



D14v0.1. Choreography in WSMO

DERI Working Draft 17 July 2004

This version:

<http://www.wsmo.org/2004/d14/v0.1/20040717/>

Latest version:

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

Previous version:

<http://www.wsmo.org/2004/d14/v0.1/20040620/>

Editor:

Dumitru Roman

Authors:

Dumitru Roman
Marin Dimitrov
Michael Stolberg

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. Conceptual Model](#)
 - [3. Formal representation](#)
 - [4. Grounding](#)
 - [Web Services](#)
 - [5. Existing approaches to Web Services Choreographies](#)
 - [6. Conclusions and further work](#)
 - [References](#)
 - [Acknowledgement](#)
-

1. Introduction

The aim of this document is to provide a conceptual model and a formalism for describing choreographies in WSMO.

The conceptual model of WSMO differentiates between Chreography and Choreography Interface.

WSMO Choreography is a collaboration between some WSMO participants in order to achieve a certain goal[1]. It describes the tasks (simple or complex actions) [2] that WSMO participants need to perform in order to achieve that goal, the interactions associated to the executions of these tasks, and the relations between tasks. As [\[Dijkman & Dumas, 2004\]](#) observe, the reasons for defining a choreography are twofold:

- To be able to verify whether the joint behaviour of some service providers or the behaviour of a particular service provider conforms to originally intended collaboration.
- To provide a starting point for a concrete collaboration that can be set up (i.e. to have a standard collaboration in which individual service providers can indicate the tasks they can perform and their associated interactions).

WSMO Choreography Interface describes the behaviour of a particular WSMO service in its communication with its environment (other parties). It corresponds to the communication tasks and their

associated interactions which the service performs and the relations between these communication tasks. Choreography Interface provides a point of entry for a service into a choreography, allowing it to check whether it can participate or not in a choreography. The reason for specifying a choreography interface is to make clear what a WSMO service can do and/or expects in terms of communication tasks and their associated interactions.

The rest of the document deals only with the WSMO Choreography interface; WSMO Choreography will be considered in a later stage of the development of WSMO Choreography Interface. [Section 2](#) presents the conceptual model for WSMO Choreography Interface, [Section 3](#) aims to provide a formal representation for the conceptual model, [Section 4](#) provides a grounding for the conceptual model, [Section 5](#) aims to provide a comparison to existing approaches to choreography modeling, and [Section 6](#) concludes this document and points out further directions.

2. Conceptual Model

The conceptual model of the WSMO choreography interface is presented in [Figure 1](#).

