Towards a formal framework for Choreography

Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi and Gianluigi Zavattaro
Department of Computer Science
Mura Anteo Zamboni, 7
40127 Bologna - Italy
{busi,gorrieri,cguidi,lucchi,zavattar}@cs.unibo.it

Abstract— **One of the main challenges in the area of Service Oriented Computing, in general, and of Web services technology, in particular, is the definition of languages and models for the description of** *choreographies*. **A choreography defines the collaborations between interacting services: more precisely, it specifies a contract containing a "global" definition of the common ordering conditions and constraints under which messages are exchanged in a services conversation.**
In this paper, starting from the analysis of the main aspects of Web services technology, we propose a simple choreography language, equipped with a formal semantics, which is intended as the starting point for the development of a framework for the design and analysis of choreographies in Service Oriented Computing.

Index Terms — **Choreography languages, Web services, Service Oriented Computing**

I. INTRODUCTION

Service-oriented computing is a new emerging paradigm for distributed computing that is strongly influencing the way software applications are designed and developed. Services are computational entities that are autonomous and heterogeneous (e.g. running on different platforms or owned by different organizations). Services are described using appropriate service description languages, published and discovered according to predefined protocols, and combined using orchestrator engines that coordinate the interaction among collaborating services. In this way, applications are built up as networks of collaborating applications distributed within and across organizational boundaries.

Web services are the current most promising technology based on the idea of Service Oriented Computing. Web services provide the basis for the development and execution of processes that are distributed over the Internet and available via standard interfaces and protocols. Web services are characterized by three main features: *loose coupling*, *statelessness* and *open endedness*.

Loose coupling refers to the design approach aiming at the reduction of the interdependencies across collaborating services. In particular, Web services applications tend to reduce the risk that changes or unexpected behaviours within one service will create unanticipated changes within other services. This approach specifically seeks to increase flexibility in adding and replacing services, and changing operations within individual services.

Statelessness is concerned with the fact that a service does not maintain the state of a conversation in which it is involved. Each message exchange is a new message exchange completely separated from the previous one. Therefore, in order to correlate multiple messages exchanged by parties in complex interactions, messages must contain information about conversation context. For instance, in the conversation between a traveller and an airplane booking service, the reservation number is inserted into each exchanged message (e.g. for booking confirmation or cancellation) as an indication of the context under which the message should be considered.

Open endedness deals with the fact that, exploiting service discovery mechanisms, it is possible to find at run-time a service which can be invoked in order to achieve a specific activity. In this sense it is not important which service is invoked but the fact that a particular request can be satisfied by an available service.

Another relevant feature for Web services is the mechanism for their reuse when complex tasks are carried out. It is often the case, to define new processes out of finer-grained subtasks that are likely available as Web services. To this aim, extensions of the Web service technology are considered which support the definition of complex services out of simpler ones. Two possible approaches are currently investigated. The first approach, referred to as Web services *orchestration*, combines available services adding a central coordinator (the orchestrator) which is responsible for invoking and combining the single sub-activities. The second approach, referred to as Web services *choreography*, does not assume the exploitation of a central coordinator but it defines complex tasks via the definition of the conversation that should be undertaken by each participant. Following this approach, the overall activity is achieved as the composition of peer-to-peer interactions among the collaborating services.

Several proposals already exist for orchestration languages, see e.g. BPML [4] by BPMI.org, XLANG [15] and BizTalk [6] (a visual specification environment for XLANG) by Microsoft, WSFL [10] by IBM, and the more recent proposal WS-BPEL4 [8] supported by a consortium comprising BEA, IBM, Microsoft, and others. On the contrary, choreography languages are still at a preliminary stage of definition. A first proposal, named WS-CDL [18], has been issued from W3C in December 2004.

