An Ontology for Executable Business Processes

Jörg Nitzsche, Daniel Wutke, and Tammo van Lessen

Institute of Architecture of Application Systems
University of Stuttgart
Universitaetsstrasse 38, 70569 Stuttgart, Germany
{joerg.nitzsche,daniel.wutke,tammo.van.lessen}
@iaas.uni-stuttgart.de
http://www.iaas.uni-stuttgart.de

Abstract The *Web Service Business Process Execution Language (WS-BPEL)* is the de facto standard for describing workflow-like compositions of Web services, so-called Web service orchestrations. In this paper an ontology for executable BPEL processes is presented, which reflects both the natural language description and the syntax given in the specification. The ontology makes BPEL process models accessible at a semantic level and thus to intelligent queries and machine reasoning.

Key words: BPM, BPEL, ontology, semantics, WSML

1 Introduction

An ontology is an explicit formal specification of a domain [1]. It consists of (i) a number of concepts represented as *classes* supporting the definition of hierarchies through (multiple) inheritance, (ii) *instances* of concepts representing concrete objects of the ontology, (iii) *relations* between concepts, and (iv) *axioms* that capture knowledge that cannot be inferred. An ontology is described using a formally defined language.

This document presents an ontology for the *Web Service Business Process Execution Language 2.0 (WS-BPEL)* [2] using the formalism of the *Web Service Modeling Language (WSML)* [3].

WS-BPEL (BPEL for short) is the de facto standard for describing Web service flows in a workflow-like manner by combining activities that represent interaction with Web services (`invoke`, `receive`, `pick`, `reply`) with control flow activities (`flow`, `sequence`, `while`). BPEL [2] has been approved by the WS-BPEL Technical Committee as a Committee Specification in 01/2007. It enables the composition of Web services [4] and the modeled process is itself exposed as a Web service. Thus it provides a recursive aggregation model for Web services.

BPEL is the foundation of process execution and represents the IT view on processes in Semantic Business Process Management (SBPM) [5] which aims at bridging the gap between the business and IT view. Building an ontology of BPEL and representing BPEL processes ontologically facilitates reasoning about

In: M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, N. Stojanovic (eds.): Semantic Business Process and Product Lifecycle Management. Proceedings of the Workshop SBPM 2007, Innsbruck, April 7, 2007, CEUR Workshop Proceedings, ISSN 1613-0073, online CEUR-WS.org/Vol-251/

executable process models. Through semantic annotation of BPEL activities (e.g. relating the concept of a service invocation to a concept that represents the purpose of the invocation), queries can be formulated on the semantics of a process, rather than its syntactic representation. Ontologies of business centric process notations such as BPMN [6] and EPC [7] together with the BPEL ontology make up the orchestration part of the process space defined in [8]. Using ontology mediation queries on the business level (BPMN, EPC) can involve reasoning BPEL processes, i.e. queries can span multiple levels of abstraction. Additionally, the semantically enriched audit log of a BPEL engine in conjunction with the process models expressed in terms of the BPEL ontology can be used for semantic mining, i.e. reasoning not only on the process model level but also on the process instance level.

WSML [3] is a family of Web ontology languages with native support for the conceptual framework of the *Web Service Modeling Ontology (WSMO)* [9]. The main features of the language are support for modeling classes, attributes, binary relations and instances. Additionally, the language supports class hierarchies. WSML has multiple variants, possessing different levels of logical expressiveness and the use of different language paradigms, providing its users with the possibility to choose between the provided expressivity and the implied complexity. WSML Core forms the foundation of all the WSML variants, providing the minimum expressivity through function- and negation-free Datalog [3]. WSML Core is extended in two directions: description logics (WSML-DL) and logic programming (WSML-Flight and WSML-Rule). The variant WSML-Full unifies both extensions and therefore allows the creation of ontologies which are undecidable.

For the BPEL ontology WSML-Flight was chosen since it is the least expressive variant that allows for cardinality constraints and inequality in the logical language, which are necessary for the definition of the ontology. The ontology was defined following the basic principles of ontology development, as described in [10].