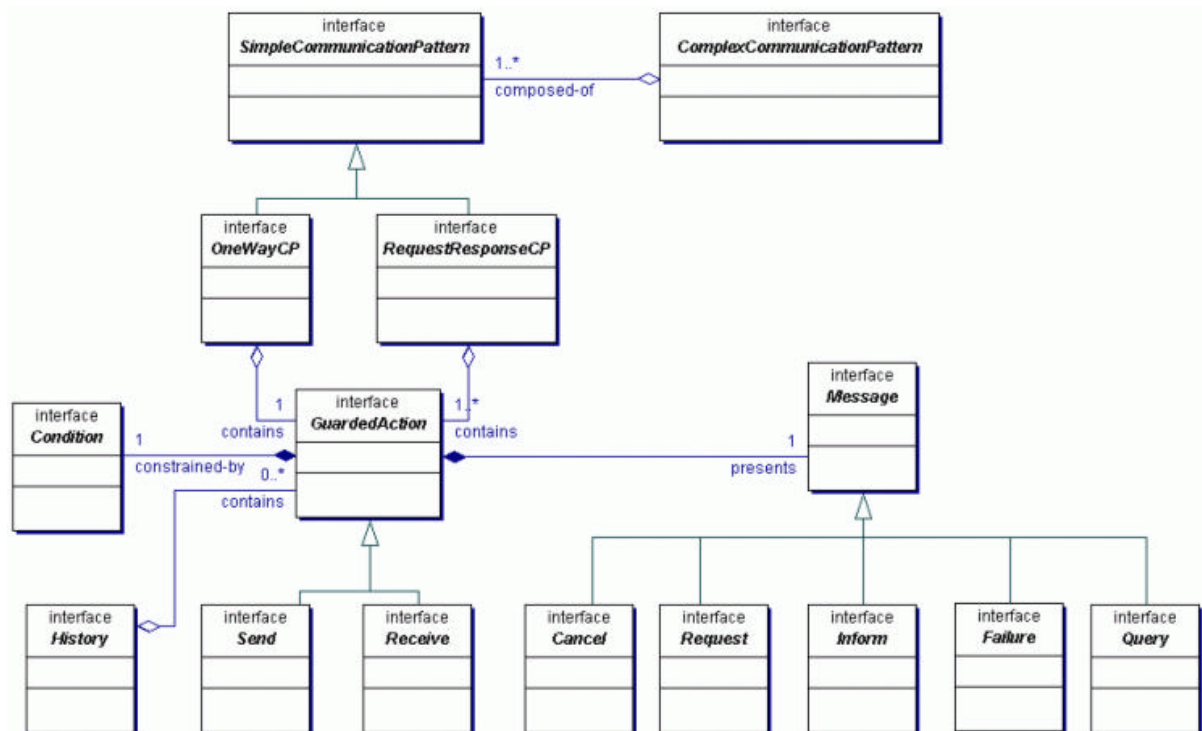


Figure 1. Choreography Interface Meta-model

A *Communication Pattern* is a pattern for communication with the service and constrains the order of messages that are exchanged during the communication by the service.

A Communication Pattern consists of a set of *guarded actions*. A guarded action is an action for which its execution is constrained by a condition.

An *action* represents the basic unit of behavior of a WSMO service. It can be of two types:

- *Receive* – the action of receiving a message.
- *Send* – the action of sending a message.

Proposals like [\[Hull et al., 2003\]](#) consider also the possibility for non specified actions to appear in between send and receive actions to allow the modeling of the services' internal state transitions. We adopt here the idea from [\[Frølund & Govindarajan, 2003\]](#) to do not model the internal transitions of the service. This separation of concerns provides the basis for inter-operability amongst services while preserving loose coupling.

Conditions are boolean expression which specify, when evaluated, whether or not their associated actions can be performed. An action is always guarded by a condition. We write $\langle c, a \rangle$ to denote the fact that action a can be performed when condition c is true.

We classify Communication Patterns in *Simple Communication Patterns* and *Complex Communication*

Patterns.

Simple Communication Patterns are:

- *One Way* – a single message is exchanged. This pattern can be further refined in:
 - *In* - the service receives a message. This pattern consists of one Receive action.
 - *Out* - the service sends a message. This pattern consists of one Send action.
- *Request/Response* – two messages are exchanged. This pattern can be further refined in:
 - *In/Out* – the service receives a message and sends back a correlated message. This pattern consists of one Receive action followed by one Send action.
 - *Out/In* - the service sends a message and receives a correlated message. This pattern consists of one Send action followed by one Receive action.

Complex Communication Patterns can be derived by composing simple communication patterns. We adopt here the ordering constructs of [\[WS-CDL, 2004\]](#) and [\[Frølund & Govindarajan, 2003\]](#):

- Sequence - restricts the series of enclosed (simple and/or complex) patterns to be enabled sequentially, in the same order that they are defined.
- Parallel - contains one or more (simple and/or complex) patterns that are enabled concurrently.
- Choice - defines that only one of two or more (simple and/or complex) patterns should be performed.

In terms of synchronization, Communication patterns can be either Synchronous, i.e the sender sends a message and then stops processing while it waits for receiving a correlated message (it might influence further processing on the sender's side), or Asynchronous, i.e the sender continues to process while waiting for receiving a correlated message. *In/Out* and *Out/In* can be either Synchronous or Asynchronous, *In*, *Out* are Asynchronous. However, this is a problem of implementation of the patterns and doesn't directly concern our model.

A Communication Pattern is said to be instantiated if:

- its conditions are specified
- the content and intention (the intended meaning of the message) of the *messages* are specified. To specify the intention of a message, we consider as a starting point a relevant, extensible, sub-set of FIPA ACL Communicative Acts [\[FIPA ACL, 2002\]](#) and adapt it to the context of WSMO services:
 - *Request* – the intention of requesting to perform some tasks.
 - *Inform* - the intention of informing a fact.
 - *Failure* – the intention of notifying that an error occurred.
 - *Cancel* – the intention of cancelling the communication.
 - *Query* - the intention of quering for information.