In this paper, starting from the analysis of Web services technology and the main concepts at the basis of WS-CDL, we design a simple choreography language named `CL`. This language represents a first step towards the definition of a formal framework for the design, analysis and development of choreographies. Following the approach of WS-CDL, in CL a choreography contains a “global” definition of the common ordering conditions and constraints under which messages are exchanged within a conversation among collaborating services. More precisely, a choreography specification contains a *declarative* and a *conversational* part. The former includes the definition of the roles embodied by the participants in the conversation. Each role has an associated identifier, a set of operations that can be invoked by other roles, and a set of variables indicating the actual state of the role within the conversation. The latter describes the ordering in the interactions among the roles required to complete the conversation. We follow a process algebraic approach for the definition of the conversational part: the basic activities are the role interactions; basic activities can be combined using parallel, choice and sequence operators. A formal semantics describing the possible evolutions of a conversation is defined in terms of a transition system.

The paper is structured as follows. In Section 2 we recall the main aspects of Web services technology and in Section 3 we present the formal definition of our simple choreography language CL. In Section 4 we show CL at work in the specification of a case study and in Section 5 we draw some conclusive remarks.

II. Web services technology

Web services are characterized by two fundamental specifications: WSDL [19] and SOAP [17]. The former defines the interface a Web service exhibits in order to be invoked by other services. The defines the application protocol the services have to follow in order to exchange messages. In the following we will use the terms Web services and services as synonymous.

Operations are the only mechanisms for interacting with a Web service. An operation contains the definition of the exchanged messages and the order in which they have to be performed, that is the incoming and outcoming messages. More precisely, WSDL supports four kinds of operation: i) *Notification*, ii) *One-Way*, iii) *Request-Response*, and iv) *Solicit-Response*. One-Way and Request-Response have a more relevant role in choreography w.r.t. the other ones. Indeed, in choreography we are interested in describing interactions between two participants which are known by the system, while usually notification mechanisms are used, at the implementation level, to describe cases where the partner is not known *a priori*. Finally, the Solicit-Response is the dual of Request-Response and, at the choreography level, they are combined in a single interaction which describes the exchanged messages between the two partners. The One-Way and Request-Response definition in WSDL follows:

- One-Way: only the incoming message is defined. In the WSDL specifications it is represented by the XML schema:

```
<wsdl:operation name="nmtoken">*
     <wsdl:input name="nmtoken"?
                 message="qname"/>?
</wsdl:operation>
```

- Request-Response: both the incoming message and the response one are defined. In the WSDL specifications it is represented by the XML schema:

```
<wsdl:operation name="nmtoken">*
     <wsdl:input name="nmtoken"?
                 message="qname"/>?
     <wsdl:output name="nmtoken"?
                  message="qname"/>?
</wsdl:operation>
```

When a Web service is invoked by using a *One-Way* operation, it receives the incoming message and starts its activities. On the other hand, when a Web service is invoked by using a Request-Response operation, the service receives the message, starts its activities and, at the end, replies to the invoker with a response message.

Web services composition deals with the management of complex systems composed by several services which collaborate within a so called *business activity*. In this sense the terms choreography and orchestration are often considered as synonymous even if there are essential differences as shown in [7]. Choreography aims at constricting the behaviours of the services involved in the system ruling the exchange of their messages. On the other hand, orchestration allows the design of a central entity which carries out a business activity invoking other services.

Here, we list some discriminating features between orchestration and choreography referring to WS-BPEL [13] as an example of orchestration language and WS-CDL [18] as an example of choreography language.

- *Executable processes*: choreography describes the system and does not provide specifications for its execution. On the contrary orchestration assumes the existence of engines able to animate specific orchestration code.
- *Interactions design*: choreography provides a top view of the system focused on the interactions between the participants. It is a definition of the rules which govern messages exchange among the parties involved in the business activity. There is no notion of a central entity (e.g. a coordinator) which carries out the activity. On the contrary orchestration is always centered on the orchestrator engine which drives all the interactions. For instance, if there are two services which require to be synchronized, the first has to send the synchronization message to the orchestrator engine which will forward it to the latter.
- *Activity state*: In choreography the state of the activity is distributed among the entities. Indeed some internal variables of the entities involved are fundamental for the progression of the choreography. After a message exchange, the choreography may require that two variables which are located on the sender site and on the receiver site must be aligned. On the contrary, orchestration – as said before– is centered on the orchestrator which is the entity which manages the communications and which stores all the state of the activity it is carrying out.

