



D14v0.1. Ontology-based Choreography and Orchestration of WSMO Services

WSMO Working Draft 28 February 2005

This version:

<http://www.wsmo.org/TR/d14/v0.1/20050228/>

Latest version:

<http://www.wsmo.org/TR/d14/v0.1/>

Previous version:

<http://www.wsmo.org/TR/d14/v0.1/20050211/>

Editors:

Dumitru Roman
James Scicluna
Cristina Feier

Co-Authors:

Michael Stollberg
Dieter Fensel

This document is also available in non-normative [PDF](#) version.

Copyright © 2005 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

Table of contents

- [1. Introduction](#)
 - [2. WSMO Choreography](#)
 - [2.1 State](#)
 - [2.2 Guarded transitions](#)
 - [3. WSMO Orchestration](#)
 - [4. Choreography and Orchestration description example](#)
 - [5. Conclusions and further work](#)
 - [References](#)
 - [Acknowledgement](#)
-

1. Introduction

Choreography and Orchestration are part of the interface definition of a WSMO service description [[Roman et al., 2004](#)]. They describe the *behavior* of the service

from two perspectives: *communication* (how to communicate with the service such that the service will provide its capability), and respectively *collaboration* (how the service collaborates with other WSMO services to achieve its capability).

Choreography describes the behavior of the service from a user point of view; this definition is in accordance to the following definition given by W3C Glossary [[W3C Glossary, 2004](#)]: *Web Services Choreography concerns the interactions of services with their users. Any user of a Web service, automated or otherwise, is a client of that service. These users may, in turn, be other Web Services, applications or human beings.*

The orchestration of a WSMO service defines how the overall functionality of the service is achieved by the cooperation of other WSMO service providers. It describes how the service works from the provider's perspective (i.e. how a service makes use of other WSMO services or goals in order to achieve its capability). This complies with the W3C definition of Web Service Orchestrations [[W3C Working Group](#)]: *An orchestration defines the sequence and conditions in which one Web Service invokes other Web Services in order to realize some useful function. That is, an orchestration is the pattern of interactions that a Web Service agent must follow in order to achieve its goal.*

The aim of this document is to provide a core conceptual model for describing choreographies and orchestrations in WSMO. The state-based mechanism for describing WSMO choreographies and orchestrations is based on the Abstract State Machines [[Gurevich, 1995](#)] methodology. ASMs have been chosen as underlying model of choreography and orchestration for the following three reasons:

- *Minimality*: ASMs provide a minimal set of modeling primitives, i.e., enforce minimal ontological commitments. Therefore, they do not introduce any ad hoc elements that would be questionable to be included into a standard proposal.
- *Maximality*: ASMs are expressive enough to model any aspect around computation.
- *Formality*: ASMs provide a rigid mathematical framework to express dynamics.

For a detailed explanation on ASMs we refer the reader to [[Börger, 1998](#)].

The rest of the document is organized as follows: [Section 2](#) provides a core conceptual model for WSMO choreographies, [Section 3](#) presents the conceptual model for WSMO orchestrations by pointing out the differences to the core conceptual model for WSMO choreographies, [Section 4](#) gives an example for choreography and orchestration, and [Section 5](#) provides conclusions and points to future work with respect to WSMO choreography and orchestration.

2. WSMO Choreography

As the WSMO Choreography is based on ASMs, it inherits the core principles of ASMs, which summarized, are: (1) it is **state-based**, (2) it represents a **state by an algebra**, and (3) it models state changes by **guarded transition rules** that change the values of functions and relations defined by the signature of the algebra.

Taking the ASMs methodology as a starting point, a WSMO choreography is state-based and consists of two elements which are defined as follows:

```

Class wsmoChoreography
  hasState type ontology
  hasGuardedTransitions type guardedTransitions

```

State

A state is described by an ontology as defined in [\[Roman et. al., 2004\] Section 4.](#)

Guarded transitions

Transition rules that express changes of states by changing the set of instances.

The rest of this section is organized as follows: [Section 2.1](#) describes what the `state` is in more details, and [Section 2.2](#) describes the `guarded transitions`.

2.1 State

In extension to a "normal" WSMO ontology, an Ontology that is used to describe states in a WSMO choreography introduces a new non-functional property. When a concept, relation or function in a choreography is defined, the attribute *mode* can be defined as a new non functional property. It can take one of the following values:

- *static* - meaning that the extension of the concept, relation, or function cannot be changed. If not explicitly defined, the attribute *mode* takes this value by default.
- *controlled* - meaning that the extension of the concept, relation, or function can only be changed by the service.
- *in* - meaning that the extension of the concept, relation, or function can only be changed by the environment. A **grounding** mechanism for this item must be provided that implements *write* access for the environment.
- *shared* - meaning that the extension of the concept, relation, or function can be changed by the service and the environment. A **grounding** mechanism for this item must be provided that implements *read/write* access for the environment.
- *out* - meaning that the extension of the concept, relation, or function can only be changed by the service. A **grounding** mechanism for this item must be provided that implements *read* access for the environment.

