



D14v0.1. Choreography in WSMO

DERI Working Draft 27 May 2004

This version:

<http://www.wsmo.org/2004/d14/v0.1/20040527/>

Latest version:

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

Previous version:

<http://www.wsmo.org/2004/d14/v0.1/20040422/>

Editors:

Dumitru Roman
Laurentiu Vasiliu
Michael Stollberg
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. Message Exchange Patterns](#)
 - [2.1 State-less MEPs](#)
 - [2.2 State-based MEPs](#)
 - [3. Conclusions and further work](#)
 - [References](#)
 - [Acknowledgement](#)
-

1. Introduction

Choreography in WSMO provides the necessary information for the user (human or not) of a WSMO compliant web service to interact with it.(i.e. it describes how the service works and how to access the service from the user's perspective). A choreography achieves this by instantiating a Message Exchange Pattern (MEP) for the given capability of a WSMO web service.

2. Message Exchange Patterns

A `message exchange pattern` is a template that establishes a pattern for the exchange of messages between two communicating parties. It specifies a sequence of conditional speech acts that either send or receive messages. We classify message exchange pattern in two types of patterns: state-less MEPs, which are described in Section 2.1 and state-based MEPs, which are described in Section 2.2.

2.1 State-less MEPs

State-less MEPs model stimulus-response patterns. They are defined as follows:

Listing 1. State-less MEP

```
state-lessMEP[
input => input
output => output
errorGenerationRule => errorGenerationRule
]
```

Input

The `input` defines the message that the service receives

Output

The `output` defines the message that the service sends.

Error Generation Rule

MEPs specify their error generation model by indicating where `errors` may occur:

- `Error Replaces Message`: any message after the first in the pattern may be replaced with an error message, which must have identical cardinality and direction
- `Message Triggers Error`: any message, including the first, may trigger an error message in response.
- `No Errors`: no errors may be generated.

Based on the cardinality of the messages, and the error generation rule that a MEP uses, we distinguish the following types of state-less MEPs:

- `in-only`: the MEP consists of only one message which has the direction "in" and uses the `No Errors` rule.
- `robust in-only`: the MEP consists of only one message which has the direction "in" and uses the `Message Triggers Error` rule.
- `in-out`: the MEP consists of one message which has the direction "in" followed by another message which has the direction "out" and uses the `Error Replaces Message` rule.
- `out-only`: the MEP consists of only one message which has the direction "out" and uses the `No Errors` rule.
- `robust out-only`: the MEP consists of only one message which has the direction "out" and uses the `Message Triggers Error` rule.
- `out-in`: the MEP consists of one message which has the direction "out" followed by another message which has the direction "in" and uses the `Error Replaces Message` rule.

2.2 State-based MEPs

State-based MEPs model conversations. They are defined as follows:

```
state-basedMEP[
activities =>> activity
conditions =>> condition
rules =>> rule
]
```

Activities

An `activity` is an elementary unit of the State-based MEP. It can be either a state-less MEP (for example, the action of receiving a message, i.e. in-only operation, or sending a message and waiting for a reply, i.e. out-in operation) or a non-specified action.

Conditions

A `condition` is a boolean expression which specifies a transition between the states of the state-based MEP. A `state` of a state-based MEP can be seen as a particular association of variable names, which appear in the conditions (i.e. inputs, outputs of the state-less MEPs), 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.

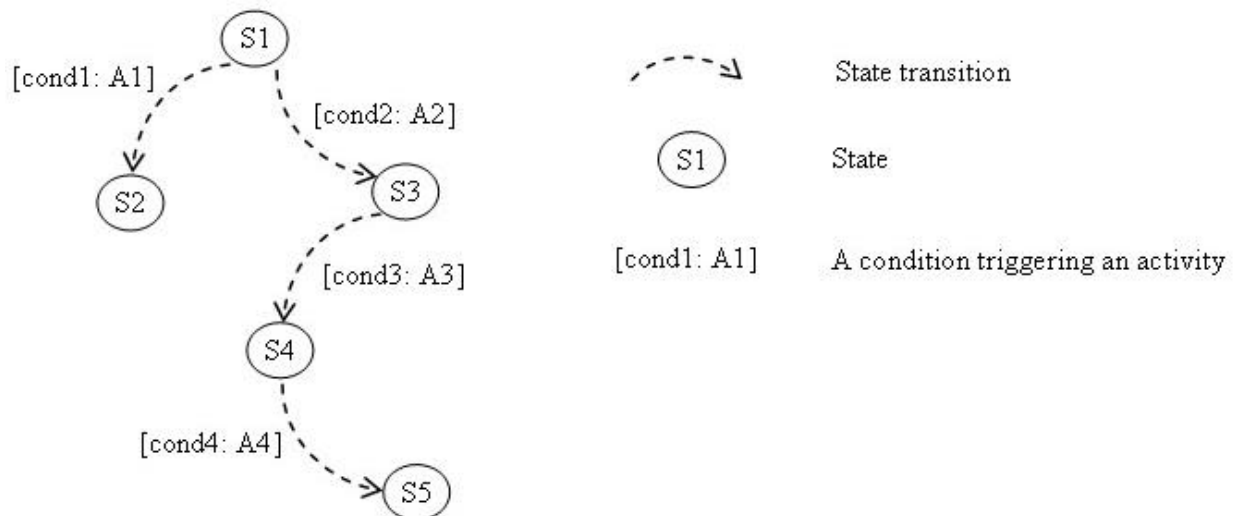


Figure 1. An example of a State-based MEP.

The figure above shows an example of a state-based MEP. The state-based MEP consists of 5 activities, which can be state-less MEP and/or non-specified actions. Each state consists of the inputs/outputs of all the state-less MEPs that are part of the state-based MEP and their values. The conditions under which the activities are executed is given by the rules presented in the listing below.

```
activities: A1, A2, A3, A4
conditions: cond1, cond2, cond3, cond4
rules:
if currentState = s1 and cond1 = true then execute (A1)
if currentState = s1 and cond2 = true then execute (A2)
if currentState = s3 and cond3 = true then execute (A3)
if currentState = s1 and cond4 = true then execute (A4)

if currentState = s1 and finished (A1) then currentState = s2
if currentState = s1 and finished (A2) then currentState = s3
if currentState = s3 and finished (A3) then currentState = s4
if currentState = s4 and finished (A4) then currentState = s5
```

5. Conclusions and further work

This document presented a preliminary draft related to choreography in WSMO. Given the fact that the way the choreography modeled in WSMO is similar to Abstract State Machine [Gurevich, 1995] model, we intend to investigate how [Abstract State Machine Language \[AsmL\]](#) can be used for modelling WSMO choreography.

References

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

[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, Esperonto, 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](#)