III. A FORMAL MODEL FOR CHOREOGRAPHY

Here we introduce a choreography formal model which is based on the concepts of *roles* and *interactions*. The former represents the behaviour that a participant has to exhibit in order to fulfill the activity defined by the choreography whereas the latter focuses on the information exchanges among the roles. In particular all the interactions will be peer-to-peer interactions as depicted in WS-CDL specifications.

We will present a language for choreography and its semantics which will allow to express the evolution of a system exploiting the interactions among its roles. The cornerstone of this language is the operation that allows roles to interact. Furthermore, we also introduce an auxiliary language, based on single message exchange between two roles, that we use to express the semantics of the choreography language. To this end, the encoding rules mapping the choreography language in the auxiliary one will be presented.

A. Variables, operations and roles

Each role can store variables and exhibit operations. As far as variables are concerned, we associate to each role a set of variables, representing the internal state of the role, that will be used in the interactions between roles. As far as operations are concerned, the operations of a role ρ are essentially the access points that will be used by the other roles to interact with ρ. To this end each role is equipped with a set of operations the role exhibits. Operations can have one of the following interaction modalities: One-Way or Request-Response.

Let Var be the set of variables ranged over by x, y, z, k. We denote with $\widetilde{x}$ tuples of variables, for instance, we may have $\widetilde{x} = \langle x_1, x_2, ..., x_n \rangle$.

Let $OpName$ be the set of operation names, ranged over by o, and $OpType = \{ow, rr\}$ be the set of operation types where ow denotes a One-Way operation whereas rr denotes the Request-Response one. An operation is described by its operation name and operation type. Namely, let Op be the set of operations defined as follows:

$$Op = \{(o,t) \mid o \in OpName,\ t \in OpType\}$$

A role is described by a role name, the set of operations it exhibits and by a set of variables. Namely, let $RName$ be the set of the role names, ranged over ρ, the set $Role$ containing all the possible roles is defined as follows:[1]

$$Role = \{(\rho, \omega, V) \mid \rho \in RName,\ \omega \in \mathbf{P}(Op), V \in \mathbf{P}(Var)\}$$

B. The choreography language

A choreography is a pair formed by a set of roles, describing the entities involved in the interaction, and by a conversation, modeling the ordering of interactions among the roles.

The choreography language CL is used to describe the conversational part of a choreography. The basic building blocks are two different primitives, one for expressing an interaction with a one-way operation and one for an interaction with a request-response operation. The language allows us to compose interactions exploiting sequence, parallel and choice operators.

[1]Given a set S, with $\mathbf{P}(S)$ we denote the powerset of S.

Formally, let CL be the set of conversations defined by the following grammar:

$$\begin{array}{rcl} C & ::= & \mathbf{0} \\ & \mid & OW(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}) \\ & \mid & RR(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{k}, C) \\ & \mid & C; C \\ & \mid & C \mid C \\ & \mid & C + C \end{array}$$

A conversation can be a terminated conversation ($\mathbf{0}$), the primitive denoting the interaction with one-way operations (OW) or with request-response ones (RR). The primitive $OW(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y})$ is an invocation of the operation o of role ρ_B, performed by role ρ_A; $\widetilde{x}$ (resp. $\widetilde{y}$) are sequences of variables of ρ_A (resp. ρ_B). As an effect of the execution of the operation, the contents of the variables $\widetilde{x}$ of ρ_A are assigned to the variables $\widetilde{y}$ of ρ_B. The primitive $RR(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{k}, C)$ is used to describe the invocation of a request-response operation. It is an invocation of the operation o of role ρ_B, performed by role ρ_A. At the beginning of the interaction, the contents of the variables $\widetilde{x}$ of ρ_A are assigned to the variables $\widetilde{y}$ of ρ_B; after, the role ρ_B executes the conversation C (the last paramenter of the RR operation); finally, the contents of the variables $\widetilde{k}$ of ρ_B are assigned to the variables $\widetilde{z}$ of ρ_A.

Note that in the Request-Response operation the interactions denoted by the last parameter C occur between the request and the response. This is a relevant difference with WS-CDL where the Request-Response interaction cannot be split avoiding the possibility to perform some interactions after the request and before the response message.