As a content language for messages, the WSML is used.

We also adopt the notion of a conversation *history* from [\[Frølund & Govindarajan, 2003\]](#) as the sequence of messages exchanged (or guarded actions executed) so far.

3. Formal representation

Formalism to be considered:

- Abstract State Machines
- State Machines
- Pi calculus
- Petri nets

4. Grounding

We provide grounding (i.e. show how the conceptual model can be coupled to existing technologies) to Web Services technologies.

4.1 Web Services

Web service technologies provide a loosely coupled infrastructure that enables cross-enterprise integration. They consist of multiple layers that, when stacked together, form the basis for a standard mechanism for discovering, describing, and invoking the functionality provided by a standalone web service:

- Protocol layer (any of the standard Internet protocols may be used to invoke web services over the network, e.g. HTTP, FTP, SMTP)
- Packaging layer (SOAP - a lightweight protocol designed for the exchange of information; it provides the mechanism for enabling cross-platform integration independent of any programming language and distributed object infrastructure)
- Information Layer (XML - a meta-language that enables cross-platform data interchange using a standard method for encoding and formatting information)
- Service Layer (WSDL - an XML language for describing Web services)

As the description of the choreography interface is part of a service description, the focus here will be on how a complex behavior of a WSMO service can make use of the service layer (WSDL) of the web services stack. For this, the following equivalences are defined:

- An *action* is equivalent to a WSDL operation.
- *In/Out* Communication Pattern is equivalent to In/Out WSDL MEP
- *Out/In* Communication Pattern is equivalent to Out/In WSDL MEP
- *In* Communication Pattern is equivalent to In-only WSDL MEP
- *Out* Communication Pattern is equivalent to Out-only WSDL MEP

5. Existing approaches to Web Services Choreographies

Approaches to be considered:

- WS-CDL
- WSCI
- BPEL4WS Abstract Processes
- [Benatallah et al.,2003]
- [Frølund & Govindarajan, 2003]
- [Bultan et al., 2003]
- [Hull et al., 2003]

6. Conclusions and further work

This document presented a conceptual model for modeling WSMO Choreography Interfaces and proposed a grounding to web services technologies . Depending on the acceptance of this model, further versions of this document will contain a mapping to some existing formalism and comparison to existing approaches.

References

[Benatallah et al.,2003] Benatallah, B., Sheng, Q., & Dumas, M. (2003): The Self-Serv environment for web services composition. *IEEE Internet Computing*, 7(1), 40–48.

[Bultan et al., 2003] Bultan, T., Fu, X., Hull, R., & Su, J. (2003, May): *Conversation specification: A new approach to design and analysis of e-service composition*. Proc. of the Intl. Conf. on the World Wide Web (WWW) (pp.403–410). Budapest, Hungary: ACM Press.

[Dijkman & Dumas, 2004] R. Dijkman and M. Dumas: *Service-oriented Design: A Multi-viewpoint Approach*. CTIT Technical Report Series No. 04-09, Centre for Telematics and Information Technology, University of Twente, The Netherlands, February 2004.

[Hull et al., 2003] Richard Hull, Michael Benedikt, Vassilis Christophides, and Jianwen Su: *E-services: A look behind the curtain*. In Proceedings of ACM Symposium on Principles of Database Systems. ACM, 2003.

[FIPA ACL, 2002] <http://www.fipa.org/specs/fipa00037/SC00037J.html>

[Frølund & Govindarajan, 2003] Frølund, S., & Govindarajan, K.: *cl: A language for formally defining web services interactions* (Technical Report HPL-2003-208). Hewlett-Packard.

[WS-CDL, 2003] <http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/>

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] Note that this goal is not not necessarily equivalent to the way WSMO-Standard defines a goal, as it involves more than one party when defining the goal.

[2] The tasks described in a choreography are only tasks that involve communication between parties involved.



[webmaster](#)

\$Date: Saturday 17 July 2004 - 15:19:02\$
