



D15v0.1. Orchestration in WSMO

DERI Working Draft 29 May 2004

This version:

<http://www.wsmo.org/2004/d15/v0.1/20040529/>

Latest version:

<http://www.wsmo.org/2004/d15/v0.1/>

Previous version:

<http://www.wsmo.org/2004/d15/v0.1/20040529/>

Editors:

Dumitru Roman
Laurentiu Vasiliu
Christoph Bussler

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. Modalities to represent Orchestrations](#)
 - [2.1 Workflow representation](#)
 - [2.2 State-based representation](#)
 - [3. Conclusions and further work](#)
 - [References](#)
 - [Acknowledgement](#)
-

1. Introduction

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).

2. Modalities to represent Orchestrations

Two modalities to represent orchestrations are presented in Section 2.1 (Workflow representation) and Section 2.2 (State-based representation).

2.1 Workflow representation

In this representation, an orchestration is defined in the following way:

Listing 1. Workflow representation of an orchestration

```
orchestration[
proxies =>> proxy
controlFlow => controlFlow
dataFlow => dataFlow
]
```

Proxies

A *proxy* is an elementary unit of the orchestration. It can be either a WSMO goal or a local representation of a WSMO service provider that the service that describes the orchestration uses in order to fulfill its functionality. In case a proxy is a local representation of a WSMO service provider, a behaviour (i.e a WSMO choreography) is associated with it, which is complementary to the behaviour of the WSMO service provider.

Control Flow

Control Flow describes the order in which the proxies should be used and the conditional logic specifying whether they should be performed at all, or possibly looped.

Data Flow

Data Flow specifies how the data generated by a proxy is used by other proxies in the orchestration.

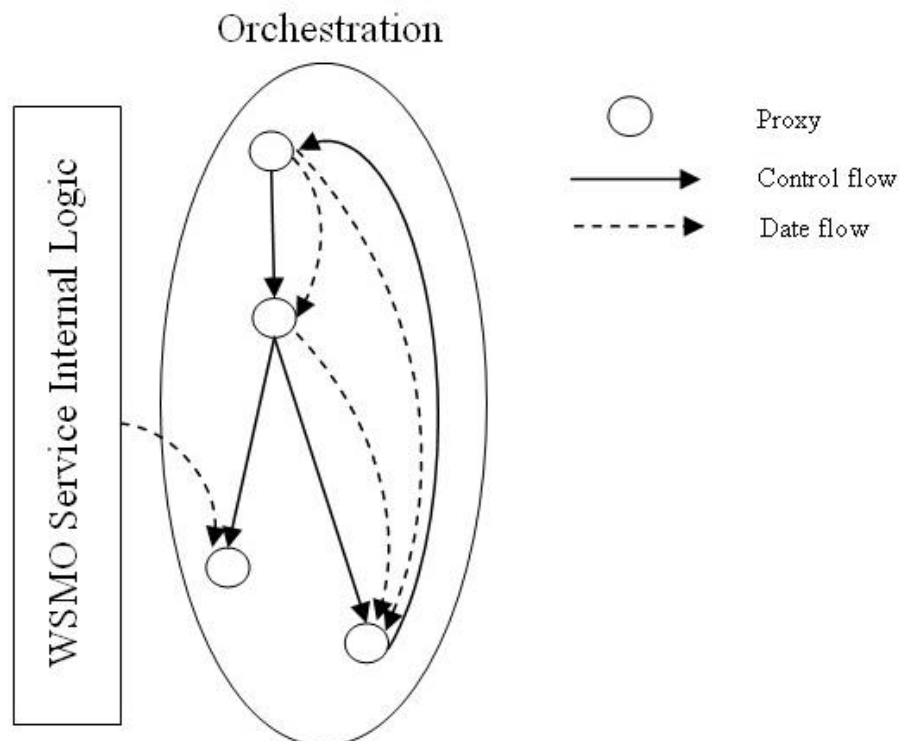


Figure 1. Workflow representation of an orchestration.

Figure 1 depicts the workflow representation of an orchestration. Adopting this representation for the WSMO orchestration would allow the reuse of the extensive work done in the area of workflow systems.

2.2 State-based representation

In this representation, an orchestration is defined in the following way:

Listing 2. State-based representation of an orchestration

```
orchestration[
activities =>> activity
conditions =>> condition
rules =>> rule
]
```

Activities

An *activity* is an elementary unit of the orchestration. For each activity in every MEP of all WSMO service providers that the service that describes the orchestration uses in order to fulfill its functionality, an *activity* with the complementary behaviour is generated in the orchestration.