2 Building the ontology

The conceptualization of BPEL is based on the XML schema defined by the BPEL specification. The general rule we follow is to express each BPEL XML element as a class in the BPEL ontology and its corresponding attributes as attributes of the class. When possible, WSML built-in data types were reused as attributes (e.g. the strings “yes” and “no” are represented as values of type boolean rather than string). In case of conflicting names for concepts defined in the ontology and WSML keywords, the concept names are represented as full IRIs [11]. Cardinality constraints defined in the XML schema definition were adopted as far as possible. This initial conceptualization is enriched by a formal description of information described in the specification in natural-language form and not expressed in the XML schema definition. Due to this semantic enrichment of the definition of the BPEL syntax, certain aspects of the specification (e.g. the concept of fault handlers, described in Section 2.1 were modeled differently to

what their XML schema definition suggests, resulting in an abstraction from the concrete specification of the language syntax.

In the following sections, the BPEL ontology is presented by means of examples of notable design choices.

2.1 Abstraction from XSD

By following the generic approach of ontologizing the XML schema definition of BPEL, unnecessary containers which do not represent any additional semantics are introduced. The `faultHandler` element for instance is only a container that allows specifying multiple `catch` elements and one `catchAll` element to be used within a process or a scope. As the BPEL ontology captures only the semantics of the language and not syntax specific details, it abstracts from these, modeling the concepts Catch and CatchAll (which both inherit from FaultHandler) directly as attributes of the concept Process (Listing 1).

```
concept Process
    nonFunctionalProperties
        xmlns hasValue "http://docs.oasis-open.org/wsBPEL/2.0/Process/executable"
        dc#description hasValue "Concept of being a <process>-Element of an executable BPEL Process"

    endNonFunctionalProperties
    [...]
    hasCatch ofType Catch
    hasCatchAll ofType (0 1) CatchAll
    hasActivity ofType (1) Activity

concept FaultHandler
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <faultHandler>-Element"
    endNonFunctionalProperties
    hasActivity ofType (1) Activity

concept Catch subConceptOf FaultHandler
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <catch>-Element"
    endNonFunctionalProperties
    hasFaultName ofType (0 1) _sqname
    hasFaultVariable ofType (0 1) Variable
    hasFaultType ofType (0 1) DataType

concept CatchAll subConceptOf FaultHandler
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <catchAll>-Element"
    endNonFunctionalProperties
```

Listing 1. Abstraction from XML Schema: Fault Handlers

In some cases however, the XML syntax directly influences the semantics of the BPEL specification. Thus an abstraction is impossible. An example for this scenario is the extension mechanism that enables introducing new activity types (Listing 2). The `extensionActivity` element represents an activity that can be used everywhere in a process where an activity is required. However, it does not come with the standard attributes an activity has to provide, but is a container for a new activity type that has to come with the standard attributes (also depicted in Listing 2).

```
concept ExtensionActivity subConceptOf BasicActivity
   nonFunctionalProperties
        dc#description hasValue "Concept of being an ExtensionActivity, i.e. by inheriting from this activity
              type new operational semantics can be defined"
     endNonFunctionalProperties
     hasActivtity ofType (1) NewActivityType

concept standardAttributes
     nonFunctionalProperties
        dc#description hasValue "Concept of providing the standard attributes For Activities in BPEL"
     endNonFunctionalProperties
     hasName ofType (0 1) _string
     hasSuppressJoinFailure ofType (0 1) _boolean
     isTarget ofType Link
     hasJoinCondition ofType (0 1) Condition
     isSource ofType (0 *) Link
```

Listing 2. Abstraction from XML Schema: Extension activity

2.2 Introduction of hierarchies

In BPEL there are two kinds of activities, basic activities (e.g. `receive`, `reply`, `invoke` and `assign`) and structured activities (e.g. `sequence`, `flow` and `while`). While this information is part of the textual description of BPEL, it is not reflected in the XML schema definition of the language syntax. This kind of knowledge can be added by introducing additional hierarchies. Listing 3 illustrates the activity hierarchy: both BasicActivity and StructuredActivity inherit from the concept Activity. The concepts representing the BPEL elements `receive`, `reply`, `invoke` or `assign` then inherit from BasicActivity whereas the concepts representing `sequence`, `flow` or `while` inherit from the concept StructuredActivity.

```
concept Activity
     nonFunctionalProperties
        dc#description hasValue "Concept of being a BPEL Activity"
     endNonFunctionalProperties

concept BasicActivity subConceptOf Activity
     nonFunctionalProperties
        dc#description hasValue "Concept of being a basic activity"
     endNonFunctionalProperties

concept StructuredActivity subConceptOf Activity
     nonFunctionalProperties
        dc#description hasValue "Concept of being a structured activity"
     endNonFunctionalProperties

concept Assign subConceptOf {BasicActivity, StandardAttributes}
     nonFunctionalProperties
        dc#description hasValue "Concept of being an <assign>-Activity"
     endNonFunctionalProperties
     hasValidate ofType (0 1) _boolean
     hasAssignOperation impliesType (1 *) AssignOperation

concept Sequence subConceptOf {StructuredActivity, StandardAttributes}
     nonFunctionalProperties
        dc#description hasValue "Concept of being a <sequence>-Activity"
     endNonFunctionalProperties
     hasOrderedActivity ofType (1) OrderedActivity
```

