A model-driven choreography conceptual framework

Sinuhe Arroyo a,⁎, Alistair Duke b, José-Manuel López-Cobo c, Miguel-Angel Sicilia d

a Digital Enterprise Research Institute, Innsbruck, Austria
b BT Next Generation Web Research, Ipswich, UK
c iSOCO, Pedro de Valdivia, 10, 28006, Madrid, Spain
d Computer Science Department, Polytechnic School, University of Alcalá, Madrid, Spain

Received 5 January 2006; received in revised form 24 April 2006; accepted 10 May 2006
Available online 13 July 2006

Abstract

A number of languages exist that try to model the external visible behavior of services. However, they constitute incomplete solutions, either because, they do not include proper support for semantics, they have a lack of technological independence, they mix internal and external aspects and finally, they do not provide consistent approaches or present ad-hoc ones to solve behavioral and structural heterogeneities, or worse, they mix both aspects resulting in confusing specifications. This paper describes SOPHIE, a conceptual framework that attempts to overcome these limitations. It allows the production of the intermediate structures that allow overcoming the heterogeneities between services from the semantic descriptions of the Message Exchange Patterns (MEPs) they follow.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Semantic Services; Choreography; MEP; SOA

1. Introduction

Services communicate with each other by exchanging messages, which allow them to make or to respond to requests. Upon the reception of a message, services react by executing some internal invisible processes, and possibly, responding with other messages. Choreography deals with describing such external visible behavior of services as message exchanges. In order to allow inter-operation among services exposing different visible behaviors, the means to map heterogeneous messages exchanges is required.

A number of approaches exist, such as BPEL4WS [3], WS-CDL [8], WSCI [4] or WSMO-Choreography [9], 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:

• a lack of technological independence (BPEL4WS, WS-CDL)

• the lack of a clear model that separates structural, behavioral and operational aspects (BPEL4WS, WS-CDL, WSCI or WSMO-Choreography)

• the lack of proper support for semantics (BPEL4WS, WSCI, WS-CDL)

• an ad-hoc approach to solve heterogeneity among message exchanges (BPEL4WS, WS-CDL, WSCI or WSMO-Choreography)

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. SOPHIE2 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

⁎ Corresponding author.
E-mail addresses: sinuhe.arroyo@deri.org (S. Arroyo), alistair.duke@bt.com (A. Duke), ozelina@isoco.com (J.-M. López-Cobo), msicilia@uah.es (M.-A. Sicilia).

0920-5489/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2006.05.004
Table 1
A first cut in classifying related languages

<table>
<thead>
<tr>
<th>Relation with communication framework</th>
<th>Layered model</th>
<th>Semantic support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tight</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Business Process Languages</td>
<td>Choreography Languages</td>
<td></td>
</tr>
<tr>
<td>Loose</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Choreography Languages</td>
<td>Semantic-driven choreography initiatives</td>
<td>SOPHIE</td>
</tr>
</tbody>
</table>

The idea of compatibility among message exchanges. It defines structural and behavioral compatibility as a pre-requisite to assert compatibility among message exchanges. Section 5 presents an example focused on the telecommunication field, which illustrates how the work is being applied. Finally, Section 6 summarizes the work presented and the future steps to improve SOPHIE.

2. Related work

In the following related work to the definition of a conceptual framework for choreography is concisely reviewed. In doing so, their core characteristics are presented, their drawbacks identified and the main ideas reused in this research are summarized.

Table 1 presents a preliminary classification based on a three dimension exam. The first dimension depicts the relation with the underlying communication framework, differentiating among tight and loose. The second one addresses the semantic support provided. Finally, the third one discriminates them depending on whether or not they follow a layered model. Based on these depiction four main categories of languages are distinguished:

- technologies with a tight relation to the underlying communication framework, lacking of a layered model and no support for semantics, such as BPEL4WS
- technologies with a tight relation to the underlying communication framework, that follow a layered model and no support for semantics, such as WSCDL
- technologies with a loose relation to the underlying communication framework, lacking of a layered model and no support for semantics, such as WSCI
- technologies with a loose relation to the underlying communication framework, with support for semantics but lacking of a layered model, such as WSMO-Choreography.

2.1. Business process languages

Business Process Languages provide the means to specify business processes and interaction protocols, representing the first attempt to model the visible behavior of services.

BPEL4WS is the main initiative classified in this group. It focuses on describing collaboration among processes through Web Service interfaces—orchestration—, rather than the sequence and cardinality of the messages exchanged—choreography—. Nevertheless, many of the concepts and ideas sketched in BPEL4WS have been adopted and improved in other choreography languages. BPEL4WS is characterized by a tight relation with the underlying communication framework, which seriously hampers its flexibility, a lack of support for semantics, which prevents the agile interoperating among Services and, a missing layered approach, which results in a confusing specification.