Finally, conversations can be: i) the sequential composition of two conversations $C; C'$ whose meaning is that C' can be performed when C completes, ii) the parallel composition of two conversations $C \mid C'$ which represents the concurrent execution of conversations C and C', and iii) the alternative composition of two conversations $C + C'$ whose meaning is that the conversation to be performed is non-deterministically selected between C and C'.

Now we are ready to define a choreography. A choreography, denoted by SC, is defined by a pair (C, Σ) where $C \in CL$ and $\Sigma \subseteq Role$. We define CL as the set containing all the choreographies.

We are interested in choreographies satisfying the well-formedness conditions listed in the following definitions.

A set of roles Σ is well-formed if it is finite and the set of role names are all distinct.

Definition 3.1: **Well-formed set of roles –** Let $\Sigma \subseteq Role$. The set of roles Σ is well-formed if the following conditions hold:

1) Σ is finite;
2) if $(\rho_i, \omega_i, \sigma_i) \in \Sigma$ and $(\rho_j, \omega_j, \sigma_j) \in \Sigma$ and $\rho_i = \rho_j$ then $i = j$.

Definition 3.2: **Well-formed choreography –**
Let (C, Σ) be a choreography; C is well-formed if:

1) for any operation $OW(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y})$ it contains the following conditions hold:
   a) $(\rho_A, \omega_A, V_A), (\rho_B, \omega_B, V_B) \in \Sigma$ for some ω_A, V_A and ω_B, V_B;
   b) $(o, ow) \in \omega_B$;
   c) $\widetilde{x}$ and $\widetilde{y}$ have the same arity;
   d) $\widetilde{x} \subseteq V_A$ and $\widetilde{y} \subseteq V_B$
2) for any operation $RR(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{k}, C)$:
   a) $(\rho_A, \omega_A, V_A), (\rho_B, \omega_B, V_B) \in \Sigma$ for some ω_A, V_A and ω_B, V_B;
   b) $(o, rr) \in \omega_B$;
   c) $\widetilde{x}$ and $\widetilde{y}$ have the same arity;
   d) $\widetilde{z}$ and $\widetilde{k}$ have the same arity;
   e) $\widetilde{x} \subseteq V_A$, $\widetilde{y} \subseteq V_B$, $\widetilde{z} \subseteq V_A$, and $\widetilde{k} \subseteq V_B$

The first condition requires that the roles involved are contained in the system. The second guarantees that the role which receives the interaction exhibits the operation used to interact. The third condition (and the fourth condition for request-response) ensure that the sender and the receiver use the same number of variables. Finally, the last condition ensure that the specified variables belong to the corresponding role.

C. Semantics

The semantics of the conversations is presented in terms of the semantics of an auxiliary language $\mathtt{CL}_P$; more precisely, we define $\mathtt{CL}_P$ and we show how to translate a conversation $C \in CL$ into $\mathtt{CL}_P$, then we present the semantics for $\mathtt{CL}_P$.

1) Syntax of $\mathtt{CL}_P$: The auxiliary language is defined by the following grammar:

$$\begin{array}{rcl} C & ::= & \mathbf{0} \\ & | & m \\ & | & C_P; C_P \\ & | & C_P \mid C_P \\ & | & C_P + C_P \\ \\ m & ::= & (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir) \end{array}$$

In the following we use CL_P, ranged over by C_P, to denote the set of conversations of such a language. We limit the description to the interaction m since the composition operators are the same and with same meaning of those of $\mathtt{CL}$. $(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir)$ means that an interaction from role ρ_A to role ρ_B is performed. In particular, o is the name of the operation $(o, t) \in Op$ on which the message exchange is performed. Variables $\widetilde{x}$ and $\widetilde{y}$ are those used by the sender and the receiver, respectively. After the interaction the values stored in $\widetilde{x}$ are assigned to the variables $\widetilde{y}$. Finally, $dir \in \{\uparrow, \downarrow\}$ indicates whether the interaction is a request ($\uparrow$) or a response ($\downarrow$) of o. Thus the dir parameter is needed for allowing us to reason on simple interaction and at the same time for preserving information about the type of the operation on which the message exchange is performed.

