Chapter 6

BASIC CONCEPTS IN CHOREOGRAPHY

Sinuhe Arroyo
Digital Enterprise Research Institute (DERI), Innsbruck, Austria – sinuhe.arroyo@deri.at

1. INTRODUCTION

Services constitute an emerging paradigm for the design of distributed software systems. Nonetheless, interoperability is a factor determining the adoption of innovation in business environments (DIP) so that interoperability must be carefully addressed as a critical element in SOA (Service Oriented Architecture) technology. Services need to interoperate with each other in order to realize the purposes of the software system they define 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. Due to the heterogeneous technological and syntactic nature of services realizing semantic web processes, communication requirements become more complex, clearly defining a balance among interoperation and decoupling.

The concepts and ideas that underlie the so-called *Semantic Web* (Berners-Lee, Hendler et al. 2001) appear as a candidate solution for such complex compatibility problems (Brogi, Canal et al. 2004). Notably, formal ontology based on description logics (Baader, Calvanese et al. 2003) provides an appropriate formalism to deal with compatibility problems.

In the context of providing support for *choreography* (i.e. the modeling of external, visible behavior of service interactions), a semantic layer could be supposed to provide the required convertibility between divergent specifications by the specification in machine-processable form of the message exchanging patterns (MEP). Scalable and reliable service communication and integration beyond simple interchanges requires

interoperable choreography as an essential service for business collaboration (Jung, Hur et al. 2004). This makes semantic compatibility between interchanges an important research objective, which has recently been stated in formal terms (Brogi, Canal et al. 2004).

The idea of overcoming the heterogeneity among messages using such semantic service-based mediation layer as choreography service(s) is thought to have a lot of potential (Watkins, Arroyo et al. 2005; Zaremba, Moran 2004). On the one hand, as the number of accessible services increases, so does the number of structural and behavioural styles, thus requiring the use of some intermediate layer to overcome heterogeneity. On the other hand, the development of new applications and integration of existing ones can be greatly decreased, as off-the-shelf services can be readily used to build bigger and more complex software systems minimizing integration efforts. In a nutshell, the design of modern applications requires a compromise among interoperation and decoupling that is sometimes hard to realize due to the heterogeneous nature of services. If services communicate by exchanging message, a choreography engine is a good mediation layer that could speed up the interoperation and development of new and existing software functionality.

A number of approaches exist, such as BPEL4WS (Andrews, Curbera et al. 2003), WS-CDL (Kavantzas, Burdett et al. 2004), WSCI (Zaremba, Moran 2004) or WSMO – Choreography (Roman, Scicluna, et al. 2005), which 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[1])
- an ad-hoc approach to solve heterogeneity among message exchanges (BPEL4WS, WS-CDL, WSCI or WSMO-Choreography)
- a central vs. decouple approach to model choreographies (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.

[1] It supports the recording of semantics, but it does not use them at all.

In the following, the main ideas and concepts behind choreography are depicted. Section 2 carefully depicts related approaches and initiatives dealing with choreographies categorizing them and reviewing their main contributions and lacks. Section 3 details the main driving principles required to model and allow interoperation among heterogeneous message exchanges. Section 4, present the new challenges in choreography. Section 5 provides a detailed description of SOPHIE an initiative that aims to overcome the limitations of existing approaches. Section 6, exemplifies the concepts and ideas sketched so far by means of a use case centered in the telecommunications field. Finally, Section 7 draws the conclusion of the chapter.

2. LITERATURE REVIEW

In the following different technologies that are related to the definition of a conceptual framework for choreography are concisely reviewed. In doing so, their core characteristics are presented and main drawbacks identified.

Table 6-1 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 WS-CDL
- 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

Table 6-1. A first cut in classifying related languages

		layered model		
		no		***yes***
relation with communication framework	***tight***	Business Process Languages	Choreography Languages	
	loose	Choreography Languages	Semantic-driven choreography initiatives	SOPHIE
		no	***yes***	
		semantic support		

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 interoperation among Services and, a missing layered approach, which results in a confusing specification.