2.2. Choreography languages

Choreography languages deal with modeling the external visible behavior of Services as a number of message...
exchanges. The initiatives detailed in this group are WSCDL and WSCI.

WS-CDL is the latest attempt of the W3C [10] to define an XML language for the description of the common and complementary behavior of services from a global point of view. In WS-CDL the interaction follows an asymmetric nature biased towards the receiver rather than the sender, refereeing to the operation performed when information is received, but not the action(s) (or operations) leading to the sending of information [5]. Moreover, the relation among the specification and the use of MEPs, understood as a key element that allows solving the heterogeneity among message exchanges is not explicitly addressed. As well, even though it allows the recording of semantic descriptions, their purpose is not clear. Besides, it lacks of any support to correlate messages and solve heterogeneities. Finally, the lack of a layered model differentiating among structural, behavioral and operational aspects helps portraying a confusing view of the language. It tries at the same time to define a model and a XML syntax, which does not discriminate among models.

WSCl is also an XML-based language aiming at describing the message interfaces of services. WSCI is not longer under development, as the W3C replaced with WS-CDL. It does not count with any support for semantics, establishing a loose relation with the underlying communication framework. It does not incorporate the notion of MEP as a means to model behavior among interacting services. Additionally, it presents a global and centralized view of the choreography, which contradicts the decoupled nature of services. Moreover, the concept of transaction as presented seems to be more related to internal aspects than to internal ones. Finally, even though it sketches the concept of state to model behavior, the idea is not fully integrated, contributing to a confusing specification that lacks a clear separation of models.

2.3. Semantic-driven choreography initiatives

Only one initiative exists that tackles the choreography problem from a semantic perspective. The main advantages of this approach revolve around the dynamic generation of mappings among parties that allows them to interoperate in a more efficient and agile way.

WSMO-Choreography represents the first attempt to model choreographies from a semantic perspective. It focuses only on the behavioral aspects of the choreography, leaving aside structural and operational considerations. The behavioral model is based on the formalism presented by ASM from which it borrows an insufficient subset of concepts that can hardly model a complete choreography. Furthermore, it does not rely on the use of conversational patterns to define the order and cardinality of messages, thus complicating the mapping task among heterogeneous interaction styles. Additionally, it does not specify how the message exchange mismatches should be identified, mapped and solved. Finally, the specification is too closed over WSMO and ASMs, leaving no room to accommodate other formalisms, such as Petri nets for the behavioral model, or OWL as underlying semantic language.

3. Syntactic model

In the following section the syntax of the choreography engine is described. The grounding building blocks are detailed, their behavioral model is depicted and the means to allow interoperacion among different behavior patterns is described.

3.1. Structural model

The structural model deals with the provision of a reusable collection of entities following different levels of abstraction that facilitate the basis for the description of a conceptual model. Table 1, enumerates the entities that allow the structural model to be defined.

Conversations are the outer most entity of the structural model. They represent the logical entity that permits to group a set of related message exchanges among parties. Conversations are composed of a set of building blocks. Elements describe elementary units of data that define a name, a type and a value that build documents. Documents are complete, self-contained groups of elements. Documents are transmitted over the wire within messages. Messages characterize pieces of information that can be exchanged among parties. As messages are exchanged, a variety of recurrent scenarios can be played out as defined by Message Exchange Patterns (MEP). A MEP defines a minimal contract among parties. They allow the sequence and cardinality of messages to be modeled, defining the order in which parties send and receive messages. The constituent description is a part that depicts the behavior of the pattern. A set of messages sent and received among parties optionally following a Message Exchanged Pattern that account for a well defined part of a conversation, is referred as a message exchange. A conversation can thus be defined as a set of message exchanges among parties, optionally following message exchange patterns to model their behavior. Every conversation need to rely on top of some communication facility, referred to as a communication network.

3.2. Behavioral model

The behavioral model cares for the description of the dynamic interaction among the entities defined in the structural model. As presented in this work, the behavioral models are based on the formalism presented by Abstract State Machines (ASMs) [5]. Nevertheless, any other formalism such as Petri nets, temporal or transactional logic can be easily modeled and plugged-in to the behavioral model. In doing so, it makes use of the entities enumerated in Table 2.