For more details on WSMO Grounding we refer the reader to [\[Kopecky et. al., 2005\]](#).

The signature of the states of a WSMO choreography is defined by all legal WSMO identifiers, concepts, relations, functions, and axioms. This signature is the same for all states. The elements that can change and that are used to express different states of a choreography, are the instances (and their attribute values) of concepts, functions, and relations that are not defined as being *static*. In conclusion, a specific state is described by a set of explicitly defined instances and values of their attributes or through a link to an instance store.

2.2 Guarded transitions

Guarded Transitions are used to express changes of states by means of rules, expressible in the following form:

if *Cond* **then** *Updates*.

Cond is an arbitrary WSML axiom, formulated in the given signature of the state. The *Updates* consist of arbitrary WSMO Ontology instance (see [Section 4.7 of WSMO 1.1](#)) statements.

3. WSMO Orchestration

Identically to a WSMO choreography, a WSMO Orchestration is state-based and consists of the same elements as a WSMO Choreography (see Listing 1). In extension, a guarded transition can also take the following form:

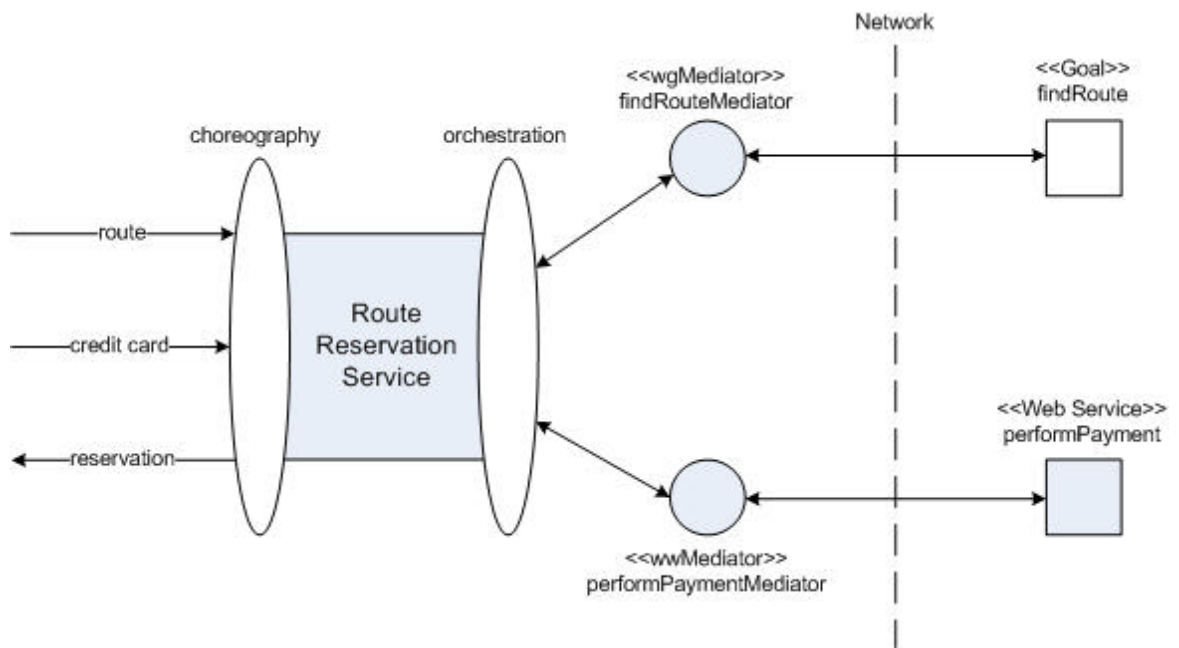
if <condition> **then** <mediator ID>

The difference with respect to choreography is that of using mediators to link the orchestration to other goals or web services. The mediators used are of type *wwMediator* or *wgMediator*. If the required service is already known, a *wwMediator* is used to link the orchestration to the choreography of the required service. If the required service is still unknown, then a *wgMediator* is used to link to the goal which expresses what is needed by the orchestration at the given state.

4. Choreography and Orchestration description example

Our service makes reservations for trips, for which the starting and ending points are located in Austria or Germany. The information requested by the service from the user is route and credit card information. In case it can provide the requested service it will perform the booking for the client. This scenario is depicted in Figure 1 below.