BPEL4WS. The Business Process Execution Language for Web Services (BPEL4WS) (Andrews, Curbera et al. 2003) is a model and a grammar for describing business work flow logic. In doing so, it represents interactions between processes and its partners through Web Service interfaces. BPEL4WS allows the creation of *abstract processes* that *describe business protocols* –public visible behavior–, as well as *executable processes* –private behavior–, that can be compiled into runtime scripts (Barros, Dumas et al. 2005).

The specification makes use of the following concepts. Business partners define groups of *partner links* that allow them to establish a number of conversational relations. A partner link models the services with which a business process interacts. Partner links are characterized by *partner link types*. A partner link type represents the conversational relationship between

two services by defining the roles of each one of them and the port types that will be receiving each others messages. Correlation among messages within a conversation is provided by means of *correlation sets*. Correlation sets provide a declarative mechanism to define correlated groups of operations. Additionally, *variables* facilitate the means to hold messages that have been received or will be sent to partners, which constitute the state of a business process. The values of compatible variables can be copied among them by means of assignments.

BPEL4WS differentiates among two types of *activities*. *Basic activities* represent the invocation of an operation on a service as a synchronous or asynchronous request or response. Basic activities can be associated with other basic activities that act as its compensation action. *Structured activities* prescribe the order in which a collection of activities takes place, permitting to describe business processes by composing basic activities. The context where activities behave is called *scope*. A scope allows defining correlation sets and a number of *event handlers* for compensation, alarms and fault, among others. Event handlers define a set of actions that are invoked concurrently if a particular event occurs. A process definition is then made of one activity, a series of partners, some specific correlation sets, and the definition a number of handlers.

In practice, BPEL4WS focuses on the description of collaborative processes –orchestration–, rather than in the detailed description of the external visible behavior –choreography–. Also, it presents a tight relation with the underlying communication framework, which prevents the use of any technology other than WSDL and SOAP. Furthermore, even though roles might not hold through out the interaction, partners are tight to roles in conversations. Additionally, it lacks of a layered model and support for semantics. Finally, the use of variables and scopes has more to do with the private behavior of the process than with the external visible one, presenting, when assimilated to choreographies, a non-desirable centralized approach that goes against the decoupled nature of services.

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 WS-CDL and WSCI.

WS-CDL is the latest attempt of the W3C (WWW) to define an XML language for the description of the common and complementary behavior of services from a global point of view. Like in the case of BPEL4WS, WS-CDL has a tight relation to the underlying communication framework, and

lacks of layered model. Additionally, it allows the recording of semantic description, even though the purpose of such feature is not clear.

WSCI 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. WSCI does not count with any support for semantics, nor follows a layered model, establishing a loose relation with the underlying communication framework.

WS-CDL. The Web Service Choreography Description Language (WS-CDL) (Kavantzas, Burdett et al. 2004) is an XML-based language for the description of the observable behavior of Web Services defined under the auspices of the W3C. WS-CDL permits defining, from a global and common point of view, multiparty contracts, which describe the visible behavior of Web Services as a number of ordered message exchanges.

The specification makes use of the following concepts. *Participants* represent the consumers and producers of information. They identify a set of related roles. *Roles* enumerate the observable behavior of a participant with respect to another one. The association of two roles to fulfill a concrete purpose is called *relationship*. A relationship represents the possible ways in which two roles can interact. *Channels* specify where and how to exchange information, they define the links between WS-CDL choreographies and the operations described in the interfaces of services. *Variables* contain information about the objects partaking in the choreography that describe the information exchanged during an interaction.

Choreographies are described in WS-CDL *documents*. WS-CDL documents, describe, from a global point of view the rules agreed among participants that govern the message exchange. They are encapsulated in *packages*. Packages enclose information that is common to all the choreographies it contains. Additionally, packages enclose *activities*. Activities can be conducted by participants. The specification details three types of activities namely, *Ordering Structure*, *WorkUnit* and *Basic* activities. Ordering Structure activities are block-structure activities that enclose a number of sub-activities. WorkUnit activities describe the conditional and possibly repeated execution of an activity. Work activities are actual work performers.

To conclude, a WS-CDL choreography can specify one exception block and one finalizer block, which are respectively activated when an exception occurs or when the choreography has completed successfully.

