

SOPHIE – Architecture and Overall Algorithm of a Choreography Service

Sinuhe Arroyo¹, José-Manuel López-Cobo²

¹ Digital Enterprise Research Institute, Innsbruck, Austria

sinuhe.arroyo@uibk.ac.at

²Atos Origin SAE, Madrid, Spain

jose.lopez@atosorigin.com

Abstract. This paper describes the architecture of **SOPHIE**, a conceptual choreography framework, which attempts to provide a complete solution to the communication requirements of heterogeneous services. It allows the production of the intermediate structures (services) that allow overcoming the heterogeneities between services from the semantic descriptions of the conversational patterns they follow.

1 Introduction and Related Work

Services communicate with each other by exchanging messages, which allow them to make or to respond to requests. In doing so, services follow different structural, behavioral and semantic models [1], requiring putting in place the mechanisms that allow overcoming their differences.

A number of approaches exist, such as BPEL4WS [3], WS-CDL [5], WSCI [4] or WSMO – Choreography [6], that can be used to model the external visible behavior of services. However none of these approaches represents a complete solution to the problem due to as they do not facilitate the means to overcome the natural heterogeneity among services in a dynamic way.

Thus, new initiatives are needed that overcome these limitations and provide interoperation mechanisms among services, which increase the degree of de-coupling and eliminate static dependencies. **SOPHIE**¹ [2] has precisely these objectives.

SOPHIE enjoys a great degree of flexibility thanks to its technological independence. It does not make any assumptions about the underlying communication framework (WSDL, SOAP), ontological language (WSML, OWL, RDF, etc) or behavioral paradigm (Abstract State Machines (ASMs), Petri nets, temporal logic, etc). In doing so, it focuses at the message exchange level assuming that messages are going to be delivered, not mixing in the specification the transport or service descrip-

¹ SOPHIE is an acronym for Semantic web services chOreograPHi servIce

tion details. Further, it does not make any assumption about the semantic language or behavioral model used, allowing the most suitable one to be plugged-in.

In the following the architecture of the SOPHIE choreography framework is presented. Section 2 covers the overall framework in which SOPHIE is built on, exposing the principles underlying in this initiative. In Section 3, we describe the Algorithm used for the Choreography. Further conclusions and future work will be exposed in Section 4.

2 The SOPHIE Choreography Framework

SOPHIE is a conceptual framework and architecture for a choreography service realized as a Semantic Service Oriented Architecture (SSOA). Services that use the choreography service fall into two main categories, namely, *initiating parties* and *answering services*. Both parties produce and consume messages. Additionally, initiating parties indicate the choreography framework by means of any of their constituents correlating services that the infrastructure for the interoperation of heterogeneous message exchanges should be established.

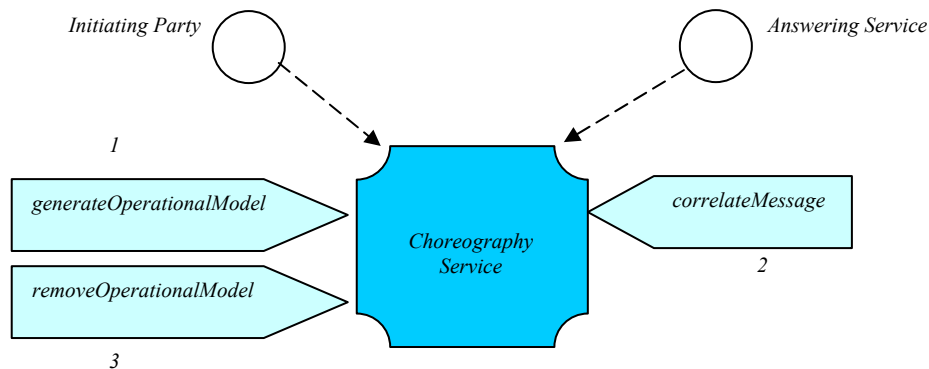


Fig. 1. Choreography Service

Figure 1 shows a high level architecture of the conceptual framework. Informally, initiating parties indicate that want to communicate with an answering service by means of “*generateOperationalModel*” (1). Once an operational model that allows the interoperation among the heterogeneous message exchanges has been created, parties can start submitting messages by means of the “*correlateMessage*” (2) primitive. Messages will go through the designated operational model, forwarding the framework the message(s) to the receiving party according to its choreography. Finally, when the conversation is finished, either party indicates that the operational model for a given conversation can be put off line, by means of the primitive “*removeOperationalModel*”.

This section gives a formal definition of the syntax and semantics of the conceptual framework. Firstly, some basic terminology and concepts that will be used are

defined. Later, the interface functions exported by the framework, together with the structural, functional and behavioral models are depicted. Finally, the semantics of the framework are described.

2.1 Basic Principles

The conceptual model presented in this paper describes the syntax and semantics of conceptual framework for choreography as semantic and syntactic models. The syntactic model depicts three different complementary models, namely: structural, behavioral and operational.