---

3. Element types, belong to a limited set of types as defined by the standard XSD [7].
4. A number of patterns are defined in Ref. [2] which are further extended in Ref. [1] with the addition of patterns that try to cover a broader spectrum of message exchanges. They take under consideration some binding-specific information, such as faults, acknowledgements, timeouts and correlation among messages. Nevertheless this extension always keeps in mind that message exchange patterns identify a minimal contract between parties, and should contain only information that is relevant to the interacting parties.
3.3. Operational model

The operational model facilitates the means to allow the interoperation among different behavioral models. Table 3, enumerates the entities that allow an operational model to be defined. Logic boxes constitute the key entity of the operational model. The outer most entities of the operational model are logic groups.

3.3.1. Logic boxes

The atomic building blocks that permit to solve a number of mismatches among interacting parties are referred to as logic boxes. Given:

- a set of input elements \( \varphi = \{ \varphi_1, \ldots, \varphi_n \} \), and a set of output elements \( \varphi' = \{ \varphi'_1, \ldots, \varphi'_n \} \);
- the set of input documents \( \sigma = \{ \sigma_1, \ldots, \sigma_n \} \), and a set of output documents \( \sigma' = \{ \sigma'_1, \ldots, \sigma'_n \} \) containing each \( \sigma_i \) and \( \sigma'_i \) and a subset of elements \( \varphi \) and \( \varphi' \) respectively

A choreography represents the outer most entity in the behavioral model. It describes the behavior of the answering service from the initiating party’s point of view [8]. It governs the message exchanges among parties in a conversation.

States, actions and events and guard conditions represent the same concepts as defined by ASM. However, the scope of events and actions has been narrowed. Particularly, actions represent the atomic task of sending a message, and events cannot trigger a state transition, since they do not specify a target state, but just a booleanExpression. Additionally, activities, entry actions and exit actions have been deliberately left out of the scope of the work, as they are not required for our purposes. Finally, the concept of guarded transitions, parts and choreography have been added.

A guarded transition defines the relationship between states by means of events, guard conditions and actions. In a nutshell, a guarded transition defines events and conditions, which when satisfied, perform certain actions and trigger the state transition as defined in the guarded condition. Parts permit guarded transitions and message exchanges to be related, defining the message exchange in terms of state transitions according to the logic of the application. Finally, a choreography can be defined as a set of parts, which govern the message exchanges among parties in a conversation.

A choreography represents the outer most entity in the behavioral model. It describes the behavior of the answering service from the initiating party’s point of view [8]. It governs the message exchanges among parties in a conversation.

The outer most entities of the operational model are logic groups.

- the set of input messages \( \delta = \{ \delta_1, \ldots, \delta_n \} \), and a set of output messages \( \delta' = \{ \delta'_1, \ldots, \delta'_n \} \) containing each \( \delta_i \) and \( \delta'_i \) and some subset of documents \( \sigma \) and \( \sigma' \) respectively
- the input message exchanges patterns \( v \), and the output message exchange pattern \( v' \)
- the domain ontologies \( \lambda \) and \( \lambda' \), corresponding to the initiating party and answering service respectively
- the ontology mapping \( \chi \) defined as the function \( \chi: \lambda \rightarrow \lambda' \), where \( \chi \) permits the domain ontology \( \lambda \) to be mapped to the one described in \( \lambda' \) and vice versa.
- the choreography ontologies \( \theta \) and \( \theta' \), corresponding to the initiating party and answering service respectively
- the ontology mapping \( \Psi \) defined as the function \( \Psi_{\phi}: \theta \rightarrow \theta' \), where \( \Psi \) permits the choreography ontology \( \theta \) to be mapped to the one described in \( \theta' \) and vice versa under the domain ontology mapping \( \chi \)
- the initiating party \( \eta \) and the answering service \( \eta' \)
- the message exchanges \( \gamma = \{ \delta, v \} \) and \( \gamma' = \{ \delta', v' \} \), corresponding to \( \eta \) and the answering service \( \eta' \), respectively.

Then, a logic box \( \beta \) is defined as the function \( \beta_{\varphi, \psi}: \gamma \rightarrow \gamma' \), which maps the input elements \( \varphi \), documents \( \sigma \) or messages \( \delta \), sent by the initiating party \( \eta \), following the message exchange pattern \( v \), into a set of output message \( \delta' \), received by the answering service \( \eta' \), and following the message exchange pattern \( v' \), using the mapping defined by \( \Psi_{\chi} \).

Currently the work defines five different types of logic boxes, namely: refiner box, merge box, split box, select box and add box.

A refiner box is a logic box that permits to match the types and names of elements, documents, messages and any other entities according to the ontologies \( \theta \) and \( \theta' \) to be matched. Fig. 1 shows an example of a refiner box, which maps the
elements \( \varphi \), documents \( \sigma \), or messages \( \delta \), into a number of output messages \( \delta' \).