In practice, the explicit association of roles to participants as modeled in relationships is a too constraining way of representing the interaction among services. Such relation should be transparent to the language and dependent only on the particular part of the message exchange conducted. Also, the use

of channels further hampers flexibility, as it adds a new restriction to the interaction, which should be overcome by the addressing mechanism of the message exchange. In this direction, the specification is too tight to a particular technology. Furthermore, the use of variables to describe the context of the interaction and activities to model the functionality of parties helps to present a centralized approach which goes against the natural decoupling that services should follow. The state of each party should be private and transparent to other parties. Additionally, 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 (Barros, Dumas et al. 2005). 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 discriminates among aspects.

WSCI. The Web Services Choreography Interface (WSCI) (Arkin, Askary et al. 2002) is an XML-based interface description language co-developed by a number of industrial partners. WSCI describes the flow of messages exchanged by Web Services in terms of dependencies among them, featuring sequencing rules, correlation, exception handling and transactions.

Interfaces describe how services are perceived to behave from a temporal and logical point of view within a *message exchange*. A service might have multiple interfaces for a given message exchange. A set of message exchange define a *conversation*. Conversations make use of *message correlation* in order to describe its structure and which *properties* must be exchanged to maintain the semantic consistency of a conversation. *Properties* are equivalent to variables and allow referencing a value or representing an abstraction for a message received. Such abstractions are conceptualized as *processes*. *Processes* are labeled with a name and represent a portion of behavior, such as receiving a message or calling a process. *Activities* represent the basic unit of behavior of a service. They are classified into *atomic* and *complex*. Atomic activities constitute a basic unit of behavior such as sending or receiving a message o waiting for an amount of time. Complex activities are recursively composed of other activities which defines a specific type of choreography for the activities it encloses. Ultimately complex activities are composed of *actions*. Activities are

executed within the environment provided by a *context.* Context might be associated with *transactions* to describe from the interface perspective the transactional properties of a number of activities. In addition *exceptions* allow to model models exceptional behavior of a service in a conversation

WSCI does not take under consideration MEPs as a means to describe the behavior of parties and cornerstone to solve heterogeneities. Also, it lacks of any support for semantics. Furthermore, it leaves aside issues such as QoS and security. 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 of a clear separation of models.

2.3 Semantic-driven Choreography Initiatives

Currently, 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 does not make any assumption about the underlying communication platform. The main draw back of WSMO Choreography is the lack of separation of models.

WSMO Choreography. The WSMO Choreography (Roman, Scicluna, et al. 2005) is an ontology-based approach that allows describing the behavior of services from the user point of view. WSMO-choreography is based on Abstract State Machines (ASMs), from which it inherits the core principles, namely, *state-based*, *state by and algebra* and *guarded transition rules.* The main building blocks of WSMO choreography are thus *states* and *guarded transitions.* States are described by a link to an instance of a WSMO ontology. Guarded transitions define transition rules that express changes of states by changing the set of instances of the ontology.

WSMO-chorography focuses only on the behavioral aspects of the choreography, leaving aside structural and operational considerations. Also, 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. DRIVING PRINCIPLES

When designing choreographies a number of driving principles need to be taken under consideration. In the following, such principles are enumerated and briefly discussed.

Conceptual framework

The first step in analyzing choreographies is to identify the different entities that partake in the interaction. A choreography service defines terms and roles for these entities.

Separation of models

The definition of a choreography service requires that the models that build it are clearly differentiated. As a result, a modular framework should be designed, where different formalism can be readily added, extended or replaced, in order to allocate the most suitable one, depending on the target application and application domain.

Semantic-driven and mediation

Due to the natural heterogeneity of the open environment where services reside, the interoperation of heterogeneous message exchanges requires the production of intermediate structures – mediators – that allow overcoming mismatches. By semantically describing the different entities that characterize the choreography service, such structures can be produced as a result of a mediation task. In doing so, mediation allows any party to speak with every other (Fensel, Bussler 2002), facilitating an intermediate layer that provides a generalized solution to resolve communication mismatches.

Technological independence

The design of a fully extensible choreography service should not make any assumptions about underlying technologies. In particular, the details regarding transport and communication frameworks should be left aside. In doing so, a choreography service should rely on such underlying technologies, defining a clear border, which allows separating the particular communication details from the conceptual model used. In addition, as new

ontological languages based on different logical formalisms are developed, independence from existing and emerging specifications should be obliged. In consequence the conceptual model is driven by the semantic description of its constituent entities, not making any constraint or assumption on the ontological language used to model such descriptions.