2) Mapping of $\mathtt{CL}$ on $\mathtt{CL}_P$: We define a mapping from $\mathtt{CL}$ conversations into terms of the auxiliary language $\mathtt{CL}_P$.

We map a one-way operation on a single interaction, while a request-response is mapped on the following sequence of operations: a first interaction, representing the request, followed by the mapping of the conversation C specified in the RR primitive, followed by the response interaction.

Definition 3.3: The function $\|-\| : CL \to CL_P$ is defined inductively as follows:

$$\|\mathbf{0}\| = \mathbf{0}$$

$$\|OW(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y})\| = (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \uparrow)$$

$$\left\|RR(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{k}, C)\right\| = (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, \uparrow); \|C\|; (\rho_B, \rho_A, o, \widetilde{k}, \widetilde{z}, \downarrow)$$

$$\|C; C'\| = \|C\|; \|C'\|$$

$$\|C \mid C'\| = \|C\| \mid \|C'\|$$

$$\|C + C'\| = \|C\| + \|C'\|$$

3) Semantics of $\mathtt{CL}_P$: The semantics of $\mathtt{CL}$ is defined in terms of a transition system which describes the evolution of a conversation expressed in the auxiliary language. $C_P \to C'_P$ means that the conversation C_P evolves in one step in a configuration C'_P. We define $\to$ as the least relation which satisfies the axioms and rules of Table I and closed w.r.t. $\equiv$, where $\equiv$ is the least congruence relation satisfying the axioms at the end of Table I.

(INTERACTION)
$$(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir) \to \mathbf{0}$$

(SEQUENCE)
$$\frac{C_P \to C'_P}{C_P; D_P \to C'_P; D_P}$$

(PARALLEL)
$$\frac{C_P \to C'_P}{C_P \mid D_P \to C'_P \mid D_P}$$

(CHOICE)
$$\frac{C_P \to C'_P}{C_P + D_P \to C'_P}$$

(STRUCTURAL CONGRUENGE)
$$\mathbf{0}; C_P \equiv C \qquad C_P \mid \mathbf{0} \equiv C_P \qquad C_P + \mathbf{0} = C_P$$
$$C_P + D_P \equiv D_P + C_P \quad C_P \mid D_P \equiv D_P \mid C_P$$
$$(C_P + D_P) + E_P \equiv C_P + (D_P + E_P)$$
$$(C_P \mid D_P) \mid E_P \equiv C_P \mid (D_P \mid E_P)$$

TABLE I
SEMANTICS OF CL_P CONVERSATIONS

The structural congruence $\equiv$, which equates the conversations whose behavior cannot be distinguished, expresses that: i) when a conversation completes then the other one which follows in sequence can be performed, ii) a terminated conversation running in parallel does not alterate the behavior of the entire conversation, iii) a conversation C composed in alternative to a terminated one behaves as C, iv) the order of conversations composed in parallel or in alternative way is not significant, and v) the associativity property on parallel and alternative composition holds.

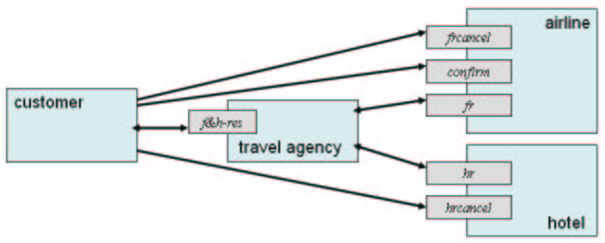

Fig. 1. Service network for flight and hotel reservation

The description of axioms and rules follows. The axiom INTERACTION describes the behavior of an interaction. ρ_A is the name of the sender role whereas ρ_B is the name of the receiver one, while o is the operation name used to interact and $\widetilde{x}$, $\widetilde{y}$ are the variables used by the sender and the receiver to exchange data, respectively. The rules SEQUENCE, PARALLEL and CHOICE are standard.