A **merge box** is a logic box as previously defined, which maps the input messages \( \delta \), sent by the initiating party \( \eta \), following the message exchange pattern \( \nu \), into a single output message \( \delta' \), received by the answering service \( \eta' \), and following the message exchange pattern \( \nu' \). Additionally, a merge box permits the types and names of elements, documents, messages and any other entities according to the ontologies \( \theta \) and \( \theta' \) to be matched. Fig. 2 shows a merge box, which merges the input messages \( \delta \) into an output message \( \delta' \).

A **split box** is a logic box as previously defined, which maps an input message \( \delta \), sent by the initiating party \( \eta \), following the message exchange pattern \( \nu \), into a number of output messages \( \delta' \), received by the answering service \( \eta' \), and following the message exchange pattern \( \nu' \). Additionally, a split box permits the types and names of elements, documents, messages and any other entities according to the ontologies \( \theta \) and \( \theta' \) to be matched. Fig. 3 shows a split box, which splits the input message \( \delta \) into a number of output messages \( \delta' \).

A **select box** is a logic box as previously defined, which selects any set of elements \( \varphi'' \) documents \( \sigma'' \) or messages \( \delta'' \), belonging to the input messages \( \delta \), sent by the initiating party \( \eta \), and following the message exchange pattern \( \nu \), transforming them into a number of output messages \( \delta' \), received by the answering service \( \eta' \), and following the message exchange pattern \( \nu' \). Additionally, a select box permits the types and names of elements, documents, messages and any other entities according to the ontologies \( \theta \) and \( \theta' \) to be matched. Fig. 4 shows a select box, which selects messages \( \delta_3 \) to message \( \delta_{n-1} \).

An **add box** is a logic box as previously defined, which adds a set of elements \( \varphi'' \), documents \( \sigma'' \) or messages \( \delta'' \), to a set of elements \( \varphi'' \), documents \( \sigma'' \) or messages \( \delta'' \), belonging to the input messages \( \delta \), sent by the initiating party \( \eta \), and following the message exchange pattern \( \nu \). It transforms them into a number of output message \( \delta' \), that will be received by the answering service \( \eta' \) according to the message exchange pattern \( \nu' \). Notice that an add box requires the use of a select box to complete its functionality. Fig. 5 shows and an add box, which adds a number of elements \( \varphi'' \), documents \( \sigma'' \) or messages \( \delta'' \), to the set of elements, documents \( \varphi'' \) or messages \( \sigma'' \), belonging to the input messages \( \delta \).

3.3.2. Logic diagrams and logic groups

A **logic diagram** is a set of interconnected logic boxes that model the relationship between the message exchange followed by the initiating party and the one used by the answering service. Logic diagrams have a well defined sequential control flow, as defined by **sequenceNumbers** which permits the message exchange among parties to be modeled.

A **logic group** is the conceptual entity that allows a number of logic diagrams that are part of the same conversation to be put together.

4. Semantic model

In the following section the semantics of the conceptual framework are described. Ontologies provide the terminology used to describe applications domains. In this work they are further used to describe the syntactic model. Table 4, distinguishes the different constituents of the semantic model.

4.1. Domain ontologies

**Domain ontologies** facilitate the general vocabulary to describe the application domain of the parties.

4.2. Choreography ontologies

Choreography ontologies provide the skeleton required to semantically describe the behavior of the syntactic model. In doing so, they define and allow instantiation of the entities of the structural and behavioral model, borrowing domain terminology from \( \lambda \) to \( \lambda' \), and the choreography model from \( \Lambda^6 \), with the aim of readily building the operational model as a result of a reasoning task. Such a reasoning task provides a set of mappings among the structures and behaviors of parties (MEPs).

4.3. Ontology mapping

An **ontology mapping** is a function that permits to link concepts and instances identified as similar. Where, **source** and **target** are

---

6. Describes the concepts of the syntactic model allowing the accommodation of different formalisms (ASMs, transactional logic, etc.) for each one of the models.
sets of domain and choreography ontologies. Furthermore, in the case of source ontology mappings are also allowed.

5. Compatibility

In order to give a precise characterization of the framework, it is necessary to provide the means to clearly assert interoperability among message exchanges. Thus, it is necessary to introduce and define the concept of compatibility among structures and behaviors. Since the main task of a conceptual framework is to provide the means that permit to generate the logic diagrams, to map the behavior and structure of heterogeneous parties, such compatibility plays a fundamental role. On one hand, it can provide the means to adapt the sequence and structure of the ones expected by the answering service. On the other hand, it provides the means to adapt the sequence and cardinality of the messages sent by the initiating party to the other hand, it provides the means to adapt the sequence and structure of the ones expected by the answering service. On the one hand, it can determine whether the content and structure of the message sent by the initiating party are compatible with the content and structure of the ones expected by the answering service. On the other hand, it provides the means to adapt the sequence and cardinality of the messages sent by the initiating party to the sequence and cardinality expected by the answering service. The following sections define the concepts of structural and behavioral compatibility between two message exchanges. The work focuses at the messageExchange level, as it constitutes the minimal coherent unit of reference.