Separation of internal and external behavior

Choreographies deal with the externally visible behavior of parties. The internal details should be clearly separated from the external ones, allowing its independent definition. Deeply in this direction, the description of collaborative process is out of the scope of this work, same as orchestration in general.

Global view vs. decentralized approach

Many traditional models call for centralized approaches where the interaction among parties is controlled by a unique point. In contrast, the nature of ubiquitous systems is decentralized. While a decentralized approach is preferred due to its flexibility to adapt to different application domains, eventually, a global point of view is chosen to control the message exchange. A choreography service should take both approaches under consideration, allowing parties to choose the most suitable one at any time.

Pattern-driven

Particular types of interactions among services, such as, negotiation or interactive information gathering, follow well established and researched Message Exchange Patterns (MEPs). The choreography service should allow the usage of such conversational protocols as main building block that can be used to overcome heterogeneities among services.

SOA-based

A realization of all this basic principles, especially regarding semantic-driveness and technological independence, the choreography service should be realized as a SOA architecture with support for semantics.

4. GOALS: SEPARATION OF MODELS AND MEDIATION

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 interoperation among services exposing different visible behaviors, the means to map heterogeneous messages exchanges is required. The problem is not trivial. On the one hand existing initiatives that tackle the choreography problem defined much interleaved conceptualizations, lacking of a clear separation of models that mix structural, behavioral and operational aspects. Furthermore, they portray choreographies from a global point of view, while services are characterized by their decentralized nature. On the other hand, such the approaches to solve heterogeneities are very much ad-hoc ones. Initiatives to overcome mismatches based on semantic descriptions should be envisioned as a core building block that provides the means to readily overcome heterogeneity by means of mediators. Still, the current state of the technology do not suffice for the degree of automation require, imposing human supported mediation techniques, which hamper the dynamism of the task.

The separation of models and support for semantic mediation are at the hearth of the challenges discussed above. Both characteristics are complementary and required in the design of a choreographies framework.

4.1 Separation of Models

The separation of models enables the definition of a flexible conceptual framework where the different abstract pieces are well decoupled from each other.

These are the most important requirements considered by the layered model:

- *Syntactic vs. Semantic*: Syntax and semantics should be clearly distinguished as in (Tsalgatidou, Pilioura 2002). While syntactic aspects identify core entities and interfaces, semantics cares for adding the machine understandable descriptions that allow the dynamic interoperation among the entities described in the syntax.
- *Separation of aspects*: Structural, behavioral and operational aspects should be clearly separated within the syntactic model.
  - *Structural* aspects deal with the provision of a reusable collection of entities following different levels of abstraction that provide the basis for the description of a conceptual model
  - *Behavioral* aspects care for the description of the dynamic interaction among the entities defined in the structural model
  - *Operational* aspects facilitate the means to allow interoperation among different operational models

A clear separation of aspects facilitates the addition, replacement and modification of the underlying paradigms without imposing the need to redesign the overall conceptual model and affecting the remaining aspects.

4.2 Mediation

Mediation refers to the ability to solve heterogeneities among heterogeneous entities. It allows parties to exchange messages, documents and the data they contain, disregard of the vocabulary and behavioral model used. Mediation by means of *mediators*, facilitates a generalized solution to resolve communication mismatches among heterogeneous parties.

Mediation is applied at two different levels:

- *Data and domain knowledge.* The interoperation of parties will require the mediation of data types and domain knowledge during the message exchange. Depending on the domain to which parties belong to different data types and domain knowledge might be used to encapsulate data and its meaning.
- *Message Exchange Patterns.* Parties follow well-defined heterogeneous message exchanges that model their external behavior. Patterns represent units of reference that allow formalizing such behavior.

By semantically describing data, domain knowledge and message exchange patterns, mappings that overcome the differences among heterogeneous behaviors and structures can be readily produced.

