



D14v0.1. Choreography in WSMO

WSMO Working Draft 22 October 2004

This version:

<http://www.wsmo.org/2004/d14/v0.1/20041022/>

Latest version:

<http://www.wsmo.org/2004/d14/v0.1/>

Previous version:

<http://www.wsmo.org/2004/d14/v0.1/20041014/>

Editor:

Dumitru Roman

Co-Authors:

Emilia Cimpian

Uwe Keller

Michael Stollberg

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 signature](#)
 - [3. State](#)
 - [4. Guarded transitions](#)
 - [5. Choreography description example](#)
 - [6. 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. 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 [\[Börger, 1998\]](#).

Taking the ASMs methodology as starting point, a WSMO choreography is defined as follows:

Listing 1. WSMO choreography definition

```
Class wsmoChoreography
  hasStateSignature type stateSignature
  hasState type state
  hasGuardedTransitions type guardedTransition
```

State Signature

A state signature defines the invariant elements of the state description.

State

A state is described by a set of "instanceOf" number "hasValue" statements.

Guarded transitions

Transition rules that express changes of states.

The rest of the document is organized as follows: [Section 2](#) of the document describes what is the signature of a state, [Section 3](#) describes the state, [Section 4](#) the guarded transitions, [Section 5](#) presents an example of how a choreography is modeled and [Section 6](#) presents the conclusions and further directions.

2. State signature

The signature of the states is given by elements of the WSMO Ontology, and it remains unchanged for all the states of the choreography:

Listing 2. State signature definition in choreography

```
Class stateSignature
  hasNonFunctionalProperty type nonFunctionalProperty
  hasImportedOntology type ontology
  hasUsedMediator type ooMediator
  hasConceptInChoreography type conceptInChoreography
  hasRelationInChoreography type relationInChoreography
  hasFunctionInChoreography type functionInChoreography
  hasAxiom type axiom
  hasIdentifiers type WSMOidentifiers
```

Non functional properties

Defined in WSMO, [Section 4.1](#).

Imported ontologies

Defined in WSMO, [Section 4.2](#).

Used mediators

Defined in WSMO, [Section 4.3](#).

Concepts in choreography

Concepts in choreography are a sub-Class of concepts defined in WSMO, [Section 4.4](#), having their non functional properties extended with the attribute *mode*, which can take one of the following values: *controlled*, *monitored*,

shared, or *out*.

Relations in choreography

Relations in choreography are a sub-Class of relations defined in WSMO, [Section 4.5](#), having their non functional properties extended with the attribute *mode*, as for concepts in choreography.

Functions in choreography

Functions in choreography are a sub-Class of functions as defined in WSMO, [Section 4.6](#), having their non functional properties extended with the attribute *mode*, as for concepts in choreography.

Axioms

Defined in WSMO, [Section 4.8](#).

Identifiers

All valid WSMO identifiers, as defined in WSMO, [Section 7.1](#).

When a concept, relation or function in a choreography is defined, the attribute *mode* of their non functional properties must be defined, and can take one of the following values:

- *static* - meaning that the instances, once are created, can not be modified.
- *controlled* - meaning that the instances are created or modified by and only by the service.
- *in* - meaning that the instances are created or modified by and only by the environment and the service is only allowed to monitor(read) them.
- *shared* - meaning that the instances can be modified by either the service or the environment.
- *out* - meaning that the instances are created or modified by and only by the service and monitored only by the environment.

3. State

A state is characterized by its signature and a set of instances. These instances are the result of the user inputs, or retrieved from the service's ontology as a result of an execution.

Listing 3. State definition in choreography

```
Class state
  hasSignature type stateSignature
  hasInstances type instance
```

4. 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 axiom without free variables, formulated in the given signature of the state using the logical language for defining formal statements defined in WSMO.

The *Updates* consist of arbitrary WSMO Ontology instance (see [Section 4.7](#) of

[WSMO 1.1](#)) statements.

Besides this **if then** rule, we allow also rules of the form:

- **choose** x with $Cond$ **do** R ,

meaning to execute rule R with an arbitrary x among those satisfying the condition $Cond$.

- **forall** x with $Cond$ **do** R ,

meaning to execute simultaneously rule R for each x satisfying the condition $Cond$.

5. Choreography description example

A service makes reservations for a trip, for which the starting and ending points are located in Austria or Germany. First, the service receives a route; based on some internal decision [1], in this case explicitly specified (if there is a route), the response of the service can either be a trip or an error message, containing the reason for the error. If the user receives the trip he is allowed to send a request for a reservation. After the user's request for the reservation, the service requests the user's credit card. Based again on some internal decision, this time not specified, the service can either send a confirmation to the service, or an error message.

Listing 4 below presents the signature of the state for this example.