5.1. Compatible entities

First, it is necessary to define when two entities are compatible (Table 5). Given the choreography ontology mapping \( \Psi \) as previously defined, two set of entities \( e = \{ e_1 \ldots e_n \} \), and \( e' = \{ e_1' \ldots e_n' \} \) are said to be compatible under \( \Psi \) if they match the relation:

\[ e \Theta \Psi e' \]

defined as follows:

\[ \{ e_1.\text{name} \ldots e_n.\text{name} \} \equiv \{ e_1'.\text{name} \ldots e_n'.\text{name} \} \]

The expression says that two sets of entities are compatible if there exists a relation in \( \Psi \) that links them as equivalent\(^7\). Notice that the definition does not impose any restriction about the cardinality of the sets, and thus an entity of \( e \) can correspond to a number of entities of \( e' \) and vice versa.

\(^7\) It is well out of the scope of the work to solve the ontology mapping and merging problem. In the remaining it will be assumed that such support is available.

Table 4

<table>
<thead>
<tr>
<th>Table 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities of the operational model</td>
</tr>
<tr>
<td>logicBox=[name, URI, type, inputMessages*, inputMep?, outputMep?, ontologyMapping?]</td>
</tr>
<tr>
<td>logicDiagram=[name, URI, inputMessageExchange?, outputMessageExchange?, logicBoxes*]</td>
</tr>
<tr>
<td>logicGroup=[name, URI, conversation?, logicDiagrams*]</td>
</tr>
</tbody>
</table>

The symbol “*” represents that there can exist zero or more instances of the attribute.

The symbol “?” represents zero or one instances of the attribute.

In the following, different expressions will be used to denote some type of compatibility relationship. For any two entities \( e \) and \( e' \), these expressions and symbols are interchangeable:

- \( e \Theta \Psi e' \)
- \( e \) matches \( e' \)
- \( e \) is linked to \( e' \)
- \( e \) is mapped to \( e' \)
- \( e \) is equivalent to \( e' \).

For example, given the set of elements \( \varphi = \{ \varphi_1 \ldots \varphi_2 \} \) and \( \varphi' = \{ \varphi_1' \ldots \varphi_4' \} \) under the mapping function \( \Psi_X \), which contains the relation \( \{ \varphi_1, \varphi_2 \} \approx \{ \varphi_1', \varphi_2' \} \) that maps “\( \varphi_1 \)” with “\( \varphi_1' \)” and “\( \varphi_2 \)” with “\( \varphi_2' \)”, \( e \Theta \Psi \varphi \varphi' \) holds.

5.2. Sufficient set

Given the set of entities \( e = \{ e_1 \ldots e_n \} \), \( e \) is said to be a sufficient set expressed:

\[ \land.e \in \{ e_1 \ldots e_n \} \]

if, it contains a minimal unduplicated meaningful set of entities, as defined by the owner party. A set of entities can define a number of sufficient sets.

For example, the set \( \varphi = \{ \varphi_1 \ldots \varphi_4 \} \) can define the sufficient sets \( \land \varphi = \{ \varphi_1 \ldots \varphi_3 \} \), or \( \land \varphi = \{ \} \), among others.

5.3. Contains relation

Given the sets of entities \( e = \{ e_1 \ldots e_n \} \) and \( e' = \{ e_1' \ldots e_m' \} \), the set of entities \( e' \) is contained in \( e \) under \( \Psi_X \), if they match the relation:

\[ e \supset e' \]

The expression states that each entity or set of entities of \( e \), is linked to another entity or sets of entities of \( e' \), under the mapping function \( \Psi_X \).

In the following, different expressions will be used to denote some type of sufficiency relationship. For any two

Table 5

<table>
<thead>
<tr>
<th>Table 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities of the semantic model</td>
</tr>
<tr>
<td>domainOntology=[name, URI]</td>
</tr>
<tr>
<td>choreographyOntology=[name, URI]</td>
</tr>
<tr>
<td>ontologyMapping=[name, URI, source, target]</td>
</tr>
</tbody>
</table>
entities $\hat{e}$ and $\hat{e}'$, these expressions and symbols are interchangeable:

- $e \Rightarrow \psi e'$
- $e$ is contained in $e'$

For example, given the set of elements $\varphi = \{\varphi_1, ... , \varphi_2\}$, and elements $\varphi' = \{\varphi'_1, ... , \varphi'_2\}$, where $\varphi \Theta \varphi'$ defines the same compatibility set as previously, then, $\varphi \Rightarrow \varphi \varphi'$ holds, since every entity or set of entities of $\varphi$ is mapped to another entity or set of entities of $\varphi'$ under $\Psi_X$.