Elaborating on the previous statements, it can be easily derived that separation of models and mediation pose different degrees of complexity. The engineering of software systems is already driven by a differentiation of models, as a means to provide scalable and flexible systems. Likewise, the Semantic Web is trying to put in place the formalisms required to agilely solve heterogeneity at the ontological level. Additionally, there should not be any doubt about the close relation among separation of models and mediation, where later requires of the former. In fact the successful mediation necessitates a clear depiction and semantic description of the syntactical aspects.

5. SOPHIE: SEMANTIC WEB SERVICES CHOREOGRAPHI ENGINE

SOPHIE (Arroyo 2006; Arroyo, López et al. 2005; Arroyo, Duke 2005; Arroyo, Sicilia et al. 2005; Arroyo, Kummenacher 2006; Arroyo, Sicilia 2005) is a conceptual framework and architecture for a choreography engine

or service realized as a Semantic Service Oriented Architecture (SSOA). SOPHIE is especially suitable for supporting the fine grained interaction among services following different structural or behavioral models following precisely the principles detailed in Section 3. It elaborates on existing initiatives (Arkin, Askary et al. 2002; Andrews, Curbera et al. 2003; Kavantzas, Burdett et al. 2004; Roman, Scicluna, et al. 2005) trying to overcome their limitations with the addition of a layered syntactical model, support for semantics, technological independence as 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). Furthermore, it relies on the use of MEPs as the core building block to semantically describe the skeleton of message exchanges.

5.1 Overall Architecture

Services that use SOPHIE fall into two categories, namely, *initiating parties* and *answering services*. Both parties produce and consume messages. Additionally, initiating parties indicate the choreography engine by means of any of its constituents *correlating services* that the infrastructure for the interoperation of heterogeneous message exchanges should be established.

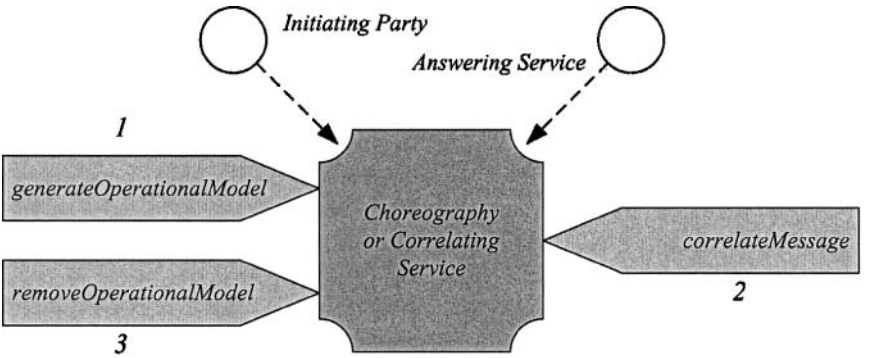

Figure 6-1. Choreography engine

Figure 6-1 shows a high level architecture of the conceptual framework realized as a single correlating service[2]. Informally, initiating parties indicate

[2] Notice that the choreography service itself is readily assimilated to one or more correlating service, in case a decentralized approach is preferred.

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*".

5.2 Models

SOPHIE makes a clear distinction between semantic and syntactic models. The semantic model details the support for semantics, while the latter details the syntax of the framework. The syntactic model depicts three different complementary models: 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 and, 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. The work presented here defines the behavioral model as Abstract State Machines (ASMs). Petri nets, temporal logic or transaction logic can however also be used instead of ASMs and easily plugged in. The semantic model is currently based on WSML. Nonetheless, the design allows to easily extending the grammar and ontology of SOPHIE to accommodate any other ontology language.

Structural model

A *conversation* represents the logical entity that permits a set of related message exchanges among parties to be grouped together. Conversations are composed of a set of building blocks. *Elements* represent units 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 sequence and cardinality to be modeled, defining the order in which parties send and receive messages. A set of messages sent and received among parties optionally following a MEP that account for a well defined part of a conversation, are referred to as a *message exchange*. A conversation can be thus defined as a set of message exchanges among

parties with the aim of fulfilling some goal. Every conversation is carried out over a communication facility, referred to as a communication network by parties. The specification differentiates among two type of parties, *initiating parties* and *answering services*. Both parties produce and consume messages, and additionally initiating parties take care of starting the message exchange.

