pi4soa Implementation Issues WS-CDL Candidate Recommendation

December 2005

Pi4 Technologies Ltd

hasChoreographyCompleted

- It is not clear what states constitute 'completed'
- The test implies 'successfully-completed' and 'unsuccessfully-completed'
- However, what happens if a sub-choreography completes with no finalizers, it automatically transitions to the 'closed' state – which would currently not be detected by this function
- Suggestion: change function description to explicitly list states, and include 'closed' state

Perform bind variable types

- Need a rule to ensure that the types of the 'this' and 'free' variables are the same type
- Also, the text says that the variable should use the 'getVariable' function, but should also say that the 'partName' should not be specified – i.e. It should not be possible to bind parts.

Choice

- Current text says only one element, out of two or more, SHOULD be selected
- What if choice contains all non-blocking conditional elements and none of them evaluate to true?
- What if you have non-observable conditions around choice elements that are interactions? These non-observable conditions could equally evaluate to false.
- Semantics of choice need to be clear from a monitoring and execution perspective, therefore would suggest changing SHOULD to MUST.

Exchange type

- Currently the type field is optional
- However, if the associated channel type has identity information, then without the message type, it is not possible to determine how the message will be correlated with the channel instance and choreography session
- Possibly we could add a rule that indicates that the type must be specified if the channel type has identity details, as these details would only be added when intending to execute/monitor choreo, instead of just as a description

Service endpoint projection

- Currently difficult to link a Qname for the endpoint description with an element in the WS-CDL description
- This will be required if we wish to estabslish link between the global and endpoint model, for the purpose of validation and reuse
- pi4soa projects endpoints based on ParticipantType, and therefore names the services after the ParticipantType
  - This is not a QName and therefore limits the scope of names that can be used for services
  - Not a good long term solution for reuse of endpoint descriptions
  - Possibly participant and role types should be changed from NCName to QName?

Channel Passing (1)

- Business messages can be correlated to a channel instance and choreography session using identity information derived (through the use of token locators) from the message contents
- However, channels passed over other channel instances do not carry any business information, and therefore how can they be correlated?
- Approach taken in pi4soa is currently restricted to use of WS-Addressing endpoint reference, which it stores the identity token values of the channel on which the exchange is occurring – however this is not interoperable, as it relies on prior knowledge of the identity token encoding within the endpoint reference

Channel Passing (2)

- Channel passing is also restrictive, as it can only be passed within its own exchange
- This is currently necessary to enable the formal verification to identify where the channels are being passed
- However, for subsequent versions of CDL, we should investigate the ability to describe message exchanges where a channel's details are passed as part of a business message, but still remain explicit enough for formal verification.

Multipart variable with single part

- Currently, if a multipart message type has only a single part, it still needs to be accessed using the part name – this is not portable between a WSDL1.1 and WSDL2.0 version of the same interface
- Suggestion: change the spec to indicate that if multipart message has only single part, then part name is optional

Activity names

- Not all activity types have a 'name' attribute
- This causes a problem when associating implementation hooks to the activity
- Important example is silentAction

hasExchangeOccurred

- Example: request for quote, while waiting for suitable quote (on refresh loop), until quote accepted

  -
    ```
    Interaction: --> request for quote
    Interaction: <-- quote
    workunit guard=”hasDurationPassed(30sec)” block=”true”
                                  repeat=“!quoteAccepted” {
         Interaction <-- updateQuote
    }
    Interaction: --> quote accepted
    ```
- Currently only possible with parallel construct setting state variable to indicate when quote is acceptable, as quote provider as no means to detect when quote has been accepted, and therefore break out of the loop
- Suggestion: having a 'hasExchangeOccurred' to finish the loop once the quote accept has been received

hasExchangeOccurred (2)

-
  ```
  Send rfq
  Receive quote
  Parallel{
      when quoteAcc
          Send quoteAcc
          [rec quoteAcced]
      when refresh &&
          !quoteAcced ..
  }
  ```

  ```
  Receive rfq
  Send quote
  Parallel{
      when .....
          Receive quoteAcc
          [rec quoteAcced]
      when refresh &&
          !quoteAcced
  }
  ```