• Business Process Design
• Autonomic Computing
• Grid Computing
• Semantic Web
What’s new since BPEL4WS 1.1?

• New activity types
  – if-then-else – replacing the BPEL4WS 1.1 switch activity
  – repeatUntil – like BPEL4WS 1.1 while activity with at least one iteration
  – validate – explicit XML schema validation of WS-BPEL variable content
  – forEach – sequential or parallel iteration controlled by a counter variable
  – compensateScope – compensate one specific child scope
  – extensionActivity – designated WS-BPEL extension point for new activity types

• Completion condition in forEach activity
• Variable initialization
• XSLT for variable transformations
  – New XPath extension function \texttt{bpel:doXslTransform(...)}
• XPath access to variable data
  – XPath variable syntax $\texttt{variable[.part]/location}$
• XML schema variables in Web service activities
  – Usability enhancement for WS-I compliant doc/lit-style WS interactions
• Scope termination handler
• Locally declared messageExchange
  – Internal correlation of receive and reply activities
• Abstract processes
  – Common base (syntax) and profiles (semantics)
WS-BPEL 2.0 Schedule

• Status (Feb 2006)
  ~ 230 of 240 issues resolved
  – Deadlines (need 2/3 majority to override)
    • No new feature issues since Aug 15, 2004
    • No new feature issue resolution proposals since April 1, 2005
    • Feature issues that are not resolved are marked as revisitable
  – Latest approved committee draft: December 21, 2005

• Next steps (Technical Committee roadmap)
  – Approved committee draft → March 15, 2006
  – Approved public review draft → May 1, 2006
  – Final committee specification → August 30, 2006
  – OASIS standard → October 15, 2006
Follow-on Work

• **BPELJ**
  – Inline Java code in activities and expressions

• **BPEL4People**
  – Human user interactions

• **BPEL Subprocesses**
  – Based on a coordination protocol

• **Transaction semantics**
  – Aligned with WS-Transaction specifications

• **Currency with related standards**
  – WSDL 2.0, XQuery, etc.
WS-BPEL Resources

• OASIS Technical Committee
  http://www.oasis-open.org
• BPEL4WS 1.1
  http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp
  http://ifr.sap.com/bpel4ws/
  http://www.siebel.com/bpel
• WS-BPEL 2.0 – latest approved committee draft (December 2005)
• Info aggregator sites
  – Wikipedia
    http://en.wikipedia.org/wiki/BPEL
  – BPEL Resource Guide
    http://bpelsource.com
• Numerous books and conference papers
• Analyst reports
WS-BPEL Standards Roadmap

- **WS-BPEL 2.0** – [OASIS](https://www.oasis-open.org)
  - Web Services – Business Process Execution Language

- **BPELJ** – [IBM](https://www.ibm.com) + [bea](https://www.bea.com)
  - WS-BPEL 2.0 Extensions for Java
    - **BPEL4People** – [IBM](https://www.ibm.com) + [SAP](https://www.sap.com)
      - WS-BPEL 2.0 Extensions for Human User Interactions
    - **BPEL-SPE** – [IBM](https://www.ibm.com) + [SAP](https://www.sap.com)
      - WS-BPEL 2.0 Extensions for Sub-Processes
```
bpel:extensionActivity

bpelj:snippet

return
"Hello world !!!";
```
WS-BPEL Extensions for Java (BPELJ)

- Authors
  - Gerhard Pfau
  - Anke Robeller
  - Andreas Schmitz

- Authors
  - Michael Rowley
  - Alexandre Alves
BPELJ Technical Goals

- Include inline Java code in BPEL processes
  - Activities
  - Conditions
  - Variable initialization

- Orchestrate long-running interactions with Java components

- Support advanced transactional capabilities
  - Atomic Scopes and Atomic Processes
BPEL and Java

- **BPEL**
  - Describe the logic of business processes
  - Maintain multiple long-running units of execution that are also interruptible
  - Selectively compensate completed activities of long-running units of execution in case of failures
  - Route incoming messages to the right place within the right process
  - Accept one of a variety of possible expected incoming message types.
  - Define a set of activities that should occur at a designated time and in pre-defined order.
  - Send messages to Web services.

