

OASIS WS-BPEL in brief

Peter Furniss

Choreology Ltd

peter.furniss@choreology.com

- Convergence of IBM's WSFL and Microsoft XLANG
- First published as BPEL4WS August 2002
 - Microsoft, IBM, BEA
 - with WS-Coordination, WS-Transactions
- Version 1.1, May 2003
 - IBM, Microsoft, BEA, SAP, Siebel
 - Input to OASIS technical committee, started May 2003
- TC chartered to produce revised edition in Feb 2004 !!
 - at 3 Dec 2003, 72 open issues, 15 resolved
 - issue list:
http://www.choreology.com/external/WS_BPEL_issues_list.html
- First internal working draft, 3 December 2003
 - not "Committee Draft", just first consolidation

- chartered May 2003
- 69 voting members
- ~ 40 companies
 - IBM, Microsoft, Oracle, Sun, HP, SeeBeyond, SAP, Choreology, numerous others
- 220+ on email list
- fortnightly conference calls
- 3rd face-to-face, Florida, 9-10 December
- <http://www.oasis-open.org/committees/wsbpel>
 - most documents and all email archives open
 - participation requires OASIS membership

- Pronunciation – rhymes with people
- Committee song:
“People, People who need BPEL”

(Proposed Diane Jordan)

- XML language for defining behaviour of a process
 - that provides web-service(s)
 - that uses web-services
 - everything it sees is a web-service
 - no other external interactions
- Graphical and other tools can map to and from it
 - not in TC scope

What do you mean by “web-service”

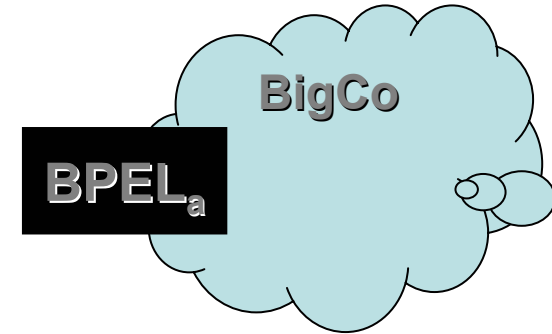
- BPEL answer:
 - anything that has WSDL is a web-service
 - all BPEL interactions are via WSDL-defined interfaces
 - BPEL “sees” only the abstract port-type definition, not the binding or port
 - bindings to non-HTTP and non-SOAP expected
 - EDI, MQ, java rmi,
 - also “local” binding – no transport
 - complication with non-standard bindings to exotica

- abstract processes - public behaviour
 - define “business protocols”
 - hide things that do not affect partner
 - constrain only the message exchange
 - what the possible replies are, not why one is chosen
- executable processes - private behaviour
 - fully define behaviour
 - portable between compliant environments
- one language, with minor specialisations
(most TC discussion concerns executable)