Listing 3. Introduction of hierarchies: Activity hierarchy

The multiple inheritance of Assign and Sequence is further elaborated in Section 2.4.

2.3 Axioms

Axioms add semantics to an ontology because they are statements that are assumed to be true without any proof. In BPEL processes communicate with other services on a `partnerLink` which specifies which role the partner service and the process itself take. The BPEL specification informally defines, that a `partnerLink` has to specify at least one role, "myRole" or "partnerRole", which is expressed using the axioms presented in Listing 4.

```
concept _"http://www.ip-super.org/ontologies/sBPEL/20070404#PartnerLink"
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <partnerLink>-Element"
    endNonFunctionalProperties
    hasName ofType (1) _string
    hasPartnerLinkType ofType (1) wsdlx#PartnerLinkType
    hasMyRole ofType (0 1) wsdlx#Role
    hasPartnerRole ofType (0 1) wsdlx#Role
    doesInitializePartnerRole ofType (0 1) _boolean

concept WellformedPartnerlink

axiom definedBy
  ?x memberOf WellformedPartnerLink :- ?x[hasMyRole hasValue ?y] memberOf
PartnerLink.

axiom definedBy
  ?x memberOf WellformedPartnerLink :- ?x[hasPartnerRole hasValue ?y]
memberOf PartnerLink.

axiom definedBy
  !- ?x memberOf PartnerLink and naf (?x memberOf WellformedPartnerLink).
```

Listing 4. Axiom: Well-formed partner link

2.4 Multiple inheritance

A workflow comprises three dimensions: process logic (what is to be done?), organization (who is supposed to do it?), and infrastructure (using which resources?) [12]. In contrast to this, a Web service flow is characterized by only the dimensions process logic, describing the control-flow of a process, i.e. what the process does, and infrastructure, defining the services that implement the activities. Since there are activities in BPEL that deal with both dimensions (control flow and interaction), they inherit from both the interaction- and the control-flow domain. Due to the BPEL extension mechanism, the control-flow domain is split into the Activity branch and the StandardAttributes branch. The chosen design approach of multiple inheritance is depicted in Listing 5. The concept Invoke inherits from WSDLInteraction, BasicActivity and StandardAttributes.

```
concept Interaction
    nonFunctionalProperties
        dc#description hasValue "Concept of interaction"
    endNonFunctionalProperties
```

```
concept WSDLInteraction subConceptOf Interaction
    nonFunctionalProperties
        dc#description hasValue "Concept of doing WSDL dependant interaction"
    endNonFunctionalProperties
    hasPartnerLink ofType (1) _"http://www.ip-super.org/ontologies/sBPEL/20070404#PartnerLink"
    hasPortType ofType (0 1) _sqname
    hasOperation ofType (1) _sqname
    hasCorrelation ofType Correlation

concept Invoke subConceptOf {WSDLInteraction, BasicActivity, standardAttributes}
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <invoke>-Activity"
    endNonFunctionalProperties
    hasCorrelation ofType Correlation
    hasInputVariable ofType (1) Variable
    hasOutputVariable ofType (0 1) Variable
    hasCatch ofType Catch
    hasCatchAll ofType (0 1) CatchAll
    hasPattern ofType (0 1) _string
    hasCompensationHandler ofType (0 1) CompensationHandler
    hasToParts ofType ToParts
    hasFromParts ofType FromParts
```