- **Java**
  - Calculate a value to be inserted into a document.
  - Construct a document to be sent to a web service using information from other documents and variables.
  - Deconstruct a document that has arrived – finding important values, converting them and then inserting them into other documents.
  - Calculate a value that will be used to affect the flow of control within the business process (e.g. loops and branches)
  - Perform side-effects without having to create Web services
  - Manipulate multiple transactional resources from within a single ACID transaction.
BPELJ Examples – Inline Java Expressions

- **Condition (Boolean Expression)**

```xml
<if>
  <condition
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
      widget.equals(getVariableProperty("PO", "productName"))
  </condition>
  ...
  <else>...</else>
</if>
```

- **Variable initialization (General Expression)**

```xml
<variable name="salesTax" type="xsd:float">
  <from
      expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:java1.4">
      subtotal * taxRate
  </from>
</variable>
```
BPELJ Examples – Inline Java Activity

- Extension Activity

```xml
<process name="purchaseOrderProcess" bpelj:xmlBinding="bpelj:DOM3" ...>
  <variables>
    <variable name="justificationDoc" type="lns:justificationDocument"/>
    <variable name="po" type="lns:POMsg" bpelj:xmlBinding="bpelj:SDO1.0"/>
  </variables>
  <sequence>
    ...
    <extensionActivity>
      <bpelj:snippet>
        // Get the approver using SDO accessor
        Approver approver = po.getApprover();
        // Get the approver's comments
        NodeList commentNodeList =
          justificationDoc.getElementsByTagName("approverComment");
        ...
      </bpelj:snippet>
    </extensionActivity>
  </sequence>
</process>
```
BPELJ Status and Progress

- IBM-BEA Whitepaper published in March 2004
  

- Specification completed (October 2005)

- **Target:** Publish BPELJ spec right after WS-BPEL 2.0
WS-BPEL Standards Roadmap

- WS-BPEL 2.0 – Web Services – Business Process Execution Language
  - BPELJ – IBM + bea
    - WS-BPEL 2.0 Extensions for Java
  - BPEL4People – IBM + SAP
    - WS-BPEL 2.0 Extensions for Human User Interactions
- BPEL-SPE – IBM + SAP
  - WS-BPEL 2.0 Extensions for Sub-Processes
WS-BPEL 2.0 Extensions

Authors
- Matthias Kloppmann
- Dieter König
- Frank Leymann
- Gerhard Pfau

Authors
- Alan Rickayzen
- Claus von Riegen
- Patrick Schmidt
- Ivana Trickovic
Abstract

- Human user interactions are currently not covered by the Web Services Business Processes Execution Language (WS-BPEL), which is primarily designed to support automated business processes based on Web services. In practice, however, many business process scenarios require user interaction. This paper describes scenarios where users are involved in business processes, and defines appropriate extensions to WS-BPEL to address these.
Human User Interactions – Goals

- Human user interactions as part of a BPEL process
  - Simple scenarios, such as manual approval
  - Complex scenarios where the data input will be performed by the human user
- Independently defined human tasks reused in a BPEL process
Human User Interactions – Scenarios

- **People activities**
  - Special kind of activity “implementation”

- **People initiating processes**
  - People eligible to starting a certain business process

- **People managing long-running processes**
  - User intervention after failed people resolution or timeout

- **Transition between human and automatic services**
  - Services are performed with or without human interaction

- **Advanced interaction patterns**
  - Four eyes principle (separation of duties) – decision made by two or more people
  - Escalation – person working on a task suddenly became ill or is just overwhelmed by work
  - Nominations – supervisor nominates task ownership to a colleague with the expertise that best matches the task
  - Chained execution – a sequence of steps executed by the same person
Web Service vs. Human Task

Translation Process
- receive document
- automatic translation
- manual translation
- reply translation

Web Service Endpoint

Abstract Organizational Group
People Resolution

- **Generic human roles**
  - Process initiator – person that actually creates an instance of the process
  - Process stakeholder – person who can influence the progress of a process instance
    - Add ad-hoc attachments
    - Forward a work item
    - Observe the progress of the process instance
  - Potential owners – people who are entitled to claim and complete the activity
  - Owner – potential owner who explicitly claimed the activity
  - Business administrator – person who is allowed to perform administrative actions on the business process like resolving failed people resolution