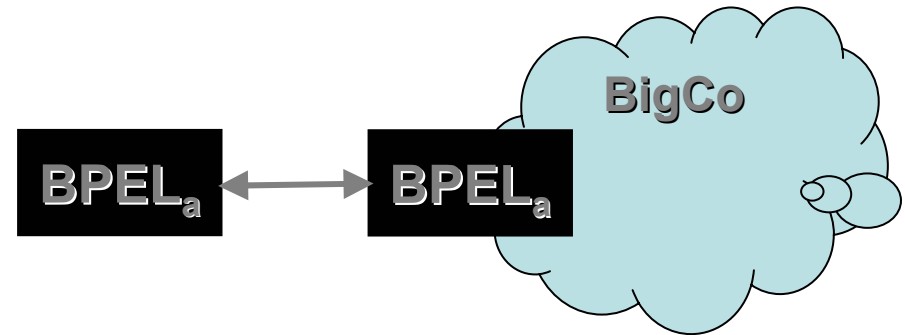
Abstract BPEL - use cases

from presentation by Tony Andrews, Microsoft, at BPEL TC first f-t-f

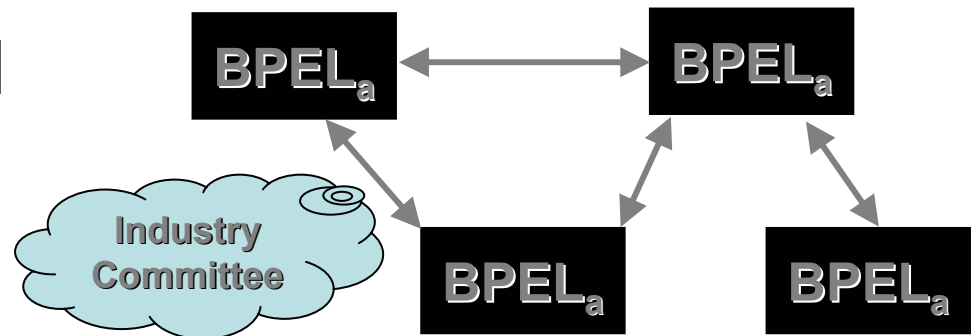
- “Here’s how I behave – work with me!”



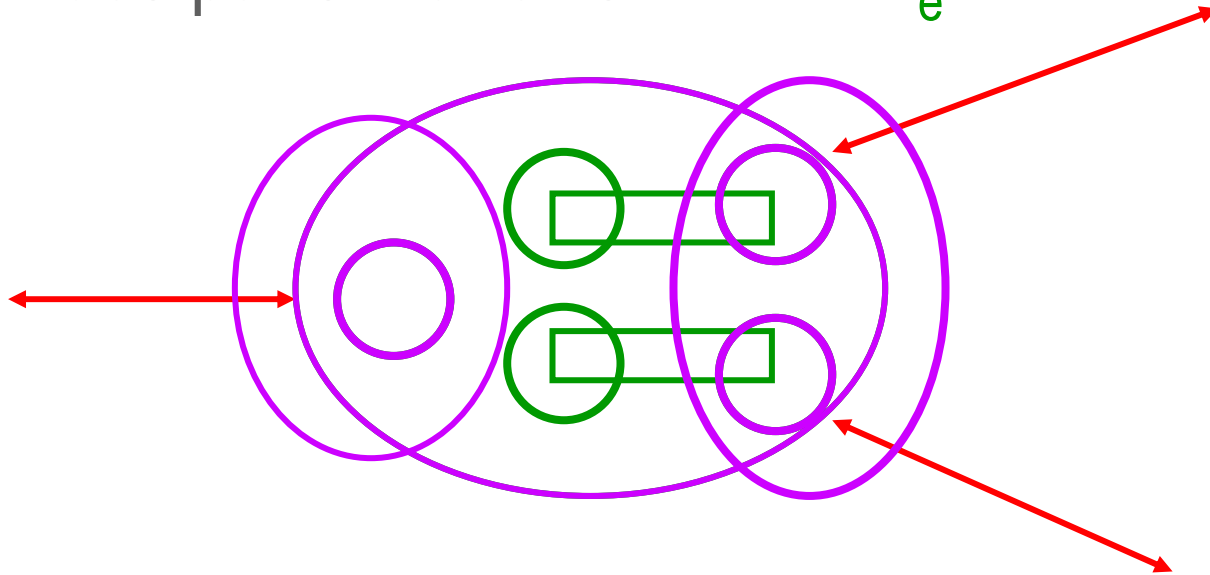
- “Here’s how I behave – please behave like *this*”



- “It would be great if we all worked together like *this*”