Listing 5. Multiple Inheritance: Invoke activity

2.5 Workaround for missing expressivity in WSML

In WSML there is no construct that allows expressing the order of elements. Since the BPEL element `sequence` defines a list of activities which are to be executed sequentially and thus requires ordering of elements, the BPEL ontology implements a workaround for this issue. The approach chosen is to model a linked list, as modeling an array would require an axiom to ensure well-formedness by defining that there is only one entry for each position of the array. This is depicted in Listing 6. The activity Sequence has one attribute OrderedActivity which is a container with an attribute Activity and optionally another OrderedActivity. This workaround enables ordering elements. It does not affect the operational semantics of BPEL which are implicit.

```
concept Sequence subConceptOf {StructuredActivity, standardAttributes}
    nonFunctionalProperties
        dc#description hasValue "Concept of being a <sequence>-Activity"
    endNonFunctionalProperties
    hasOrderedActivity ofType (1) OrderedActivity

concept OrderedActivity
    nonFunctionalProperties
        dc#description hasValue "concept of being an item of an ordered Activity list"
    endNonFunctionalProperties
    hasActivity ofType (1) Activity
    hasOrderedActivity ofType (0 1) OrderedActivity
```

Listing 6. Ordered List of Activities

The same approach was used for modeling `elseif` branches which also require ordering (OrderedConditionalBranch).

3 Sample process

By means of a Virtual travel agency process (Figure 1) we show the applicability of the ontology to represent business processes. The travel booking process receives a request, prepares the input data for Web service invocations, invokes the HotelBooking and the FlightBooking Web service, aggregates the result and sends the result back to the requester. The process model in terms of instances of the BPEL ontology is in the appendix. Using this representation, simple queries like "Which activities deal with hotel booking?" can be answered. As invoke is sub concept of basicActivity which is sub concept of activity, it can be inferred that an invoke is an activity. Compared to e.g. a relational databases where a user has to explicitly encode the knowledge that invoke, receive and reply are activities within the query, this knowledge can be inferred by the reasoner when using the BPEL ontology. In conjunction with the process instance data, queries like "Which process instances deal with journeys to Germany or Austria?" can be answered. Assuming appropriate domain ontologies are used to describe the destinations it can be inferred for instance that Berlin and Stuttgart are cities in Germany and Innsbruck and Vienna are located in Austria and are thus included in the result set of the query.

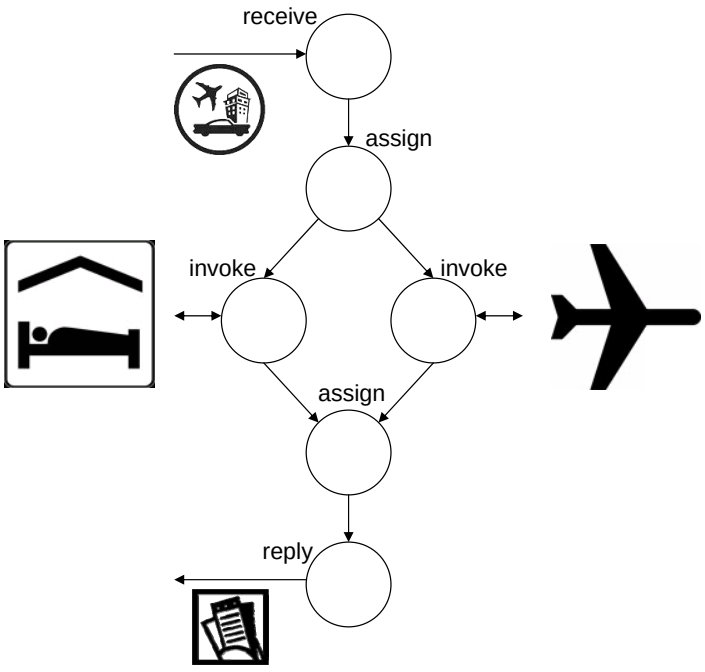

Figure 1. Virtual Travel Agency Process

4 Conclusions

In this document, an ontology of the BPEL specification was presented and the most essential design decisions were explained. Due to the chosen formalism, not all constraints given by the BPEL language can be expressed, i.e. the correctness of a BPEL process can not be checked using the ontology. However, this does not affect the application of the ontology for the purpose of process reasoning: (i) reasoning about process models, (ii) reasoning across different levels of abstraction, i.e. across modelling notations and languages on the business level and the IT level and (iii) reasoning including process instance data.