Figure 1. Route Reservation Service accepts a route and a credit card and returns a reservation to the client



[Listing 2](#) describes the Trip Reservation Ontology, containing concepts, relations and functions needed for making a trip reservation. Another ontology used for specifying the choreography of the above mentioned service is the Purchase Ontology. [Listing 3](#) describes this ontology.

```

namespace {<"http://www.wsmo.org/ontologies/tripReservationOntology">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
  tc <"http://www.wsmo.org/ontologies/trainConnection#">,
  prs <"http://www.wsmo.org/mediators/owlPersonMediator.wsml#">
} ontology <"http://www.wsmo.org/ontologies/tripReservationOntology#">
nonFunctionalProperties
  dc#title hasValue "Trip Reservation Ontology"
  dc#creator hasValue "DERI Innsbruck"
  dc#description hasValue "an ontology for describing trip
    reservation related knowledge"
  dc#publisher hasValue "DERI International"
  dc#contributor hasValues "Titi"
  dc#date hasValue "2004-10-22"
  dc#type hasValue <"http://www.wsmo.org/2004/d2/#ontology">
  dc#format hasValue "text/html"
  dc#language hasValue "en-us"
  dc#rights hasValue <"http://deri.at/privacy.html">
  version hasValue "$Revision 1.17 $"
endNonFunctionalProperties

importsOntology <"http://www.wsmo.org/ontologies/trainConnection#">
usesMediator
  <"http://www.wsmo.org/mediators/owlPersonMediator.wsml">

concept route
  nonFunctionalProperties
    dc#description hasValue "concept of a route between two stations"
  endNonFunctionalProperties
  startLocation impliesType tc#station
  endLocation impliesType tc#station

concept reservation
  nonFunctionalProperties
    dc#description hasValue "concept of reservation, containing
      a reservation holder"
  endNonFunctionalProperties
  reservationNumber ofType xsd#integer
  reservedRoute impliesType route
  reservationHolder impliesType prs#person

relation connectionExists/2
  nonFunctionalProperties
    dc:description hasValue "connection existence relationship
      between two stations"
  endNonFunctionalProperties

```

Listing 3. Purchase Ontology.

```

namespace {<"http://www.wsmo.org/ontologies/purchaseOntology#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
  prs <"http://www.wsmo.org/mediators/owlPersonMediator.wsml#">
} ontology <"http://www.wsmo.org/ontologies/purchaseOntology#">
nonFunctionalProperties
  dc#title hasValue "Purchase Ontology"
  dc#creator hasValue "DERI Innsbruck"
  dc#description hasValue "an ontology for describing purchase
    related knowledge"
  dc#publisher hasValue "DERI International"
  dc#contributor hasValues "Titi"
  dc#date hasValue "2004-10-22"
  dc#type hasValue <"http://www.wsmo.org/2004/d2/#ontology">
  dc#format hasValue "text/html"
  dc#language hasValue "en-us"
  dc#rights hasValue <"http://deri.at/privacy.html">
  version hasValue "$Revision 1.17 $"
endNonFunctionalProperties

usesMediator
  <"http://www.wsmo.org/mediators/owlPersonMediator.wsml">

concept creditCard
  nonFunctionalProperties
    dc#description hasValue "concept of credit card, containing an
      owner, a balance and an expiry date"
  endNonFunctionalProperties
  owner impliesType prs#person
  balance ofType xsd#integer
  expiryDate ofType xsd#dateTime

relation validCreditCard/2
  nonFunctionalProperties
    dc#description hasValue "Function that checks whether a credit card
      is valid or not"
    dc#relation hasValue {ValidCreditCardDef,
      FunctionalDependenceValidCreditCard}
  endNonFunctionalProperties

axiom ValidCreditCardDef
  definedBy
    forall ?x, ?y (
      validCreditCard(?x, ?y) equivalent
      ?x memberOf creditCard and
      ?y = true impliedBy
        neg(wsml#date\ -less\ -than(?x.expiryDate, wsml#current\ -date()))
      or
      ?y = false impliedBy
        wsml#date\ -less\ -than(?x.expiryDate, wsml#current\ -date())
    ).

axiom FunctionalDependencyValidCreditCard
  definedBy
    forall ?x,?y1,?y2 (
      false impliedBy ValidCreditCard(?x,?y1) and
      (ValidCreditCard(?x,?y2) and ?y1 != ?y2) .

