



# D14v0.1. Choreography in WSMO

WSMO Working Draft 11 February 2005

**This version:**

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

**Latest version:**

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

**Previous version:**

<http://www.wsmo.org/2005/d14/v0.1/20050124/>

**Editor:**

Dumitru Roman  
James Scicluna

**Co-Authors:**

Emilia Cimpian  
Uwe Keller  
Michael Stollberg  
Cristina Feier  
Dieter Fensel

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

Copyright © 2004 [DERI](#)®, All Rights Reserved. [DERI](#) liability, trademark, document use, and software licensing rules apply.

---

## Table of contents

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

## 1. Introduction

Choreography in WSMO [[Roman et al., 2004](#)] is part of a service interface description; it 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 aim of this document is to provide a core conceptual model for describing choreographies in WSMO. The state-based mechanism for describing WSMO choreographies is based on the Abstract State Machines [Gurevich, 1995] methodology. The reason for choosing the ASMs as a basis for WSMO choreography is that ASMs provide a high flexibility in modelling systems, being at the same time scientifically well founded. For a detailed explanation on ASMs we refer the reader to [Börger, 1998]. The core principles of ASMs 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:

Listing 1. WSMO choreography definition

```
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 the document is organized as follows: Section 2 of the document describes what the state, Section 3 describes the guarded transitions, Section 4 presents an example of how a choreography is modeled, and Section 5 presents the conclusions and further directions.

## 2. 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 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.

### 3. 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. It cannot refer to elements of the signature with mode *out*. The *Updates* consist of arbitrary WSMO Ontology instance (see [Section 4.7](#) of [WSMO 1.1](#)) statements. It cannot refer to elements of the signature with mode *inandstatic*.

### 4. Choreography 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.

[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.

Listing 2. Trip Reservation 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](#), 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](#).

```

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.

```

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 <b>memberOf</b> trc#route[   startLocation <b>hasValue</b> tc#InnsbruckHbf,   endLocation <b>hasValue</b> tc#MünchenOstHbf ]} </pre>
The state after the execution of the first transition rule: a partial reservation has been created	<pre>{routeInstance <b>memberOf</b> trc#route[   startLocation <b>hasValue</b> tc#InnsbruckHbf,   endLocation <b>hasValue</b> tc#MünchenOstHbf ], reservationInstance <b>memberOf</b> trc#reservation[   reservedRoute <b>hasValue</b> routeInstance ]} </pre>
The state where a partial reservation exists and a credit card information is received	<pre>{routeInstance <b>memberOf</b> trc#route[   startLocation <b>hasValue</b> tc#InnsbruckHbf,   endLocation <b>hasValue</b> tc#MünchenOstHbf ], reservationInstance <b>memberOf</b> trc#reservation[   reservedRoute <b>hasValue</b> routeInstance ], creditCardInstance <b>memberOf</b> trc#creditCard[   owner <b>hasValue</b> prs#John,   expiryDate <b>hasValue</b> "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 <b>memberOf</b> trc#route[   start <b>hasValue</b> loc#InnsbruckHbf,   end <b>hasValue</b> loc#MünchenOstHbf ], reservationInstance <b>memberOf</b> trc#reservation[   reservedRoute <b>hasValue</b> routeInstance,   reservationHolder <b>hasValue</b> prs#John ], creditCardInstance <b>memberOf</b> trc#CreditCard[   owner <b>hasValue</b> prs#John,   expiryDate <b>hasValue</b> "2006-10-10" ]} </pre>

## 6. Conclusions and further work

This document presented a core conceptual model for modeling WSMO Choreographies based on the ASMs methodology. Future versions of this document will give a precise translation of the model to ASMs in order to benefit from using ASMs interpreters, thus having an environment for executing choreographies.

## 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 RW<sup>2</sup> 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](#)