- $BPEL_a$ hides parts that exist in $BPEL_e$

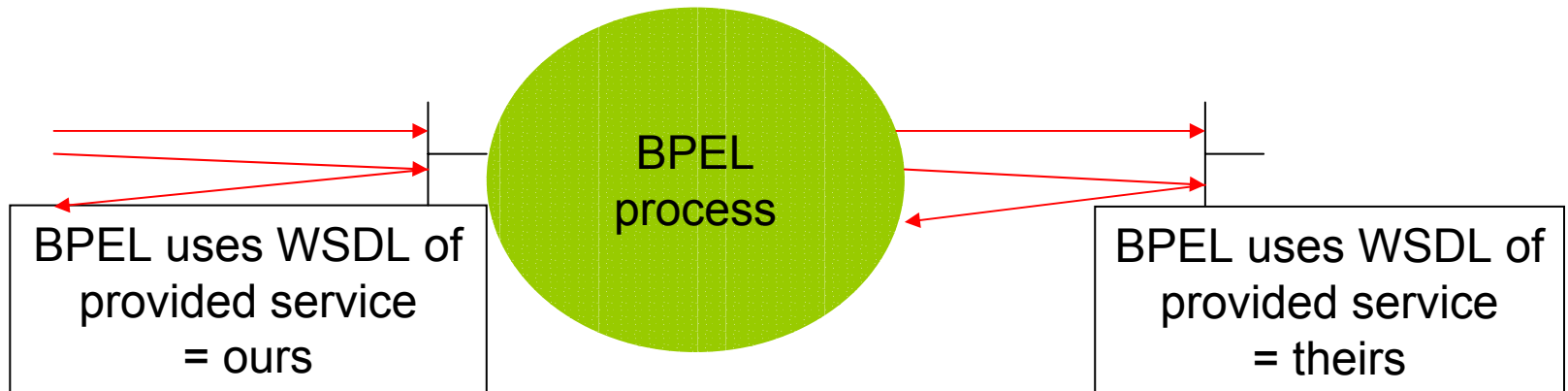


- Using $BPEL_a$ for all parties implies that some abstract processes will not be implemented in $BPEL_e$
 - someone's got to do the real work
 - interact with the user
 - work with the database
 - move the telescope
- $BPEL_a$: $BPEL_e$ - could be automated
 - abstract to executable skeleton; extract behaviour
- non-BPEL : $BPEL_a$ - manual
- motivated selection when going to $BPEL_a$

Main pieces of a BPEL document

- Communication - offering and using web-services
 - inbound and outbound
- Partners
 - who we do things with
- Variables
 - what is communicated
- Correlation sets
 - instance identity - how do we know who we are
- Activities
 - what we do
- Handlers
 - coping with the (slightly) unexpected

- WSDL Message Exchange Patterns
 - WSDL 1.1 has several
 - common practice / Basic Profile 1.0 is just to use
 - in-out = request-response offered
 - in = one-way received
 - BPEL follows this limitation
 - refers to own and others services by the initial receiver's wsdl



- incoming:

```
<receive partnerLink="purchasing"  
  portType="tns:purchaseOrderPT"  
  operation="sendPurchaseOrder"  
  variable="PO">  
</receive>
```

- from whoever is the other role in “purchasing”
- involving the portType and operation given
- keep the inbound message in our variable “PO”
- can have other stuff in the body – e.g. correlation sets
- used for
 - in of our wsdl in-out – incoming ‘synchronous’ request
 - in of our wsdl in – incoming one-way
 - includes semantic reply to an outgoing one-way

- reply to a “synchronous” receive:

```
<reply partnerLink="purchasing"  
  portType="Ins:purchaseOrderPT"  
  operation="sendPurchaseOrder"  
  variable="Invoice"/>
```

- back to whoever is the other role in “purchasing”
- involving the portType and operation given
- load the outbound message from our variable “Invoice”

- using an asynchronous (wsdl in) service:

```
<invoke partnerLink="invoicing"
  portType="Ins:computePricePT"
  operation="initiatePriceCalculation"
  inputVariable="PO">
</invoke>
```

 - invoke on whoever is the other role in “invoicing”
 - involving the portType and operation given
 - load the *outbound* message from our variable “PO”
- any reply will be a separate receive
 - same partnerLink
 - but myRole portType

- using a synchronous (wsdl in-out) service:

```
<invoke partnerLink="shipping"  
  portType="tns:shippingPT"  
  operation="requestShipping"  
  inputVariable="shippingRequest"  
  outputVariable="shippingInfo">  
</invoke>
```

- to whoever is the other role in “shipping”
- involving the portType and operation given
- load the outbound message from our variable “shippingRequest”
- store inbound message in our variable “shippingInfo”

- defines a type of relationship or conversation
- gives role name to a portType
- can tie two opposite portTypes together, as asynchronous conversation pair

```
<plnk:partnerLinkType name="shippingLT">  
  <plnk:role name="shippingService">  
    <plnk:portType name="pos:shippingPT"/>  
  </plnk:role>  
  <plnk:role name="shippingRequester">  
    <plnk:portType name="pos:shippingCallbackPT"/>  
  </plnk:role>  
</plnk:partnerLinkType>
```