```

As explained in [Section 2.1](#), the non-functional properties of some of the concepts, relations and functions used in the choreography of a service are extended with the attribute *mode*. For those elements that have the mode *in*, *out* or *shared*, a grounding mechanism must be provided. Such a mechanism is attached to an element by introducing an extra attribute *grounding* of the associated non-functional properties whose value refers to it. For enriching elements defined in external ontologies with these two non-functional properties, the ontology that describes the state of the service imports the relevant ontologies and redefines these elements by inheriting their definition and defining values for the two additional non-functional properties (or just one in the case of entities with the mode *static* or *controlled*). In our example, the relevant ontologies for the choreography are those described above, namely the Trip Reservation Ontology and the Purchase Ontology. They are imported by the ontology that describes the state of this choreography, namely the Trip Reservation Choreography Ontology. The content of this new ontology is provided in [Listing 4](#).

Listing 4. Trip Reservation Choreography Ontology.

```

namespace
  {<"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
  tr <"http://www.wsmo.org/ontologies/tripReservationOntology#">,
  po <"http://www.wsmo.org/ontologies/purchaseOntology#">
} ontology <"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">
nonFunctionalProperties
  dc#title hasValue " Trip Reservation Choreography Ontology"
  dc#creator hasValue "DERI Innsbruck"
  dc#description hasValue "an ontology for that redefines concepts
    from other ontologies in order to use them in the choreography
    by defining for those two additional non-functional
    properties: mode and grounding"
  dc#publisher hasValue "DERI International"
endNonFunctionalProperties

importsOntology {<"http://www.wsmo.org/ontologies/tripReservationOntology#">,
  <"http://www.wsmo.org/ontologies/purchaseOntology#">
}

concept route subConceptOf tr#route
  nonFunctionalProperties
    mode hasValue in
    grounding hasValue reservationWSDL#reserveRoute
  endNonFunctionalProperties

concept reservation subConceptOf tr#reservation
  nonFunctionalProperties
    mode hasValue out
    grounding hasValue reservationWSDL#reserveRoute
  endNonFunctionalProperties

concept creditCard subConceptOf po#creditCard
  nonFunctionalProperties
    mode hasValue in
    grounding hasValue reservationWSDL#reserveRoute
  endNonFunctionalProperties

```

[Listing 5](#) contains the definition of the Trip Reservation Service. The capability offered by it, Reservation Service Capability, is presented in [Listing 6](#), and its

choreography, Trip Reservation Service Choreography, is described in [Listing 7](#). Note that Listing 6 and Listing 7 are assumed to be in the same namespace (i.e. the Trip Reservation Service namespace).

Listing 5. Trip Reservation Service definition.

```

namespace {<"http://www.wsmo.org/ontologies/tripReservationService#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
  tr <"http://www.wsmo.org/ontologies/tripReservationOntology#"> ,
  tc <"http://www.wsmo.org/ontologies/trainConnection#">,
  po <"http://www.wsmo.org/ontologies/purchaseOntology#">,
  loc <"http://www.wsmo.org/ontologies/location#"> ,
  trc <"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">}

webService <"http://www.wsmo.org/ontologies/tripReservationService.wsml">
  nonFunctionalProperties
    dc#title hasValue "Trip Reservation Service"
    dc#creator hasValue "DERI Innsbruck"
    dc#description hasValue "service for online trip reservations for Austria
      and Germany"
    dc#publisher hasValue "DERI International"
    dc#contributor hasValues "Titi"
    dc#date hasValue "2004-10-22"
    dc#type hasValue <"http://www.wsmo.org/2004/d2/#service">
    dc#format hasValue "text/html"
    dc#language hasValue "en-us"
    dc#coverage hasValues {tc#austria, tc#germany}
    dc#rights hasValue <"http://deri.at/privacy.html">
    version hasValue "$Revision 1.17 $"
  endNonFunctionalProperties

  importsOntology {
    <"http://www.wsmo.org/ontologies/location#">
  }

  usesMediator
    <"http://www.wsmo.org/mediators/owlPersonMediator.wsml">

  capability reservationServiceCapability

  interface reservationServiceInterface
    choreography reservationServiceChoreography

```

Below, [Listing 6](#) presents the capability of the service by defining its precondition and postcondition. The precondition expresses the fact that in order to be executed, the service requires a route, for which the start and the end locations have to be in Austria or in Germany, and the existence of a connection between the given start and end locations. The assumption is that a valid credit card is provided by the client. The postcondition expresses the fact that, in case of a successful execution of a service, a reservation is created. Finally, the effect specifies that the credit card is charged with the cost of the reserved route.