### 5.4. Sufficient relation

Given the sufficient sets of entities $\hat{e} = \{e_1, ... , e_n\}$ and $\hat{e}' = \{e'_1, ... , e'_n\}$, the set of entities $\hat{e}'$ constitutes a sufficient set of entities with respect to $\hat{e}$ under $\Psi_X$ if they match the relation:

$\hat{e} \Pi \varphi \hat{e}'$

defined as follows:

$\hat{e} \Theta \varphi \hat{e}'$ and $\hat{e} \Rightarrow \psi \hat{e}'$

The expression states that $\hat{e}$ is compatible with $\hat{e}'$ under $\Psi_X$, and that $\hat{e}'$ encloses all the sufficient meaningful entities required by $\hat{e}$, according to a mapping function defined in $\Psi_X$.

In the following, different expressions will be used to denote some type of sufficiency relationship. For any two entities $\hat{e}$ and $\hat{e}'$, these expressions and symbols are interchangeable:

- $e \varnothing \psi e'$
- $e$ is a sufficient set for $\hat{e}'$
- $e$ suffices $\hat{e}'$

For example, if the initiating party defines as sufficient the elements $\hat{\varphi} = \{\varphi_1, ... , \varphi_2\}$, and the corresponding answering service the elements $\hat{\varphi}' = \{\varphi'_1, ... , \varphi'_2\}$, where $\varphi \Theta \varphi'$ characterizes the same compatibility set as $\varphi \Theta \varphi'$, then, the relation does not hold. The reason is that the sufficient set of the answering service is bigger than the sufficient set of the initiating party, written $\varphi \Rightarrow \psi \varphi'$, and thus, $\varphi \Pi \psi \varphi'$.

### 5.5. Covering relation

A set of entities $e = \{e_1, ... , e_n\}$ it is said to cover another set of entities $e' = \{e'_1, ... , e'_n\}$ under $\Psi_X$ if they match the relation:

$e \Omega \varphi e'$

defined as follows:

$\hat{e} \Pi \varphi \hat{e}'$

The expression states that the set of entities $\hat{e}$ covers the set of entities $\hat{e}'$, if and only if $\hat{e}'$ contains a sufficient set of compatible entities under $\Psi_X$, as required by $\hat{e}$. In the following, different expressions will be used to denote some types of covering relationship. For any two set of entities $e$ and $e'$, these expressions and symbols are interchangeable:

- $e \varnothing \psi e'$
- $e$ covers $e'$

For example, given the sets of elements $\varphi$ and $\varphi'$, that define respectively the sufficient sets $\hat{\varphi}$ and $\hat{\varphi}'$, and the mapping function $\Psi_X$ as previously, then $\varphi \Omega \varphi \varphi'$, since $\varphi \Pi \varphi \varphi'$ does not hold.

### 5.6. Structural compatibility

Given the structural models, $\delta$ and $\delta'$, belonging to the message exchanges $\gamma$ and $\gamma'$ that define the sufficient sets $\hat{\varphi}$ and $\hat{\varphi}'$, and the mapping function $\Psi_X$, as previously, then $\varphi \Omega \varphi \varphi'$, since $\varphi \Pi \varphi \varphi'$ does not hold.

The expression states that there is a sufficient set of elements of $\varphi$ belonging to the message exchange $\gamma$, which covers a sufficient set of elements of $\varphi'$ belonging to the message exchange $\gamma'$ under $\Psi_X$, as required by $\delta$, and vice versa.

For example, given the structural models $\delta$ and $\delta'$ and the corresponding sufficient sets $\hat{\delta} = \{\varphi_1, \varphi_2\}$ and $\hat{\delta}' = \{\varphi'_1, \varphi'_2\}$, as required by $\delta$, and vice versa.

---

**Fig. 6. Example of compatibility.**

**Fig. 7. Logic diagram.**

5.7. Behavioral compatibility

Elaborating on the previous definitions it is possible to establish when two message exchanges are behaviorally compatible. Given

- the set of actions α = {a1, ..., an} used to model the behavior v of the message exchange γ, and α′ = {a′1, ..., a′n} respectively for γ′
- the set of boolean expressions β = {b1, ..., bn} used to model the behavior v′ of the message exchange γ, and β′ = {b′1, ..., b′n} respectively for γ where, the pairs [α, β] or [α′, β′] are non-empty, thus guarantying that every stimulus has its counter part.