- **People links**
  - Represent groups of people who participate in the execution of the process
    - Can be parameterized in order to pass data from the process instance to a people query
    - May be reused in different steps of the process
    - Qualifies a generic human role

- **People resolution**
  - The act of identifying people who take on the responsibility for a particular generic human role
People Links and People Resolution

Brochure Creation Process

- **create**
  - "Authors"
    - Select staff
      - Where qualification = "tech writer"

- **approve**
  - "Approvers"
    - Select staff
      - Where responsibility = "marketing"

Org Database

- Departments
  - Department1
    - Member1
    - Member2
  - Department2
    - ...

- Users
  - Group1
    - Member1
    - Member2
  - Group2
    - ...

- Roles
  - Role1
    - Member1
    - Member2
  - Role2
    - ...

SOA on your terms and our expertise
People Activities and Tasks

- People activity
  - Basic activity in the process “implemented” by an action performed by a human being
  - The “partner” of a people activity is specified by a people link
  - May be associated with different groups of people, one for each generic human role

- Task
  - Indivisible unit of work, performed by a human being
  - Properties: priority, input/output data, deadlines
  - Operations for client applications: query available tasks, claim task, revoke claim, complete task
  - States: ready, claimed, finished, failed
  - Location of task definition: inline or standalone
People Activities and Tasks

- Inline Task Definition
  - Task can only be used inside of process context
  - Task is interoperable as well as portable

- Standalone Task Definition
  - Task can be used outside of process context
  - Communication with task is based on coordination protocol
  - Task is interoperable as well as portable

- Invoke activity
  - Task is just a Web service with no additional constraints
  - Easy to switch between performing the task by humans or programs
BPEL Process with Inline Task

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bpel:process ... >
  ...
  <b4p:peopleLinks>
    <b4p:peopleLink name="worker">
      <b4p:argument name="region" type="xsd:string"/>
      <b4p:argument name="city" type="xsd:string"/>
    </b4p:peopleLink>
    ...
  </b4p:peopleLinks>
  ...
  <bpel:extensionActivity name="MyPeopleActivity">
    <!-- People Activity -->
    <b4p:peopleActivity inputVariable="in" outputVariable="out">
      <b4p:task name="MyInlineTask"
        description="This task is used to do something really useful."
        priority=42
        dueDate=$dueDateForTaskX ...
        <b4p:peopleLink ref="worker" genericHumanRole="potential-owner">
          <b4p:argument ref="region">
            <from variable="input" property="xyz:region"/>
          </b4p:argument>
          <!-- argument "city" defined during deployment -->
        </b4p:peopleLink>
        ...
      </b4p:task>
    </b4p:peopleActivity>
  </bpel:extensionActivity>
  ...
</bpel:process>
```
Standalone Task

- May be invoked outside of a BPEL process
- May be invoked from a BPEL process
  - People activity – refers to the standalone task definition
  - Invoke activity – Web service implemented by the standalone task