capability reservationServiceCapability

nonFunctionalProperties

dc:title **hasValue** "Trip Reservation Service Capability"
 dc:creator **hasValue** "DERI Innsbruck"
 dc:description **hasValue** "description of the capability of the reservation
 service of providing trip reservations for Austria and Germany"
 dc:publisher **hasValue** "DERI International"
 dc:contributor **hasValues** "Titi"
 dc:date **hasValue** "2004-10-22"
 dc:format **hasValue** "text/html"
 dc:language **hasValue** "en-us"
 dc:rights **hasValue** <"http://deri.at/privacy.html">
 version **hasValue** "\$Revision 1.17 \$"

endNonFunctionalProperties

sharedVariables ?route, ?creditCard, ?cardOwner

assumption

nonFunctionalProperties

dc:description **hasValue** "the given credit card must be valid"

endNonFunctionalProperties

definedBy

(?creditCard **memberOf** po#creditCard[
 owner **hasValue** ?cardOwner
]) **and**
 (po#validCreditCard(?creditCard, true))

precondition

nonFunctionalProperties

dc:description **hasValue** "a connection must exist for the given route"

endNonFunctionalProperties

definedBy

?route **memberOf** tr#route[
 startLocation **hasValue** ?start,
 endLocation **hasValue** ?end
] **and**
 (?start[
 tc#locatedIn **hasValue** loc#austria]
or
 ?start[
 tc#locatedIn **hasValue** loc#germany])
and
 (?end[
 tc#locatedIn **hasValue** loc#austria]
or
 ?end[
 tc#locatedIn **hasValue** loc#germany
]) **and**
 (tr#connectionExists(?start,?end))

postcondition

nonFunctionalProperties

dc:description **hasValue** "a reservation for the given route is created"

endNonFunctionalProperties

definedBy

?reservation **memberOf** tr#reservation[
 reservationHolder **hasValue** ?cardOwner
 route **hasValue** ?route

```

]
effect
  nonFunctionalProperties
    dc:description hasValue "credit card must be finally charged"
  endNonFunctionalProperties
  definedBy
    ?reservation memberOf tr#reservation[
      price hasValue ?reservationCost
    ] and
    ?creditCard[
      balance hasValue@pre ?initialBalance
    ] and
    ?creditCard[
      balance hasValue (?initialBalance-?reservationCost)
    ].

```

For defining the choreography of the service, we also need to define the guarded transitions ([Listing 7](#)). Notice that the current state is described in the condition part of the rules and the state changes are modelled in the *then* part. As stated, these changes are modelled by changes in the attribute values of the instances.

Listing 7. Guarded Transitions in the Choreography of the Trip Reservation Service.

```

choreography reservationServiceChoreography

state <"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">

guardedTransitions reservationServiceTransitionRules

/*rule 1: when a routeInstance is received, if it fulfils the condition regarding
the start and the end destination, a partial reservation is created
( no reservation holder, for now) */

if (routeInstance memberOf trc#route[
  startLocation hasValue ?start,
  endLocation hasValue ?end
] and
(?start[
  tc#locatedIn hasValue loc#austria]
or
?start[
  tc#locatedIn hasValue loc#germany])
and
(?end[
  tc#locatedIn hasValue loc#austria]
or
?end[
  tc#locatedIn hasValue loc#germany
]) and
(tr#connectionExists (?start,?end))
then
(reservationInstance memberOf trc#reservation[
  reservedRoute hasValue routeInstance
])

/*rule 2: if a partial reservation exists and if a credit card information
is received that points to a valid credit card, the partial reservation
is completed with the name of the credit card owner*/

if ((reservationInstance memberOf trc#reservation) and
(creditCardInstance memberOf trc#creditCard[
  owner hasValue ?cardOwner
]) and
po#validCreditCard(creditCardInstance, true))
then
reservationInstance[
  reservationHolder hasValue ?cardOwner
])

```

[Table 1](#) provides a trace of the choreography states for the choreography defined above in the case of a particular interaction with a client. This interaction is such that the client provides a route as expected by the service in order to be able to provide a reservation (the origin is InnsbruckHbf, in Austria, and the destination is MünchenOstHbf, in Germany) and a valid credit card (its expiration date is in the future). As said before each state is characterized by an ontology, but a set of instances and their attribute values makes the distinction between states. Thus, the table refers to a state as being the set of dynamic instances and leaves apart the state signature:

Table 1. Choreography states trace for a particular interaction with the client.