The structural model provides the grounding pillars of the framework. The behavioral model permits to model the conduct of the structural model. Finally, the operational model facilitates the means to allow the interoperation of different behavioral models. This layered approach enables a straight mechanism to extend the different models, for example Petri nets, temporal logic or transaction logic can be used instead of Abstract State Machines (ASMs) for the behavioral model.

The work presented here defines the behavioral model as ASMs. Still, any other suitable paradigm can be easily plugged-in. The terms used, in particular the terms choreography, state and guarded transition, follow the formalism proposed in [6] and [1]. Furthermore, the semantic model is currently based on WSM. Nonetheless, the design allows to easily extending the grammar and ontology of SOPHIE to accommodate any other ontology language.

Since the work models a software system that is distributed over the Web means of message exchanges including, some abstractions are needed for software components, network nodes, communication, and in general to model the interaction between components. Here the elements of the model are defined.

2.1.1 Entities

Initiating parties and *answering services* are active entities (parties) that produce and consume messages in a conversation. Active entities play both roles at the same time. However, initiating parties commence the conversation, while answering services react on the messages received from them. For simplicity both active entities are modeled as separate parties. Additionally, *correlating services* define active entities that enable interoperation among all other active entities.

2.1.2 Conversations

A *conversation* represents the logical entity that permits to group a set of related message exchanges among parties. Conversations are composed of a number of building blocks.

Elements represent elementary unit of data that build up *documents*. Documents are complete, self-contained groups of elements that are transmitted over the wire within *messages*. Messages characterize the primitive piece of data that can be exchanged among parties. As messages are exchanged, a variety of recurrent scenarios can be played out. *Message Exchange Patterns (MEP)*, identify placeholders for messages, that allow to model its sequence and cardinality, defining the order on

which parties send and receive messages. A set of messages sent and received among parties, optionally following Message Exchanged Patterns are referred as *message exchange*. They characterize a well defined part of a conversation. A *conversation* can be thus defined as a set of message exchanges among parties with the aim of fulfilling some goal. Every conversation need to rely on top of some communication facility referred as *communication network*.

2.1.3 Choreographies

A *choreography* describes the behavior of the answering service from the initiating party point of view [6]. It governs the message exchanges among parties in a conversation. A choreography as presented in this work is based on the Abstract State Machines (ASMs)² formalism. ASMs allow specifying the sequences of states the choreography goes through during its lifetime, together with its responses to events. In this sense the main building blocks of abstract state machines are up to some extent re-draw.

- A *state* is a situation during the lifetime of a choreography in which it waits for some event or satisfies some condition.
- Conditions are modeled as *booleanExpression*. Boolean expressions are expressions evaluated to “*true*” or “*false*” as in any programming or ontology language.
- An *action* represents the atomic task of sending a message to a party.
- *Events* represent occurrences of stimulus. They do not specify state transitions. Transitions among states are defined by *guarded transitions*.
- *Guarded transitions* allow modeling the relations between two states by means of *events*, *actions* and *guard conditions*.
- *Guard conditions* are rules that specify a target state.
- *Parts* permit to relate guarded transitions and message exchanges, defining the message exchange in terms of state transitions.

Finally, a choreography comprises a set of parts that define a conversation.

2.1.4 Logic Boxes

The atomic building blocks that permit to solve the mismatches among interacting parties are referred as *logic boxes*. A logic box facilitates the means to reorganize the content of documents, its mapping to messages, and the order and cardinality of messages, enabling the interoperation among heterogeneous message exchanges. Additionally, and depending on the type of box, the differences in the vocabulary used to describe the application domain can be overcome. Currently the specification defines five different types of logic boxes, namely: *refiner box*, *merge box*, *split box*, *select box*, *add box*. Logic boxes are grouped into *logic diagrams*. Logic diagrams permit to model the relation among the message exchange pattern followed by the initiating party, and the one used by the answering service. Logic diagrams are assimilated, for implementation purposes, to *correspondence tables*. A correspondence table is a

² Activities, entry actions and exit actions have been deliberately left out of the scope of the work, as they are not required for purpose.

logical structure, similar to routing tables, which defines relations among incoming and outgoing messages as a realization of a logic diagram. A number of logic diagrams defining a conversation are referred as *logic group*.

2.1.5 Ontologies

Ontologies define the semantics of the framework. They facilitate a formal and consensual vocabulary as data and information machine-processable semantics for the shared and common understanding of a domain that can be mediated for the understanding of interacting parties. *Domain ontologies* supply the general vocabulary to describe the application domain of parties. *Choreography ontologies* make available the terminology that describes the choreography of parties. In doing so they define the different entities (concepts) taking part in a choreography. *Ontology mappings* characterize the conceptual entity that allows to link similar ontological concepts and instances.

3 Overall Algorithm

The overall algorithm is derived from the interface definition of the conceptual model (see Figure 1). In a nutshell, it allows parties to request the creation of the operational models, correlate messages and finally put off-line the operational model once is no longer needed.