Behavioural model

A *choreography* describes the behavior of the answering service from the initiating party's point of view (Roman, Scicluna, et al. 2005). It governs the message exchanges among parties in a conversation. Normally ASMs or Petri Nets are used to model the sequences of states the choreography goes through during its lifetime, together with its responses to events.

Operational model

The atomic building blocks that permit a number of mismatches among interacting parties to be resolved are *logic boxes*. A logic box facilitates the reorganization of the content of documents, its mapping to messages, and the order and cardinality of messages, thus enabling the interoperation among parties following different message exchange patterns. 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*.

Semantic Model

Ontologies define the semantics of the engine. They provide a vocabulary that can be mediated for the understanding of interacting parties. *Domain ontologies* facilitate the general vocabulary to describe the application domain of the answering service and the initiating party. The *choreography ontology model* provides the conceptual framework and vocabulary required to describe choreographies. In doing so, it defines and allows reusing concepts for the definition of the structural and behavioral models of each party's choreography. Finally, *ontology mappings* put in place the mechanisms to link similar ontological concepts and instances and readily produce the operational model as a result of a reasoning task.

5.3 Interface Functions

SOPHIE exports the functions listed in Table 6-1.

Table 6-1. Interface functions of SOPHIE

generateOperationalModel (URI *choreographyOntology*$_i$, URI *domainOntology*$_i$, URI *choreographyOntology*$_a$, URI *domainOntology*$_i$): *operationalModel*
removeOperationalModel (URI *operationalModel*, URI *message*, URI *choreographyOntology*): *void*
correlateMessage (URI *operationalModel*, URI *message*): *operationalModel*

Intuitively, **generateOperationalModel** (*choreographyOntology*$_i$, *domainOntology*$_i$, *choreographyOntology*$_a$, *domainOntology*$_i$) permits an initiating party following the choreography "*choreography*$_i$" to indicate the choreography service that wishes to establish a conversation with an answering service following the choreography "*choreography*$_a$". As a consequence, the operational model that will allow them to send and receive messages according to the different message exchange patterns they are using needs to be built. Additionally, the domain ontologies "*domainOntology*$_i$", and "*domainOntology*$_a$" that describe the domain of each one of the parties are also supplied. They allow the mediation of the models used by the interacting parties. As a result, the choreography service will generate and return the identifier of the operational model that will govern the conversation.

removeOperationalModel (*operationalModel, message, choreographyOntology*) declares that the operational model "*operationalModel*" is not any longer required by the parties taking part in the conversation, and thus can be put off line.

The function **correlateMessage** (*operationalModel, message*) states that the initiating party or the answering service desire their counterpart to receive the message "*message*", which should go through the opertinal model "*operationalModel*" in order to be adapted to the requirements of the receiving choreography.

Initiating parties and answering services are not specified as part of the parameters of the interface functions. It is important to do so because we might want to allow parties to send messages to the framework on behalf of others. In any case, the parameter *message*, specifies the sender and final

receiver of a concrete message. Also, for simplicity the concept of correlating party has been omitted in this section.

6. CASE STUDY

SOPHIE is currently being trialed as part of the DIP (DIP) project 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 B2B Gateway currently uses the Business Process Specifications of ebXML to represent the required choreography.

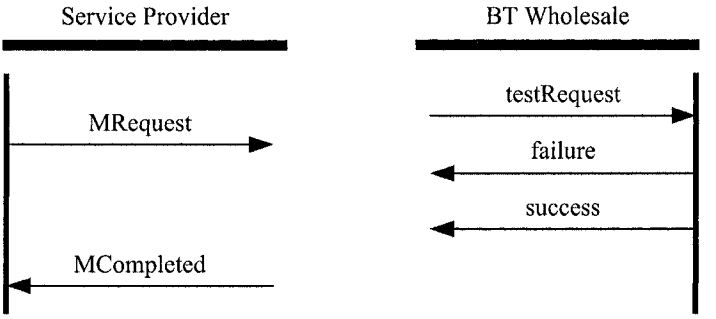

Figure 6-2. Request-Response and In-Multi-Out MEPs

The example applies the operational model of SOPHIE to the broadband test interface in order to illustrate how a partner's differing choreography could be integrated. Figure 6-2 shows the choreography of interacting parties following different MEPs as a realization of the same semantic web process. The Service Provider uses the message exchange *tPontTestRequest* following the MEP *request-response* while BT Wholesale makes use of the message exchange *eCoTestRequest* following the *In-Multi-Out* one. More concretely, the Service provider starts the message exchange with the *MRequest* message, while BT Wholesale expects the message *testRequest.* Additionally, BT provides two different response message (*failure* and *success*) indicating whether the test was accepted or rejected, and if accepted, the result of the test, while the *Service Provider* awaits the reception of a single message named *MCompleted* accounting for both of them.