Listing 4. State signature example

```
//The signature of the state

namespace <<http://www.wsmo.org/example/stateSignatureExample/>>
  dc:<<http://purl.org/dc/elements/1.1#>>
  prs:<<http://www.wsmo.org/2004/d3/d3.3/v0.1/20041008/resources/owlPersonMediator.wsml>
  xsd: <<http://www.w3.org/2001/XMLSchema#>>
  tc:<<http://www.wsmo.org/ontologies/trainConnection#>>
  po: <<http://www.wsmo.org/ontologies/purchase#>>
  targetNamespace: <<http://www.wsmo.org/example/stateSignatureExample/>>

stateSignature <<http://www.wsmo.org/example/stateSignatureExample.wsml>>

nonFunctionalProperties
  dc:title hasValue "State signature"
  dc:description hasValue "An example of a state signature."
  dc:date hasValue "2004-10-22"
  dc:format hasValue "text/plain"
  dc:language hasValue "en-US"
  version hasValue "$Revision 0.1 $"
endNonFunctionalProperties

//concepts
concept route
  nonFunctionalProperties
    dc:description hasValue "concept of a route between two stations"
    mode hasValue in
  endNonFunctionalProperties
  sourceLocation type tc:station
```

```

destinationLocation ofType tc:station

concept trip subConceptOf tc:trip
nonFunctionalProperties
  dc:description hasValue "a trip as defined in the 'tc' ontology and adapted for
    choreography"
  mode hasValue out
endNonFunctionalProperties

concept reservation
nonFunctionalProperties
  dc:description hasValue "concept of reservation, containing a reservation holder"
  mode hasValue in
endNonFunctionalProperties
reservationHolder ofType prs:person

concept creditCard subConceptOf po:creditCard
nonFunctionalProperties
  dc:description hasValue "concept of credit card as defined in the 'po' ontology and
    adapted for choreography"
  mode hasValue shared
endNonFunctionalProperties
concept confirmation
nonFunctionalProperties
  dc:description hasValue "concept of a confirmation for a trip"
  mode hasValue out
endNonFunctionalProperties
trip ofType trip
customerName ofType prs:person
no ofType xsd:integer

concept failure
nonFunctionalProperties
  dc:description hasValue "the fact of not achieving the desired end"
  mode hasValue out
endNonFunctionalProperties
reason ofType xsd:string

//relations
relation routeExists
nonFunctionalProperties
  dc:description hasValue "route existence relationship between two stations"
endNonFunctionalProperties
sourceLocation ofType tc:station
destinationLocation ofType tc:station

function internalDecision
nonFunctionalProperties
  dc:description hasValue "a buildin nullary function which evaluates to true if some service
    internal conditions are true"
endNonFunctionalProperties

//WSMO identifiers - WSMO variables in our example for storing instances
WSMOidentifier providedTrip
WSMOidentifier nameOfCustomer

```

Listing 5 below presents the service description for the example.

```

namespace <<<http://www.wsmo.org/ontologies/serviceExample#>>
  sg:<<http://www.wsmo.org/example/stateSignatureExample#>>
  targetnamespace:<<http://www.wsmo.org/ontologies/serviceExample#>>

service <<http://www.wsmo.org/ontologies/serviceExample.wsml>>

nonFunctionalProperties
  dc:title hasValue "Trip Reservation Service"
  dc:creator hasValue "DERI Innsbruck"
  dc:description hasValue "service for online trip reseravtions 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: 1.14 $"
endNonFunctionalProperties

importedOntologies <<http://www.wsmo.org/example/stateSignatureExample>>

capability _#
  precondition
    axiom _#
      nonFunctionalProperties
        dc:description hasValue "the service receives a route, for which the
          start and end location have to be in Austria or in Germany, a reservation request,
          for which a credit card is needed to be payed with."
        endNonFunctionalProperties
      definedBy
        (?route memberOf sg:route
          startLocation hasValue ?start,
          endLocation hasValue ?end) and
        (?start.locatedIn = austria or ?start.locatedIn = germany) and
        (?End.locatedIn = austria or ?End.locatedIn = germany) and
        (?reservation memberOf sg:reservation) and
        (?CreditCard memberOf po:creditCard)
      postcondition
        axiom _#
          nonFunctionalProperties
            dc:description hasValue "in case of a sucessful execution of a service, a confirmation
              is sent to the user"
            endNonFunctionalProperties
          definedBy
            (?confirmation memberOf sg:confirmation)

interface _#
  choreography _#

  //If the service receives a route from the user, and if the route exists, then a trip
  //is sent to the user and the trip is stored (in a WSMO identifier - a variable
  //in this case - "providedTrip") as it is needed in subsequent steps during the communication;
  //if the service is not able to find a route, then a failure is sent to the user.
  forall ?x with (?x memberOf sg:route)
    if routeExists(?x.sourceLocation, ?x.destinationLocation) then
      choose ?y with (?y memberOf sg:trip and
        ?y.start hasValue ?x.sourceLocation and
        ?v.end hasValue ?x.destinationLocation)

```

```

    sg:providedTrip hasValue ?y
  else
    choose ?y with (?y memberOf sg:failure and
      ?y.reason hasValue "Inexistent route")

  //If the service receives a reservation request from the user, the service requests the user's
  //credit card and stores the user's name (in a WSMO identifier - a variable in this
  //case - "nameOfCustomer").
  forall ?x with (?x memberOf sg:reservation)
    sg:nameOfCustomer hasValue ?x.reservationHolder
    choose ?y with (?y memberOf sg:creditCard)

  //If the service receives the credit card, then based on some internal decision which is not
  //shown to the user, it can either send a confirmation containing the trip, the user's name
  //and the confirmation number for the trip or sends the failure reason for not being able
  //to provide the user with a confirmation.
  forall ?x with (?x memberOf sg:creditCard)
    if internalDecision then
      choose ?y with (?y memberOf confirmation and
        ?y.trip hasValue sg:providedTrip
        ?y.customerName hasValue sg:nameOfCustomer)
    else
      choose ?y with (?y memberOf failure)

```

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.

[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/>.

Acknowledgement

The work is funded by the European Commission under the projects [DIP](#), [Knowledge Web](#), [SEKT](#), [SWWS](#), and [Esperanto](#); by [Science Foundation Ireland](#) under the [DERI-Lion](#) project; and by the Vienna city government under the [CoOperate](#) program.

The editors would like to thank to all the [members of the WSMO working group](#) for their advice and input into this document.

[1] The internal decision of the service can be explicitly specified by the service in the form of a condition. This situation could help the user of the service in the sense that he/she can assume the validity of the condition, based on the definition of the condition in some ontology that the service uses.



[webmaster](#)

\$Date: Friday 22 October 2004 - 20:59:23\$