5 Acknowledgments

We would like to thank Axel Polleres for his helpful comments and explanations regarding the definition of the WSML axioms used in the ontology and Barry Norton for fruitful discussion about the subject matter.

The work published in this article was partially funded by the SUPER project[1] (contract no. FP6-026850) and the TripCom project[2] (contract no. FP6-027324) under the EU 6th Framework Programme Information Society Technologies.

References

1. Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In Guarino, N., Poli, R., eds.: Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, Kluwer Academic Publishers (1993)
2. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web services business process execution language version 2.0. Committee specification, OASIS Web Services Business Process Execution Language (WSBPEL) TC (January 2007)
3. de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., Fensel, D.: D16. 1v0. 2 The Web Service Modeling Language WSML. WSML Final Draft March **20** (2005)
4. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR Upper Saddle River, NJ, USA (2005)
5. Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. IEEE International Conference on e-Business Engineering (ICEBE 2005). Beijing, China (2005) 535–540
6. White, S.: Business Process Modeling Notation (BPMN) Version 1.0. Business Process Management Initiative, BPMI. org, May (2004)

[1] `http://www.ip-super.org/`
[2] `http://www.tripcom.org`

7. Hoffmann, W., Kirsch, J., Scheer, A.: Modellierung mit ereignisgesteuerten Prozessketten:(methodenhandbuch, Stand: Dezember 1992). Iwi (1993)
8. Hepp, M., Roman, D.: An Ontology Framework for Semantic Business Process Management. In: eOrganization: Service-, Prozess, Market- Engeneering. Volume 1., 8th international conference Wirtschaftsinformatik. Karlsruhe, Germany (2007) 423–440
9. Lausen, H., Polleres, A., Roman, D.: Web Service Modeling Ontology (WSMO). W3C Member Submission **3** (2005)
10. Noy, N., McGuinness, D.: Ontology Development 101: A Guide to Creating Your First Ontology. Knowledge Systems Laboratory, March (2001)
11. Duerst, M., Suignard, M.: Internationalized Resource Identifiers (IRIs). (2005)
12. Leymann, F., Roller, D.: Production workflow. Prentice Hall (2000)

Appendix

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _"http://www.ip-super.org/ontologies/sBPEL/TravelBookingProcess#",
  sbpel _"http://www.ip-super.org/ontologies/sBPEL/20070404#",
  dc _"http://purl.org/dc/elements/1.1#" }

ontology sbpelProcess

    nonFunctionalProperties
        dc#title hasValue "Travel Booking sBPEL Process"
        dc#creator hasValue "Joerg Nitzsche"
        dc#publisher hasValue "SUPER European Integrated Project"
        dc#subject hasValue {"sBPEL", "business process", "workflow"}
        dc#language hasValue "en-UK"
        dc#date hasValue "$Date: 2007/04/04$"
    endNonFunctionalProperties

    importsOntology
        { _"http://www.ip-super.org/ontologies/sBPEL/20070404#",
        _"http://www.ip-super.org/ontologies/wsdlExtension4BPEL/20070126#" }

instance processTravelBooking memberOf Process
  hasName hasValue "ContentProvision"
  hasTargetNamespace hasValue "http://www.ip-super.org/ontologies/prereview"
  hasPartnerLink hasValue travelBookingPL, hotelBookingPL, flightBookingPL}
  hasActivity hasValue processFlow
  hasVariable hasValue {varTravelBookingRequest, varHotelBookingRequest, varFlightBookingRequest,
        varTravelBookingResponse, varHotelBookingResponse, varFlightBookingResponse}

instance typeTravelBookingRequest memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/TravelAgency.wsdl#
        travelBookingRequestMessage"

instance varTravelBookingRequest memberOf Variable
  hasName hasValue "varTravelBookingRequest"
  hasDataType hasValue typeTravelBookingRequest

instance typeTravelBookingResponse memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/TravelAgency.wsdl#
        travelBookingResponseMessage"

instance varTravelBookingResponset memberOf Variable
  hasName hasValue "varTravelBookingResponse"
  hasDataType hasValue typeTravelBookingResponse

instance typeHotelBookingRequest memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/HotelService.wsdl#
```