Since both message exchanges are compatible, there exists a logic diagram which allows mapping the content, sequence and cardinality of both message exchanges.

Taking as input ontological mapping Ψ_X, that links the concepts that are similar in both choreography models the operational model that overcomes the heterogeneity ob both MEPs can be overcome a result of a reasoning task. Figure 6-3 details such model.

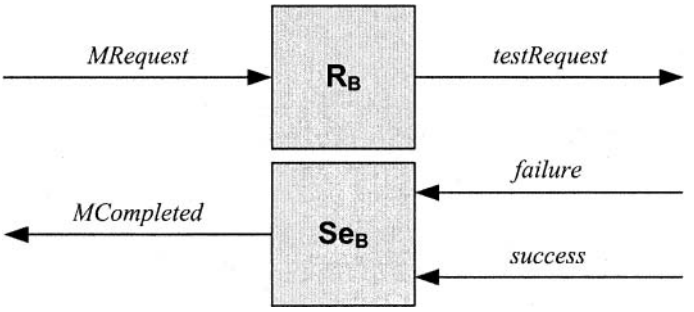

Figure 6-3. Resulting logic diagram

A refiner box is used to map the elements and documents within the message "*MRequest*" to the message "*testRequest*" as expected. Additionally, a select box was put in place to map the elements and documents used in the messages "*failure*" and "*success*" to the message "*MComplete*" containing the document "*DCompleted*". Figure 6-3 shows the resulting logic diagram.

7. CONCLUSIONS

This chapter has presented the main ideas and principles behind service choreography. In so doing it has carefully reviewed the main initiatives in the field with the aim of pointing out their drawbacks. Taking as starting point this analysis, the main driving principles and desire features, when it comes to modeling choreographies, were identified. Later, the most relevant challenges in the field, separation of models and support for semantic mediation were discussed. Based on this theoretical work, the core principles and architecture of a choreography engine that relies on the semantic description of MEPs to allow interoperation among heterogeneous services was presented. Finally, the concepts depicted on the framework as applied in the Assurance Integration Use case part of the DIP project (DIP) have been presented.

8. QUESTIONS FOR DISCUSSION

Beginner:
1. Usage and benefits of a centralized point of control vs. a decentralized one.
2. Benefits and extension of a layered model for choreography.

Intermediate:
1. Try to find real use cases where the support for transaction support within choreographies is required.
2. Discuss the benefits/drawbacks subsumed by a model that does not make any assumptions with respect to the underlying technology and one that is rigidly tight to a particular one.
3. Value added of using MEP to describe semantic business processes and their relation to choreography.

Advanced:
1. Discuss the main pros and cons of the different choreography-related initiatives paying special attention to their variable usage.

9. SUGGESTED READINGS

- 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. Practitioners should find this specification to be quite valuable companion.
- Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Susan Struble S., Takacsi-Nagy, P., Trickovic, I. and Zimek, S. *Web Service Choreography Interface (WSCI) 1.0*, http://www.w3.org/TR/wsci/, 2002. This work provides a very consistent choreography specification.
- Arroyo, S. *SOPHIE - Semantic services chOreograPHi engInE*, PhD Thesis, To appear. The work provides a good insight to choreography from a semantic point of view.
- Kavantzas, 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. This paper presents the WS-CDL specification.

- Barros, A., Dumas, M. and Oaks, P. *A Critical Overview of the Web Services Choreography Description Language (WS-CDL)*, BPTrends, March 2005. This paper presents a nice critical overview of WS-CDL.
- Roman, D., Scicluna, J., Feier, C., (eds.) Stollberg, M and Fensel, D. *D14v0.1. Ontology-based Choreography and Orchstration of WSMO Services*, http://www.wsmo.org/TR/d14/v0.1/, March, 2005. This deliverable is a good introduction to WSMO choreography.