State Informal Description	State Formal Description
The initial state	\emptyset
State in which a route instance has been received	<pre>{routeInstance memberOf trc#route[startLocation hasValue tc#InnsbruckHbf, endLocation hasValue tc#MünchenOstHbf]} </pre>
The state after the execution of the first transition rule: a partial reservation has been created	<pre>{routeInstance memberOf trc#route[startLocation hasValue tc#InnsbruckHbf, endLocation hasValue tc#MünchenOstHbf], reservationInstance memberOf trc#reservation[reservedRoute hasValue routeInstance]} </pre>
The state where a partial reservation exists and a credit card information is received	<pre>{routeInstance memberOf trc#route[startLocation hasValue tc#InnsbruckHbf, endLocation hasValue tc#MünchenOstHbf], reservationInstance memberOf trc#reservation[reservedRoute hasValue routeInstance], creditCardInstance memberOf trc#creditCard[owner hasValue prs#John, expiryDate hasValue "2006-10-10"]} </pre>
The state after the execution of the second transition rule: the reservation is completed with the reservation holder being instantiated as the credit card owner	<pre>{routeInstance memberOf trc#route[start hasValue loc#InnsbruckHbf, end hasValue loc#MünchenOstHbf], reservationInstance memberOf trc#reservation[reservedRoute hasValue routeInstance, reservationHolder hasValue prs#John], creditCardInstance memberOf trc#CreditCard[owner hasValue prs#John, expiryDate hasValue "2006-10-10"]} </pre>

Listing 8 defines the guarded transitions for the Route Reservation Orchestration. The first rule fires when a route instance is available and uses the *findRouteMediator* to find the required route. This mediator links to a Goal ([Listing 9](#)) for which a Web Service is to be discovered. The second rule fires when both the route and credit card instances are available and passes the latter to the *performPaymentMediator* to perform the purchase transaction. Note that although in this rule the state is described by both the route and credit card instances, only the latter is passed to the mediator.

Listing 8. Guarded Transitions in the Orchestration of the Trip Reservation Service.

```
orchestration reservationServiceOrchestration

guardedTransitions reservationServiceTransitionRules

/*rule 1: when a routeInstance is received, the service must make
use of another service which provides the route*/

if (routeInstance memberOf tr#route[
  startLocation hasValue ?start,
  endLocation hasValue ?end
] and
(?start[tc#locatedIn hasValue loc#austria]
 or
 ?start[tc#locatedIn hasValue loc#germany])
and
(?end[tc#locatedIn hasValue loc#austria]
 or
 ?end[tc#locatedIn hasValue loc#germany
]) and

(tr#connectionExists (?start,?end))
then
  findRouteMediator

/*rule 2: if a route is available, then the credit card is charged
via another service*/

if ((creditCardInstance memberOf po#creditCard[
  owner hasValue ?cardOwner
]) and
(routeInstance memberOf tr#route[
  startLocation hasValue ?start,
  endLocation hasValue ?end
] and
(?start[tc#locatedIn hasValue loc#austria]
 or
 ?start[tc#locatedIn hasValue loc#germany])
and
(?end[tc#locatedIn hasValue loc#austria]
 or
 ?end[tc#locatedIn hasValue loc#germany
]) and
(tr#connectionExists (?start,?end)) and
po#validCreditCard(creditCardInstance, true))
then
  performPaymentMediator
```

Listing 9 below presents the WG Mediator which links the Trip Reservation Service to the goal that, if fulfilled, would provide the required route.

Listing 9. WG Mediator linking the Orchestration of the Trip Reservation Service to the Find Route Goal

```
namespace {<"http://www.wsmo.org/mediators/findRouteMediator#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
}

wgMediator _"http://www.wsmo.org/mediators/findRouteMediator"
nonFunctionalProperties
  dc#title hasValue "Trip Reservation Service to Find Route Goal
  linker"
  dc#creator hasValue _"http://www.deri.org/foaf#deri"
  dc#description hasValue "WG Mediator linking Trip Reservation
  Service to a goal to Find a Train Route"
  dc#publisher hasValue _"http://www.deri.org/foaf#deri"
  dc#contributor hasValue _"http://www.deri.org/foaf#james"
  dc#date hasValue "2005-02-28"
  dc#type hasValue _"http://www.wsmo.org/2004/d2/#wgMediator"
  dc#language hasValue "en-us"
  dc#relation hasValue {_"http://www.wsmo.org/ontologies/tripReservationService",
    _"http://www.wsmo.org/ontologies/findRouteGoal"}
  dc#rights hasValue _"http://www.deri.org/privacy.html"
  version hasValue "$Revision: 1.11 $"
endNonFunctionalProperties

source _"http://www.wsmo.org/ontologies/tripReservationService"

target _"http://www.wsmo.org/ontologies/findRouteGoal"

useService _"http://www.wsmo.org/mediationServices/Trip2FindRoute"
```

Listing 10 below presents the WG Mediator which links the Trip Reservation Service to the Perform Payment Service.