```
        hotelBookingRequestMessage"

instance varHotelBookingRequest memberOf Variable
  hasName hasValue "varHotelBookingRequest"
  hasDataType hasValue typeHotelBookingRequest

instance typeHotelBookingResponse memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/HotelService.wsdl#
        hotelBookingResponseMessage"

instance varHotelBookingResponset memberOf Variable
  hasName hasValue "varHotelBookingResponse"
  hasDataType hasValue typeHotelBookingResponse

instance typeFlightBookingRequest memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/FlightService.wsdl#
        flightBookingRequestMessage"

instance varFlightBookingRequest memberOf Variable
  hasName hasValue "varFlightBookingRequest"
  hasDataType hasValue typeFlightBookingRequest

instance typeFlightBookingResponse memberOf MessageType
  hasMessageType hasValue "http://ip-super.org/processes/FlightService.wsdl#
        flightBookingResponseMessage"

instance varFlightBookingResponset memberOf Variable
  hasName hasValue "varFlightBookingResponse"
  hasDataType hasValue typeFlightBookingResponse

instance travelBookingPL memberOf _"http://www.ip-super.org/ontologies/sBPEL/20070308#PartnerLink"
  hasName hasValue "travelBookingPL"
  hasPartnerLinkType hasValue travelBookingPLT
  hasMyRole hasValue provider

instance travelBookingPLT memberOf PartnerLinkType
  hasName hasValue "travelBookingPLT"
  hasRole hasValue provider

instance provider memberOf Role
  hasName hasValue "provider"
  hasPortType hasValue "http://ip-super.org/processes/TravelAgency.wsdl#TravelBookingPortType"

instance hotelBookingPL memberOf _"http://www.ip-super.org/ontologies/sBPEL/20070308#PartnerLink"
  hasName hasValue "hotelBookingPL"
  hasPartnerLinkType hasValue hotelBookingPLT
  hasPartnerRole hasValue provider

instance hotelBookingPLT memberOf PartnerLinkType
  hasName hasValue "hotelBookingPLT"
  hasRole hasValue provider

instance provider memberOf Role
  hasName hasValue "provider"
  hasPortType hasValue "http://ip-super.org/processes/HotelService.wsdl#HotelBookingPortType"

instance flightBookingPL memberOf _"http://www.ip-super.org/ontologies/sBPEL/20070308#PartnerLink"
  hasName hasValue "flightBookingPL"
  hasPartnerLinkType hasValue flightBookingPLT
  hasPartnerRole hasValue provider

instance flightBookingPLT memberOf PartnerLinkType
  hasName hasValue "flightBookingPLT"
  hasRole hasValue provider

instance provider memberOf Role
  hasName hasValue "provider"
  hasPortType hasValue "http://ip-super.org/processes/FightService.wsdl#FlightBookingPortType"
```

```
instance processFlow memberOf Flow
  hasActivity hasValue {recTravelBookingRequest, assSplitRequest, invHotelService, invFlightService,
      assMergeResults, repTravelBookingResponse}
  hasLink hasValue {receiveToSplitRequest, splitRequestToInvHotelService, splitRequestToInvFlightService,
      invHotelServiceToMergeResults, invFlightIServiceToMergeResults, mergeResultsToReply}

instance receiveToSplitRequest memberOf Link
  hasName hasValue "receiveToSplitRequest"

instance splitRequestToInvHotelService memberOf Link
  hasName hasValue "splitRequestToInvHotelService"

instance splitRequestToInvFlightService memberOf Link
  hasName hasValue "splitRequestToInvFlightService"

instance invHotelServiceToMergeResults memberOf Link
  hasName hasValue "invHotelServiceToMergeResults"

instance invFlightIServiceToMergeResults memberOf Link
  hasName hasValue "invFlightIServiceToMergeResults"

instance mergeResultsToReply memberOf Link
  hasName hasValue "mergeResultsToReply"

instance recTravelBookingRequest memberOf Receive
  hasName hasValue "recTravelBookingRequest"
  hasCreateInstance hasValue true
  hasVariable hasValue varTravelBookingRequest
  hasPartnerLink hasValue travelBookingPL
  hasOperation hasValue "bookFlight"
  isSource hasValue receiveToSplitRequest

instance assSplitRequest memberOf Assign
  hasName hasValue "assSplitRequest"
  hasAssignOperation hasValue {copyHotelRequest, copyFlightRequest}
  isTarget hasValue receiveToSplitRequest
  isSource hasValue {splitRequestToInvHotelService, splitRequestToInvFlightService}