10. REFERENCES

Arkin, A., Askary, S., et al. (2002). Web Service Choreography Interface (WSCI) Version 1.0, http://www.w3.org/TR/wsci/.

Arroyo, S. SOPHIE - Semantic services chOreograPHi engInE, PhD Thesis, To appear.

Arroyo, S., López Cobo, J.M. et al. (2005). Structural models of patterns of message interchange in decoupled hypermedia systems. International Workshop on Peer to Peer and Service Oriented Hypermedia: Techniques and Systems in conjunction with sixteenth ACM Conference on Hyertext and Hypermedia, (HT'05). Salzburg, Austria.

Arroyo, S. and Duke, A. (2005). SOPHIE - A Conceptual model for a Semantic Choreography Framework. Workshop on Semantic and Dynamic Web Process (SDWP'05) in conjunction with the International Conference on Web Services ICWS'05, Orlando, Florida, EEUU.

Arroyo, S., Sicilia, M. A. and López-Cobo, J. M. "Choreography Frameworks for Business Integration: Addressing Heterogeneous Semantics." Computers in Industry. Submitted.

Arroyo, S. and Kummenacher, R. A Choreographed Approach for Ubiquitous and Pervasive Learning. Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to their full potential. Miltiadis D. Lytras and Ambjorn Naeve Editors. IDEA Group Inc. [to appear late 2006]

Arroyo, S. and Sicilia, M. A. (2005). SOPHIE – Architecture and Overall Algorithm of a Choreography Service. First Online Metadata and Semantics Research Conference.

Andrews, T., Curbera, F. et al. (2003). Business Process Execution Language for Web Services Version 1.1,
ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf.

Baader, F., Calvanese, D. et al. (2003). The Description Logic Handbook. Theory, Implementation and Applications. Cambridge, UK.

Barros, A., Dumas, M. et al. (2005). A Critical Overview of the Web Services Choreography Description Language (WS-CDL).BPTrends.

Berners-Lee, T., J. Hendler, et al. (2001). The Semantic Web. Scientific American. **May 2001**.

Booch, G., Rumbaugh, J. et al. (1999). The Unified Modelling Language User Guide. Addison-Wesley.

Brogi, A., Canal, C., et al. (2004). "Formalizing Web Services Choreographies." Electronic Notes in Theoretical Computer Science **105**, 73-94.

DIP-Data, Information, and Process with Semantic Web Services,
http://dip.semanticweb.org/.

Chen, M. (2003). "Factors affecting the adoption and diffusion of XML and Web services standards for E-business systems." International Journal of Human-Computer Studies **58**(3), 259-279.

Erl, T. (2004). Service-Oriented Architecture – A Field Guide to Integrating XML and Web Services, Prentice Hall.

Fensel, D. and Bussler, C. (2002). "The Web Service Modeling Framework (WSMF)." Electronic Commerce: Research and Applications **1**:113-137.

Jung, J., Hur, W. et al. (2004). "Business process choreography for B2B collaboration." IEEE Internet Computing **8**(1), 37 – 45.

Kavantzas, N., Burdett et al. (2004). Web Services Choreography Description Language Version 1.0,
http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/

Roman, D., Scicluna, J., Feier, C. et al. (2005). D14v0.1. Ontology-based Choreography and Orchstration of WSMO Services,
http://www.wsmo.org/TR/d14/v0.1/

Tsalgatidou, A. and Pilioura, T. (2002). "An Overview of Standards and Related Technology." Web Services Distributed and Parallel Databases **12**:135–162.

World Wide Web Consortium,
http://www.w3.org/

Watkins, S., Arroyo, et al. (2005). D8.3: Prototype Platform Design" Case Study B2B in Telecommunications. DIP (Data, Information and Processes). **June 2005**.

XML Schema Part 2: Datatypes Second Edition, (2004)
http://www.w3.org/TR/xmlschema-2/.

Zaremba, M. and Moran M. (2004). D13.4v0.1 WSMX Architecture,
http://www.wsmo.org/2004/d13/d13.4/v0.1/20040622/.