```xml
<?xml version="1.0" encoding="UTF-8"?>
<b4p:task name="MyStandaloneTask"
    description="This task is used to do something really useful."
    priority=42 ...>
  <interface portType="foo:MyFooPortType" operation="myOperation"/>
  <b4p:peopleLink name="worker" genericHumanRole="potential-owner">
    <b4p:argument name="region" type="xsd:string" value="$input/region"/>
  </b4p:peopleLink>
  ...
</b4p:task>
```
Operations on Tasks

- Query available tasks
  - Return a list of unfinished work or claimed tasks
- Claim task
  - Take over the responsibility for a task
- Revoke claim
  - Abandon the responsibility for a task – the task is returned to the potential owners
- Complete task
  - Successfully finish work on a claimed task
- Fail task
  - Finish task indicating failure
WS-BPEL Standards Roadmap

- **WS-BPEL 2.0** – [OASIS]
  - Web Services – Business Process Execution Language
- **BPELJ** – [IBM] + [bea]
  - WS-BPEL 2.0 Extensions for Java
- **BPEL4People** – [IBM] + [SAP]
  - WS-BPEL 2.0 Extensions for Human User Interactions
- **BPEL-SPE** – [IBM] + [SAP]
  - WS-BPEL 2.0 Extensions for Sub-Processes
WS-BPEL Extensions for Sub-Processes

- Authors
  - Matthias Kloppmann
  - Dieter König
  - Frank Leymann
  - Gerhard Pfau

- Authors
  - Alan Rickayzen
  - Claus von Riegen
  - Patrick Schmidt
  - Ivana Trickovic
Abstract

- Designing complex and large business processes requires a language that supports modularization and re-use, in a portable, interoperable way. This paper outlines an extension to WS-BPEL that allows for the definition of sub-processes that can be reused within the same or across multiple WS-BPEL processes. The paper describes different invocation scenarios and introduces an appropriate coordination protocol used for interoperable invocation of sub-processes across infrastructures from different vendors.
Sub-Processes – Definition and Invocation

WFMS 1
- Parent Process
- Inline Sub-Process
- Local Sub-Process
- Remote Sub-Process

WFMS 2

call

call

call
Sub-Process Invocation – Request-Response Operation

call ...
partnerLink=...
requestOperation=op

op

receive (input)

reply (output)

Partner Link Type
Sub-Process Invocation – Two One-Way Operations

- Parent Process
  - subp:call
  - EPR(op')
  - Partner Link Type
    - partnerLink=...
    - requestOperation=op
    - responseOperation=op'

- Sub-Process
  - receive (input)
  - invoke (output)
  - op
  - op'

call ...
partnerLink=...
requestOperation=op
responseOperation=op'
Definition of a Standalone Sub-Process

- Regular BPEL process, with appropriate restrictions
  - Single initiating receive activity, consuming the input message of the implemented operation
  - “Logical request-response operation”
    - One request-response operation (response returned via reply activity)
    - Two one-way operations (response returned via invoke activity)
  - The reply/invoke activity is “the last” operation of the sub-process
  - No interaction beyond the single call-return between the sub-process and its parent process
Definition of an Inline Sub-Process

- Standard lexical visibility rules
- Similar to standalone sub-processes, it must implement a single logical request-response operation
- If no data needs to be passed into the sub-process, it doesn’t need to have any receive (or reply) activity at all

```xml
<bpel:scope>
  <bpel:variables>
    <bpel:variable name="input" messageType="..."/>
    <bpel:variable name="output" messageType="..."/>
  </bpel:variables>
  <subp:sub-processes>
    <bpel:process name="mysub-process">
      <!-- Standard process definition with appropriate limitations/relaxations -->
    </bpel:process>
    ...
  </subp:sub-processes>
  ...
</bpel:scope>
```
Invocation of a Sub-Process

- Syntax is based on BPEL invoke activity syntax
- Standalone sub-process invocation

```xml
<subp:call name="handleOrder"
    partnerLink="myOrderProcess"
    requestOperation="submitOrderer"
    inputVariable="myOrder"
    outputVariable="orderingResult"
    responseOperation="receiveOrderResult"/>
```

- Inline sub-process invocation

```xml
<subp:call xmlns:s="http://stuff-is-us.com"
    process="s:myProcess"
    inputVariable="var1"
    outputVariable="var2"/>
```
Compensation of a Sub-Process

- The sub-process is treated like a scope
  - One may specify a compensation handler at the process level which is invoked after completion of the process in order to undo effects of the execution of the process

- The call activity is treated as a proxy for the sub-process
  - It can be the target of a compensate activity
  - Default compensation behavior is propagated to call activities as well

```xml
<scope>
  <compensationHandler>
    <sequence>
      <compensateScope target="handleOrder"/>
    </sequence>
  </compensationHandler>
  <call name="handleOrder"
       partnerLink="myOrderProcess" operation="submitOrdner"
       inputVariable="myOrder" outputVariable="orderingResult"/>
</scope>
```
WS-BPEL Standards Roadmap

- WS-BPEL 2.0 – OASIS
  - Web Services – Business Process Execution Language
- BPELJ – IBM. + bea
  - WS-BPEL 2.0 Extensions for Java
- BPEL4People – IBM. + SAP
  - WS-BPEL 2.0 Extensions for Human User Interactions
- BPEL-SPE – IBM. + SAP
  - WS-BPEL 2.0 Extensions for Sub-Processes