Given:

- the domain ontologies λ and λ'
- the choreography ontologies θ and θ'
- the input messages δ_i , and δ'_i

Then, the algorithm defines the structures and the logic for the overall functioning of the conceptual model. The pseudocode of the algorithm is detailed in Table 1.

Table 1. Pseudocode of *run* algorithm

```

1 run ( $\lambda, \lambda', \theta, \theta', \delta_i, \delta'_i$ ):void
2  $T \leftarrow \text{generateOperationalModel}(\lambda, \lambda', \theta, \theta')$ 
3 IF messageFromInitiatingParty THEN
    $\tau \leftarrow \text{correlateMessage}(T, \tau, \delta_i, \theta')$ 
4 IF messageFromAnsweringService THEN
    $\tau \leftarrow \text{correlateMessage}(T, \tau', \delta'_i, \theta)$ 
5 IF removeOperationalModel THEN
   removeOperationalModel( $\tau$ )

```

In a first step, the tuple of correspondence tables T is generated for the interoperation of the syntactic and semantic models of the initiating party and the answering service. Then, each time a message is received the defined correlating tables τ and τ' are used to correlate it according to the choreography of the addressee. Finally, and in case either party indicates so, the operational model is made off-line.

3.1 Operational Algorithms

The operational model defines a layer of abstraction among parties that allows overcoming their mismatches. In the following the details of the algorithm that permits to create such layer are depicted. Given:

- the domain ontologies λ and λ'
- the choreography ontologies θ and θ' that contain the description of the message exchanges $\gamma = \{ \delta, v \}$ and $\gamma' = \{ \delta', v' \}$, and the corresponding sufficient sets $\wedge\gamma = \{ \wedge\delta, \wedge v \}$ and $\wedge\gamma' = \{ \wedge\delta', \wedge v' \}$

as input parameters, and:

- the correspondence table T

as output parameter. Then, the algorithm calculates a correspondence table T that overcomes the heterogeneities of γ and γ' as defined in θ and θ' , in case their models are compatible. The pseudocode of the algorithm is detailed in Table 2.

Table 2. Pseudocode of the *generateLogicDiagram* algorithm

```

1 generateOperationalModel ( $\lambda, \lambda', \theta, \theta'$ ):T
2  $\chi \leftarrow \text{mapDomainOntologies}(\lambda, \lambda')$ 
3  $\Psi_\chi \leftarrow \text{mapChoreographyOntologies}(\theta, \theta', \chi)$ 
4 IF  $\Psi_\chi$  THEN
     $\text{compatible} \leftarrow \text{assertCompatibility}(\Psi_\chi, \gamma, \gamma')$ 
5  $\text{deadlocks} \leftarrow \text{preventDeadlocks}(\Psi_\chi, \gamma, \gamma')$ 
6 IF NOT  $\text{deadlocks}$  AND  $\text{compatible}$  THEN
     $T \leftarrow \text{generate}(\Psi_\chi, \delta, \delta')$ 
7 return  $T$ 

```

In a first step, the mapping between the domain ontologies λ and λ' is established, and thus, the function χ defined. Then, using χ , the mapping function Ψ_χ between the choreography ontologies θ and θ' , is calculated. Also, taking as input the ontology mapping Ψ_χ , and the structural and behavioral models of γ and γ' , compatibility is asserted. Later, the behavioral models of γ and γ' under the mapping Ψ_χ are analyzed to detect possible deadlocks. Finally, and in case both choreographies are compatible and contain no deadly embraces, the operational model T is calculated, and returned as result of the algorithm. If the choreographies are not compatible, the algorithm returns **NULL**.

4 Conclusion and future work

This paper has presented the architecture and overall algorithm of **SOPHIE**, an extensible conceptual framework realized as a SSOA that is especially suitable for supporting the fine grained interaction among services following different structural, behavioral or operational models. Section 2 presented an overview of the framework, the basic principles where sketched, the main parties identified and the core concepts presented. Finally, section 3 introduced the overall algorithm that drives the behavior of the service.

References

1. Arroyo, S.: "SOPHIE – Modeling Choreographies: Prospects for Application to Learning Objects", Learning Objects and Learning Designs 1(2). September 2005.
2. Sam Watkins, S., Arroyo, S., Duke, A., Richardson, M., Schreder, B., Wahler, A.: "D8.3: Prototype Platform Design" Case Study B2B in Telecommunications. DIP (Data, Information and Processes), 2005.
3. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: "Business Process Execution Language for Web Services", <ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf>, 2003.
4. Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Susan Struble S., Takacs-Nagy, P., Trickovic, I. and Zimek, S.: "Web Service Choreography Interface (WSCI) 1.0", <http://www.w3.org/TR/wsci/>, 2002.
5. Kavantzias, N., Burdett, D., Ritzinger, G.,: "Web Services Choreography Description Language Version 1.0", <http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/>, April 2004.
6. Roman, D., Scicluna, J., Feier, C., (eds.) Stollberg, M and Fensel, D.: "D14v0.1. Ontology-based Choreography and Orchestration of WSMO Services", <http://www.wsmo.org/TR/d14/v0.1/>, March, 2005.