It is said that the behavior models v and v′, belonging to the message exchanges γ and γ′, with the sufficient sets ^v and ^v′, defined as ^v = ^{a1, b1} and ^v′ = ^{a′1, b′1}, are behaviorally compatible under ^Ψ if and only if, they are structurally compatible and match the relation:

δθψγv′

defined as follows:

Ωψv′ and ^Ωψ^v

The expression states that there is a sufficient set of actions α belonging to the message exchange γ, which covers a sufficient set of boolean expression β′ belonging to the message exchange γ′ under ^Ψ, and vice versa.

Notice that the core elements to model behavior are actions, guardConditions and events. Since guardConditions and events use booleanExpressions to signal stimulus, it is straightforward to establish a link among them, in order to determine behavioral compatibility. For example, given the behavioral models v and v′ and the corresponding sets of actions and boolean expressions α = {a1, ..., a′1}, β = {b1, ..., b′1}, α′ = {a′1, ..., a′n}, and β′ = {b′1, ..., b′n}, which define the sufficient sets ^α = {a1}, ^β = {b1}, ^α′ = {a′1}, and ^β′ = {b′1, ..., b′n} respectively. Then ^v = ^{a1, b1} and ^v′ = ^{a′1, b′1}, under the mapping function ^Ψ of the message exchange γ that defines {a1} = {b1}, meaning that α is compatible with {b1, b′1}, and vice versa expressed ^α θ ψ ^β and ^α′ θ ψ ^β′, then v θ ψ v′ holds, since ^v = Ωψv′ and ^v = Ωψ^v.

5.8. Overall compatibility

Finally, two message exchanges γ = {δ, v} and γ′ = {δ′, v′}, are said to be compatible under ^Ψ, if they match the relation:

γθψγ′v

defined as follows:

δθψδv′ and vθψv′

The expression states γ and γ′ are structurally and behaviorally compatible under ^Ψ. As a consequence of the compatibility, it can be stated that there exists a logic diagram ζ that allows mapping the structural and behavioral models of γ and γ′.

For example, given the message exchanges γ = {δ, v} and γ′ = {δ′, v′}, it holds that δ θ S ψ δ′ and v θ Bψ v′ and thus, can be stated that γ θ Bψ γ′.

6. Example

SOPHIE is currently being trialed as part of the DIP project8 B2B in Telecommunications case study, hosted by BT. SOPHIE has been applied to BT Wholesale’s B2B Gateway which allows BT’s ISP partners to integrate their Operation Support Systems with those of BT and, for example, carry out tests on BT’s network as part of their broadband assurance activities.

The example applies the conceptual model of SOPHIE to the broadband test interface in order to illustrate how a partner’s differing choreography could be integrated. The service provider uses the message exchange “tRequestOutOnly” following the message exchange pattern “outOnly” and the structural model “trStructure”, while BT makes use of the message exchange “acceptTestOnAck” that follows the message exchange pattern “inOnAck” and the structural model “testStructure”. Fig. 6 shows the sequence and cardinality of the messages of each MEP.

The service provider defines the structural model for compatibility as the entities:

• α = {tName, testData}, ^α = {tName, testData} and the message exchange pattern “outOnly”, to model the external behavior as the actions and boolean expressions:

• β = { }, ^β = { } while BT uses the message exchange defines the structural model for compatibility as the entities:

• ^φ = {tName, ack, testData}, ^φ = {tName, testData} and the message exchange pattern inOnAck, to model the external behavior, as the following actions and boolean expressions:

• α = {sendAck} and the sufficient sets ^α = { } 
• β = {testNameReceived, testDataReceived} and the sufficient sets ^β = {testNameReceived, testDataReceived}

Both parties make available their domain and choreography ontologies, λ, λ′, θ and θ′ respectively. The mapping function ^Ψ defines the following compatible entities:

• ^φ, ^φ: {tName, testData} ≈ {tName, testData}
• ^α, ^α, ^β and ^β: {sendRequest} ≈ {testNameReceived, testDataReceived}

It can be stated that:

• there exists a covering relation among the sufficient sets of the structural models

8 http://dip.semanticweb.org/.
there exists a covering relation among the sufficient sets of the behavioral models

which implies that the structural and behavioral entities of “tRequestOutOnly” are contained in the structural and behavioral entities of “acceptTestOnAck”, and vice versa under the mapping function Ψύ. Then, both message exchanges are structurally compatible, expressed \( trStructure ∈ ΨύtestStructure \). Additionally, they are behaviorally compatible, expressed \( outOnly ∈ Ψύ\text{PinAck} \). Thus, they are compatible, expressed \( tRequestOutOnly ∈ ΨύacceptTestOnAck \).