Conditions

A *condition* is a boolean expression which specifies a transition between the states of the orchestration. A *state* of an orchestration can be seen as a particular association of variable names, which appear in the conditions (i.e. inputs, outputs of the *activities*), to values, in the style of a dictionary: {(name1, val1), (name2, val2), ... }.

Rules

A *rule* specifies which activities can be executed and under which conditions.

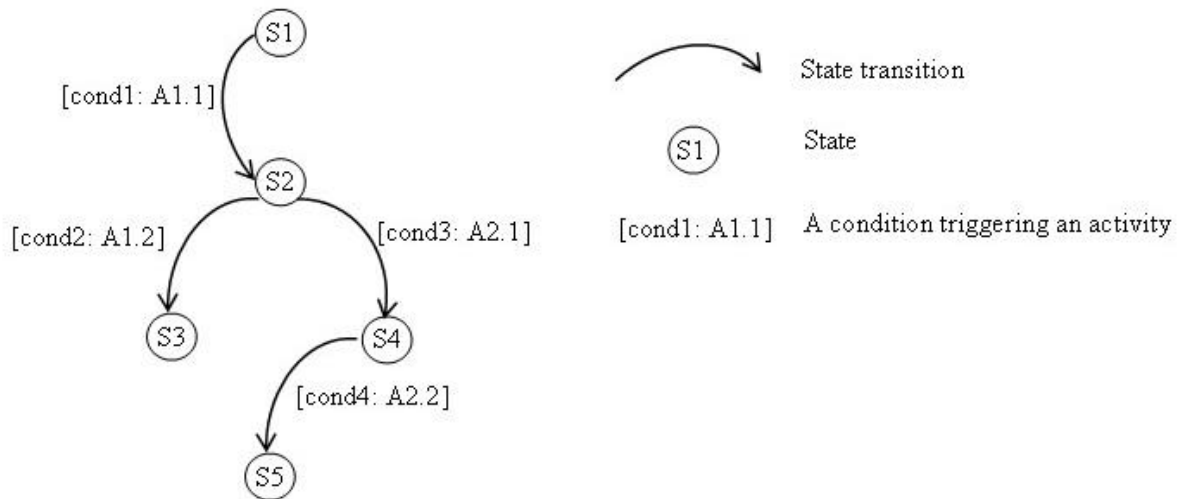


Figure 2. An example of an orchestration specified using the state-based representation.

The figure above shows an example of an orchestration specified using the state-based representation. The orchestration consists of 4 activities, generated from 2 WSMO service providers (A1.1 and A1.2 from one service and A2.1 and A2.2 from the other service) that the service that describes the orchestration uses in order to fulfill its functionality. Each state consists of the inputs/outputs of all activities and their values. The conditions under which the activities are executed is given by the rules presented in the listing below.

```

activities: A1.1, A1.2, A2.1, A2.1
conditions: cond1, cond2, cond3, cond4
rules:
if currentState = s1 and cond1 = true then execute (A1.1)
if currentState = s2 and cond2 = true then execute (A1.2)
if currentState = s2 and cond3 = true then execute (A2.1)
if currentState = s4 and cond4 = true then execute (A2.2)

if currentState = s1 and finished (A1.1) then currentState = s2
if currentState = s2 and finished (A1.2) then currentState = s3
if currentState = s2 and finished (A2.1) then currentState = s4
if currentState = s4 and finished (A2.2) then currentState = s5

```

5. Conclusions and further work

This document presented a preliminary draft related to orchestration in WSMO. If the orchestration will be represented using the workflow representation then further investigation in the workflow systems will be needed. If the orchestration will be represented using the state-based representation, which is similar to Abstract State Machine [Gurevich, 1995] model, then further investigation in using [Abstract State Machine Language \[AsmL\]](#) to model orchestrations will be needed.

References

[AsmL] Abstract State Machine Language. Available at <http://research.microsoft.com/fse/asm/>

[Gurevich, 1995] Yuri Gurevich: *Evolving Algebras 1993: Lipari Guide*, Specification and Validation Methods, ed. E. Börger, Oxford University Press, 1995, 9--36.

Acknowledgement

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperanto, COG and h-TechSight; 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.

[webmaster](#)