IV. EXAMPLE

Let us consider the following simple example which describes a typical scenario where a travel agency provides the flight and hotel reservation service. The system is composed of four roles whose names are ρ_c, ρ_{ta}, ρ_f and ρ_h representing the customer, the travel agency, the airline and the hotel, respectively. Customers interact with the travel agency sending data about the travel (e.g., the name of the customer, the departure and the destination of the travel, and the period in which the customer intend to perform the travel). The travel agency interacts with the airline and, once having obtained the flight reservation, performs the hotel reservation as well. When both reservations have been completed the travel agency responds to the customer who decides whether to accept (in this case it confirms the flight while we assume that the hotel does not need such a notification) or to cancel the flight and the hotel reservations (in this case it notifies to both airline and hotel that he intends to cancel the reservation).

We proceed by describing the roles defining the operations they exhibit. The travel agency, the airline and the hotel exhibit operations while the customer does not:
$\omega_{ta} = \{(f\&h{-}res, rr)\}$
$\omega_f = \{(fr, rr), (confirm, ow), (frcancel, ow)\}$
$\omega_h = \{(hr, rr), (hrcancel, ow)\}$.

The interaction schema is depicted in Figure 1. The request-response operation $f\&h{-}res$ is used by customers to request the flight and hotel reservation, fb (resp. hr) is the request-response operation used to book a flight (resp. reserve a room in the hotel). The request-response operations use a response to communicate the result of the flight/hotel reservation that is propagated to the customer. At this stage the customer can decide whether confirm or cancel the flight and hotel reservations.

We define separately the variables used in the interactions by using the conventions that variables denoted by x are used by the customer, y by the travel agency, z by the airline and finally k by the hotel. The definition of the tuples used in the interactions follows:

$\widetilde{x}_{cdata} = \langle name_x, from_x, to_x, depdate_x, retdate_x\rangle$
$\widetilde{x}_{f\&h} = \langle deptime_x, retime_x, frcode_x, hrcode_x\rangle$
$\widetilde{y}_{cdata} = \langle name_y, from_y, to_y, depdate_y, retdate_y\rangle$
$\widetilde{y}_f = \langle deptime_y, retime_y, frcode_y\rangle$
$\widetilde{y}_h = \langle hrcode_y\rangle$
$\widetilde{z}_f = \langle deptime_z, retime_z, frcode_z\rangle$
$\widetilde{z}_{cdata} = \langle name_z, from_z, to_z, depdate_z, retdate_z\rangle$
$\widetilde{k}_f = \langle deptime_k, retime_k, frcode_k\rangle$
$\widetilde{k}_h = \langle hrcode_k\rangle$

The formalization of the choreography $SC_{F\&H{-}res}$ describing the flight and hotel reservation process follows.

$$SC_{F\&H{-}res} = (C_{F\&H}, \Sigma)$$

where Σ contains the four roles, defined in such a way that they contain the variables defined above for the interactions, while $C_{F\&H}$ is defined as follows:

$$\begin{aligned}
&C_{F\&H} = \\
&\quad RR(\rho_c, \rho_{ta}, f\&h{-}res, \widetilde{x}_{cdata}, \widetilde{y}_{cdata}, \widetilde{x}_{f\&h}, \widetilde{y}_f \circ \widetilde{y}_h, C_{ta});\\
&\quad (C_{f\&h{-}confirm} + C_{f\&h{-}cancel})\\
&C_{ta} = RR(\rho_{ta}, \rho_f, fb, \widetilde{y}_{cdata}, \widetilde{z}_{cdata}, \widetilde{y}_f, \widetilde{z}_f, \mathbf{0});\\
&\quad RR(\rho_{ta}, \rho_h, hr, \widetilde{y}_f, \widetilde{k}_f, \widetilde{y}_h, \widetilde{k}_h, \mathbf{0})\\
&C_{f\&h{-}confirm} = OW(\rho_c, \rho_f, confirm, frcode_x, frcode_z)\\
&C_{f\&h{-}cancel} = OW(\rho_c, \rho_f, frcancel, frcode_x, frcode_z) \mid\\
&\quad OW(\rho_c, \rho_h, hrcancel, hrcode_x, hrcode_k)
\end{aligned}$$