Listing 10. WW Mediator linking the Orchestration of the Trip Reservation Service to the Perform Payment Service

```
namespace {<"http://www.wsmo.org/mediators/performPaymentMediator#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
}

wwMediator _"http://www.wsmo.org/mediators/performPaymentMediator"
nonFunctionalProperties
  dc#title hasValue "Trip Reservation Service to Perform Payment Service linker"
  dc#creator hasValue _"http://www.deri.org/foaf#deri"
  dc#description hasValue "WW Mediator linking Trip Reservation Service to a
    service performing a payment transaction"
  dc#publisher hasValue _"http://www.deri.org/foaf#deri"
  dc#contributor hasValue _"http://www.deri.org/foaf#james"
  dc#date hasValue "2005-02-28"
  dc#type hasValue _"http://www.wsmo.org/2004/d2/#wwMediator"
  dc#language hasValue "en-us"
  dc#relation hasValue {_ "http://www.wsmo.org/ontologies/tripReservationService",
    _"http://www.wsmo.org/ontologies/performPaymentService"}
  dc#rights hasValue _"http://www.deri.org/privacy.html"
  version hasValue "$Revision: 1.11 $"
endNonFunctionalProperties

source _"http://www.wsmo.org/ontologies/tripReservationService"

target _"http://www.wsmo.org/ontologies/performPaymentService"

useService _"http://www.wsmo.org/mediationServices/Trip2PerformPayment"
```

Listing 11 below defines the findRouteGoal which is linked by the findRouteMediator in [Listing 9](#).

Listing 11. Goal description which is invoked in order to find the required route from another service.

```
namespace <http://www.wsmo.org/ontologies/findRouteGoal#>
  dc <http://purl.org/dc/elements/1.1#>

  po <http://www.wsmo.org/ontologies/purchase#>
  prs <http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owl/PersonMediator.wsml>
  targetnamespace: <http://www.wsmo.org/ontologies/findRouteGoal#>

goal findRoute

  nonFunctionalProperties
    dc:title hasValue "Find Route"
    dc:creator hasValue "DERI Innsbruck"
    dc:description hasValue "A goal that requires a train route between Germany and Austria"
    dc:publisher hasValue "DERI International"
    dc:contributor hasValue "http://www.deri.org/foaf#james"
    dc:date hasValue "2004-10-22"
    dc:format hasValue "text/html"
    dc:language hasValue "en-us"
    dc:rights hasValue <http://deri.at/privacy.html>

    version hasValue "$Revision 1.17 $"
  endNonFunctionalProperties

usedMediators
  <http://www.wsmo.org/mediators/findRouteMediator>

capability

  sharedVariables ?route

  postcondition
  nonFunctionalProperties
    dc:description hasValue "the service needs a route, for which the
      start and end location have to be in Austria or in Germany"
  endNonFunctionalProperties

  definedBy
  ?route memberOf ts#route[
    startLocation hasValue ?start
    endLocation hasValue ?end
  ] and
  (?start = Germany or ?start = Austria) and
  (?end = Germany or ?end = Austria).

  effect
  nonFunctionalProperties
    dc:description hasValue "A trade is done if a route is available"
  endNonFunctionalProperties

  definedBy
  ?trade memberOf po#trade[
    item hasValue ?route
  ]
]
```

Listing 12 below defines the Web Service responsible for carrying out the credit card payment.

```

namespace {<"http://www.wsmo.org/ontologies/tripReservationService#">,
  dc <"http://purl.org/dc/elements/1.1#">,
  xsd <"http://www.w3.org/2001/XMLSchema#">,
  po <"http://www.wsmo.org/ontologies/purchaseOntology#">,
  loc <"http://www.wsmo.org/ontologies/location#"> ,
  trc <"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">}
webService <"http://www.wsmo.org/ontologies/performPaymentService.wsml">
nonFunctionalProperties
  dc#title hasValue "Perform Payment Service"
  dc#creator hasValue "DERI Innsbruck"
  dc#description hasValue "Service for performing a credit card transaction"
  dc#publisher hasValue "DERI International"
  dc#contributor hasValues "http://www.deri.org/foaf#james"
  dc#date hasValue "2005-02-28"
  dc#type hasValue <"http://www.wsmo.org/2004/d2/#service">
  dc#format hasValue "text/html"
  dc#language hasValue "en-us"
  dc#coverage hasValues {tc#austria, tc#germany}
  dc#rights hasValue <"http://deri.at/privacy.html">
  version hasValue "$Revision 1.0 $"
endNonFunctionalProperties

importsOntology {
  <"http://www.wsmo.org/ontologies/purchaseOntology#">
}

usesMediator
  <"http://www.wsmo.org/mediators/performPaymentMediator">

capability performPaymentService

  nonFunctionalProperties
    dc#title hasValue "Peform Payment Service"
    dc#creator hasValue "DERI Innsbruck"
    dc#description hasValue "description of the capability of the reservation
      service of providing trip reservations for Austria and Germany"
    dc#publisher hasValue "DERI International"
    dc#contributor hasValues "http://www.deri.org/foaf#james"
    dc#date hasValue "2005-02-28"
    dc#format hasValue "text/html"
    dc#language hasValue "en-us"
    dc#rights hasValue <"http://deri.at/privacy.html">
    version hasValue "$Revision 1.17 $"
  endNonFunctionalProperties

  sharedVariables ?creditCard, ?cardOwner, ?reservationCost

  assumption
    nonFunctionalProperties
      dc#description hasValue "the given credit card must be valid"
    endNonFunctionalProperties
    definedBy
      (?creditCard memberOf po#creditCard[
        owner hasValue ?cardOwner
      ]) and
      (po#validCreditCard(?creditCard, true))

  precondition
    nonFunctionalProperties
      dc:description hasValue "The client is in fact the owner of the credit card"