Since they are compatible, it can be stated that there exists a logic diagram ζ which allows mapping both message exchanges. Fig. 7, shows such logic diagram, where the message “tRequest” containing the document “Info” and the elements of φ is converted into the messages “testName” and “testData”, containing each one a document and enclosing the elements “testName” and “testData” respectively. Additionally, the action of sending the message “ack” is cancelled.

7. Conclusion and future work

This paper has presented SOPHIE, an extensible conceptual framework that is especially suitable for supporting the fine grained interaction among services following different structural, behavioral or operational models. SOPHIE elaborates on current existing initiatives trying to overcome their limitations with the addition of a layered syntactical model, support for semantics and technological independence. Also, the use of MEPs as the core building block to describe the skeleton of messages has been introduced. In addition, the minimal entities that can determine when two message exchanges are compatible have been identified. Moreover, the resources to assert such compatibility have been provided. Finally, the concepts depicted on the framework had been applied to the Assurance Integration Use case as part of the DIP project Ref. [11]. In the future, this work will be extended with the inclusion of support for Petri nets as part of behavioral model. The algorithms that allow the operational model to be produced will be defined and the whole framework will be implemented as a realization of a SOA.

References


Sinuhé Arroyo is a Scientific Project Manager and Semantic Web specialist in the Information Engineering Research Unit at University of Alcalá de Henares. He is also the Scientific and Technical Manager of the EU STREP funded project LUISA and member of the European Association for Semantic Web Education. Between 2002 and 2005 he was a researcher at the Digital Enterprise Research Institute at the University of Innsbruck, Austria, where he conducted extensive research in the field of Semantic Web Services Choreography. Additionally to his research role Sinuhé was responsible for managing various EU-funded projects. He was also very active in EU-framework 6, participating and leading the elaboration of various accepted proposals. Previously Sinuhé was Project leader at iSOCO in Madrid, Spain, responsible for the architecture and development of various applications in the field of B2B applying Semantic Web technology. Sinuhé obtained his M. Sc. in Computer Science at Universidad de Malaga (UMA), Malaga, Spain, in 2000. He also holds a B.Sc. in Software Engineering from the University Complutense de Madrid (UCM), Madrid, Spain, in 1997. Sinuhé authored various book chapters and acted as referee for several Semantic Web related conferences and journals. He has published numerous peer-reviewed papers in international conferences and impact journals.

Alistair Duke graduated from Aston University, UK, with an MEng in Electronic Systems Engineering and from Loughborough University, UK with a PhD in Collaboration Systems for Concurrent Engineering. He is currently a Principal Researcher with the Next Generation Web Research Group of BT. His primary interest is the Semantic Web and its application to the fields of Knowledge Management and Web Services. He currently works on the EU DIP project where his focus is on applying Semantic Web Services to the Telecommunications industry and on the EU SEKT project where leads the Knowledge Access workpackage.

Ozelin López was born in 1972 in Granada (Spain). He holds a degree in Computer Engineering from the University of Málaga and currently is taking his PhD in the field of Semantic Web Services. He has been working since 1999 in Research projects, in two companies. He worked at Atos Origin, as technical manager of the Software & Services Unit of the STREAM department being responsible for the management of the Semantic Web Services related project Infrawebs, TAO and LUISA and collaborating in other Service-Oriented projects like SeCSE and SERENITY. Ozelin was very active in EU FP6 acquiring a relevant number of projects for Atos Origin. Currently he works at iSOCO as Research Manager where he is managing the Semantic Web Services cluster and collaborating in Semantic Web and Semantic Web Services projects like DIP, SEKT or SUPER.
Dr. Miguel A. Sicilia obtained a University degree in Computer Science from the Pontifical University of Salamanca in Madrid, Spain (1996), and a PhD from Carlos III University in Madrid, Spain (2002). In 1997 he joined an object-technology consulting firm, after enjoying a research grant at the Instituto de Automática Industrial (Spanish Research Council). From 1997 to 1999 he worked as assistant professor at the Pontifical University, after which he joined the Computer Science Department of the Carlos III University in Madrid as a lecturer, working simultaneously as a software architect in e-commerce consulting firms, and being a member of the development team of a personalization engine. Since 2002 to October 2003, he worked as a full-time lecturer at Carlos III University working actively in the area of adaptive hypermedia. He currently works as a full-time professor at the Computer Science Department, University of Alcalá (Madrid). His research interests are primarily adaptive hypermedia, learning technology and human–computer interaction, with special focus on the role of uncertainty and imprecision handling techniques on those fields. He is also editor of several journals in those areas as well as in several scientific journals related to the field of metadata, Semantic Web and e-learning.