instance copyHotelRequest memberOf Copy
  hasFromSpecification hasValue fromTravelRequestHotel
  hasToSpecification hasValue toHotelRequest

instance fromTravelRequestHotel memberOf CopyVariablePart
  hasVariable hasValue varTravelBookingRequest
  hasPart hasValue "http://ip-super.org/processes/TravelAgency.wsdl#wsdl11.messagePart(
      travelBookingRequestMessage/Hotel)"

instance toHotelRequest memberOf CopyVariablePart
  hasVariable hasValue varHotelBookingRequest
  hasPart hasValue "http://ip-super.org/processes/HotelService.wsdl#wsdl11.messagePart(
      hotelBookingRequestMessage/Hotel)"

instance copyFlightRequest memberOf Copy
  hasFromSpecification hasValue fromTravelRequestFlight
  hasToSpecification hasValue toHotelRequest

instance fromTravelRequestFlight memberOf CopyVariablePart
  hasVariable hasValue varTravelBookingRequest
  hasPart hasValue "http://ip-super.org/processes/TravelAgency.wsdl#wsdl11.messagePart(
      travelBookingRequestMessage/Flight)"

instance toHotelRequest memberOf CopyVariablePart
  hasVariable hasValue varFlightBookingRequest
  hasPart hasValue "http://ip-super.org/processes/FlightService.wsdl#wsdl11.messagePart(
      flightBookingRequestMessage/Flight)"

instance invHotelService memberOf Invoke
```

```
  hasName hasValue "invHotelService"
  hasVariable hasValue varHotelBookingRequest
  hasPartnerLink hasValue hotelBookingPL
  hasOperation hasValue "bookHotel"
  isTarget hasValue splitRequestToInvHotelService
  isSource hasValue invHotelServiceToMergeResults

instance invFlightService memberOf Invoke
  hasName hasValue "invFlightService"
  hasVariable hasValue varFlightBookingRequest
  hasPartnerLink hasValue flightBookingPL
  hasOperation hasValue "bookFlight"
  isTarget hasValue splitRequestToInvFlightService
  isSource hasValue invFlightServiceToMergeResults

instance assMergeResults memberOf Assign
  hasName hasValue "assMergeResults"
  hasAssignOperation hasValue {copyHotelResponse, copyFlightResponse}
  isTarget hasValue {invHotelServiceToMergeResults, invFlightIServiceToMergeResults}
  isSource hasValue mergeResultsToReply

instance copyHotelResponse memberOf Copy
  hasFromSpecification hasValue fromHotelResponse
  hasToSpecification hasValue toTravelResponseHotel

instance fromHotelResponse memberOf CopyVariablePart
  hasVariable hasValue varHotelBookingResponse
  hasPart hasValue "http://ip-super.org/processes/HotelService.wsdl#wsdl11.messagePart(
        hotelBookingResponseMessage/Hotel)"

instance toTravelResponseHotel memberOf CopyVariablePart
  hasVariable hasValue varTravelBookingResponse
  hasPart hasValue "http://ip-super.org/processes/TravelAgency.wsdl#wsdl11.messagePart(
        travelBookingResponseMessage/Hotel)"

instance copyFlightResponse memberOf Copy
  hasFromSpecification hasValue fromFlightResponse
  hasToSpecification hasValue toTravelResponseFlight

instance fromFlightResponse memberOf CopyVariablePart
  hasVariable hasValue varFlightBookingResponse
  hasPart hasValue "http://ip-super.org/processes/FlightService.wsdl#wsdl11.messagePart(
        flightBookingResponseMessage/Flight)"

instance toTravelResponseFlight memberOf CopyVariablePart
  hasVariable hasValue varTravelBookingResponse
  hasPart hasValue "http://ip-super.org/processes/TravelAgency.wsdl#wsdl11.messagePart(
        travelBookingResponseMessage/Flight)"

instance repTravelBookingResponse memberOf Reply
  hasName hasValue "repTravelBookingResponse"
  hasVariable hasValue varTravelBookingResponse
  hasPartnerLink hasValue travelBookingPL
  hasOperation hasValue "bookFlight"
  isTarget hasValue mergeResultsToReply
```

Listing 7. Virtual Travel Agency Process