- named instance of a partnerLinkType
 - could be multiple partnerLinks of same type
- states which is me and which is him in a conversation

```
<partnerLink name="shipping"  
  partnerLinkType="Ins:shippingLT"  
  myRole="shippingRequester"  
  partnerRole="shippingService"/>
```

- typed
 - WSDL message
 - XML schema simple types
 - XML scheme element
- manipulation
 - set from inbound, to outbound messages
 - assign from/to
 - other variables
 - parts & properties of variables of type wsdl message
 - partnerLink endpoints
 - simple expressions
 - literals
 - Xpath expressions to get inside complex variables

- one process definition may have lots of instances
 - which message is for/from which instance ?
 - don't require/rely on environment or carrier protocol to identify
- define which fields of the messages distinguish the instance
 - e.g. purchase order number; username + their taskid
 - fields to be used are declared as properties of a message variable
 - properties defined as bits of variable using XPath (or XQuery)
 - if there **is** context id or request id field – use that
 - this may require bending the wsdl

- **structured activities – can contain other activities**
 - <sequence>** one after the other
 - <flow>** in parallel
 - <pick>** choose by inbound message
 - <switch>** choose by expression evaluation
 - <while>** iteration
 - <scope>** nest, with declarations and handlers, synchronize
- **communication**
 - <invoke>** send msg to partner; possibly receive response
 - <receive>** accept msg from partner
 - <reply>** send msg to partner as response to <receive>
- **other**
 - <assign>** manipulate variables
 - <wait>** for duration / until time
 - <terminate>** end the process
 - <compensate>** run compensation handler of inner scope
 - <throw>** exit with fault to outer scope
 - <empty>** do nothing

- support Directed Activity Graph style
 - activities can be source and target of links
 - activity with target links does not run till source completes normally
 - links can cross structured activity boundaries

Why links AND structure ?

BPEL is merge of two specifications and approaches

- handlers are declared for process or scope
- “watching” for the lifetime of the scope:
 - eventHandler
 - onMessage – a <receive> that could happen any time
 - onAlarm – time dependent
 - faultHandler
 - catches <throw>n or generated fault
 - scope exits abnormally
- “watching” after the scope has exited normally
 - compensationHandler

- compensationHandler is installed when scope ends
- triggered from faultHandler or compensationHandler of enclosing scope
- uninstalled when process ends

BUT

- whole process can have a compensationHandler
- triggered by unspecified means
- uninstalled when [unspecified]

BTM = Business Transaction Management

- BPEL currently shares model with WS-Transaction Business Activity, version 1
 - but no constructs to use or control WS-T BA
- Choreology input to BPEL TC
- between services, compensation is too specific
 - reversible is contingent
 - contingent is waiting for one of confirm, cancel
 - add a confirmHandler, at least at process level
- bpel activity to initiate, terminate or register in business transaction

- bits of BPEL should be re-usable
 - sub-functions – scopes with parametersOR
 - separate processes
- scope : process differences
 - communication – global variable or invoke/receive
 - compensation handling – specified or magic
- if process-level compensation is sorted out (= coordination mechanism), separate processes is much nicer

- OASIS WS BPEL Technical Committee top page
 - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- WS-BPEL Issues list
 - http://www.choreology.com/external/WS_BPEL_issues_list.html
- Presentations from the first Technical Committee face-to-face, May 2003.
 - These are links to emails to the bpel list containing publically accessible copies of the talks. The presentations are in links towards the bottom of the archived emails.
 - Dieter Roller (IBM): BPEL4WS introduction
 - <http://lists.oasis-open.org/archives/wsbpel/200305/msg00168.html>
 - Satish Thatte (MSFT) : BPEL Goals and nuances
 - <http://lists.oasis-open.org/archives/wsbpel/200305/msg00170.html>
 - Tony Andrews (MSFT) : Abstract Processes
 - <http://lists.oasis-open.org/archives/wsbpel/200305/msg00172.html>
- Choreology submission on bpel and business transaction management
 - <http://www.oasis-open.org/committees/download.php/3263/BPEL.and.Business.Transaction.Management.Choreology.Submission.html>