```

endNonFunctionalProperties

definedBy

```
?creditCard memberOf po#creditCard and
?reservationCost memberOf po#financialAmount and
?cardOwner memberOf prs#person
?creditCard[
  owner hasValue ?cardOwner
]
```

postcondition

nonFunctionalProperties

dc:description **hasValue** "acknowledge the transaction"

endNonFunctionalProperties

definedBy

```
?ack memberOf trc#acknowledgement[
  creditCard hasValue ?creditCard
]
```

effect

nonFunctionalProperties

dc:description **hasValue** "charge the credit card"

endNonFunctionalProperties

definedBy

```
?reservationCost memberOf po#financialAmount and
?creditCard memberOf po#creditCard[
  balance hasValue@pre ?initialBalance
] and
?creditCard[
  balance hasValue (?initialBalance-?reservationCost)
].
```

choreography reservationServiceChoreography

state <"http://www.wsmo.org/ontologies/tripReservationChoreographyOntology#">

guardedTransitions reservationServiceTransitionRules

if (?creditCard **memberOf** po#creditCard **and**

```
?reservationCost memberOf po#financialAmount and
?cardOwner memberOf prs#person
?creditCard[
  owner hasValue ?cardOwner
])
```

then

```
(?ack memberOf trc#acknowledgement[
  success hasValue true
] and
?reservationCost memberOf po#financialAmount and
?creditCard memberOf po#creditCard[
  balance hasValue@pre ?initialBalance
] and
?creditCard[
  balance hasValue (?initialBalance-?reservationCost)
])
```

5. Conclusions and further work

This document presented a core conceptual model for modeling WSMO

Choreographies and Orchestrations based on the ASMs methodology. Sequential and parallel workflows can be smoothly expressed in this core model. We did not fix the mathematical model defining parallelism since this is beyond a standardization proposal. We also did not define any tooling (like modeling the flow by UML activity diagrams and automatically mapping them on our ASM-based representation) since this is beyond the scope of this standardization proposal. However, it is also evident that we do not expect the user to directly work with such representation.

References

- [Börger, 1998]** Egon Börger: "High Level System Design and Analysis Using Abstract State Machines", Proceedings of the International Workshop on Current Trends in Applied Formal Method: Applied Formal Methods, p.1-43, October 07-09, 1998
- [Gurevich, 1995]** Yuri Gurevich: "Evolving Algebras 1993: Lipari Guide", Specification and Validation Methods, ed. E. Börger, Oxford University Press, 1995, 9--36.
- [Kopecky et. al., 2005]**J. Kopecky, D. Roman (authors): WSMO Grounding, WSMO deliverable D24.2 version 0.1. available from <http://www.wsmo.org/2005/d24/d24.2/v0.1/20050119/>
- [Roman et al., 2004]** D. Roman, H. Lausen, and U. Keller (eds.): Web Service Modeling Ontology (WSMO), WSMO deliverable D2 version 1.1. available from <http://www.wsmo.org/2004/d2/v1.1/>
- [W3C Glossary, 2004]** Hugo Haas, and Allen Brown (editors): Web Services Glossary, W3C Working Group Note 11 February 2004, available at <http://www.w3.org/TR/ws-gloss/>

Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, InfraWebs, SEKT, SWWS, ASG and Esperanto; by Science Foundation Ireland under the DERI-Lion project and by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects RW² and TSC. The editors would like to thank to all the [members of the WSMO working group](#) for their advice and input into this document.



[webmaster](#)

\$Date: Monday 28 February 2005 - 19:44:50\$