The first interaction which is performed is the request to the travel agency's operation $f\&h{-}res$ where $\widetilde{x}_{cdata}$ contains information about the name, the departure and arrival locality and the travel period which populate variables $\widetilde{y}_{cdata}$ that the travel agency uses to request the flight reservation. The values returned by the fr operation, containing the ticket code and information about the departure and return time of the flights, are now used by the travel agency to perform the request to the hr operation which supplies the hotel reservation and returns a reservation code. At this stage the conversation C_{ta} performed by the travel agency is completed and then it responds to the customer by passing $\widetilde{y}_f \circ \widetilde{y}_h$ (we use $\circ$ to denote the concatenation of tuples) which contains information about both flight and hotel reservations. Finally, when the the $f\&h{-}res$ operation completes, the customer can confirm or cancel the reservations; such a behavior is modeled by using the choice operator. In the case it confirms the reservations, the conversation consists of a one-way performed on $confirm$ operation of the airline; in the opposite case the conversation is composed of two one-way requests running in parallel, one on $frcancel$ and the other one on $hrcancel$ to cancel the flight and the hotel reservation, respectively (in both cases the variable containing the reservation code is passed).

V. CONCLUSIONS

We have presented `CL`, choreography language, presenting its syntax, its formal semantics, and a case study to show its expressivity. We consider this work as the first step towards the definition of a formal framework for designing applications based on the Service Oriented Computing paradigm. In particular, we intend to show the interdependencies among choreography and orchestration languages in the design of applications by considering the second one a complementary approach to the choreography. In other words, we consider the orchestration as a further development step of the system described by the choreography. As future work we plan to formally define such a relationship that could be used, e.g., to check the correctness of an orchestration w.r.t. a given choreography.

To the best of our knowledge, this is the first attempt towards the definition of a formal framework for choreography while, on the other hand, there are several works which formally deal with services orchestration that we outline below. The only work which deals with another choreography language is [3] where the Web Service Choreography Interface [20] (WSCI) language is modeled. WSCI allows to specify the service composition from a different point of view w.r.t. the one of WS-CDL and our proposal CL. In particular, WSCI specifications describe the behavior of a single participant when it plays a role within a complex conversation; in other words, it represents a sort of projection of the entire conversation to the considered participant. In our case, instead, the choreography describes all interdependencies among the different interactions between roles.

As previously mentioned, here we list some of the works related with Web services orchestration available in the literature. The works [2], [5], [9], [12] formally reason on long running transactions mechanisms used in orchestration languages which, essentially, allow to program both the basic activity and those to be performed in the case the transaction fails. Security issues have been investigated in [1] where WS-Security [14] has been used as a case study. Another aspect is concerned with correlation sets (used, e.g., by WS-BPEL) which provide a mean for correlating several interactions between services; such a mechanism have been formalized in an orchestration language in [16]. Finally, WSSecSpaces [11] is a data-driven coordination service, equipped with an operational semantics, supporting some mechanisms for a secure collaboration among services.

REFERENCES

[1] K. Bhargavan, C. Fournet, and A.D. Gordon. A semantics for web services authentication. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 198–209. ACM, 2004.

[2] L. Bocchi, C. Laneve, and G. Zavattaro. A Calculus for Long-Running Transactions. In *FMOODS*, volume 2884 of *LNCS*, pages 124–138. Springer Verlag, 2003.

[3] A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo. Formalizing web services choreographies. In M. Bravetti and G. Zavattaro, editors, *Proc. of 1st International Workshop on Web Services and Formal Methods (WS-FM 2004)*, volume 105 of *ENTCS*. Elsevier, 2004.

[4] Business Process Modeling Language (BPML). [http://www.w3.org/TR/wsdl].

[5] M. J. Butler and C. Ferreira. An Operational Semantics for StAC, a Language for Modelling Long-Running Business Transactions. In *COORDINATION*, volume 2949 of *LNCS*, pages 87–104. Springer Verlag, 2004.

[6] Microsoft Corporation. Microsoft BizTalk Server. [http://www.microsoft.com/biztalk/default.asp].

[7] C.Peltz. Web services orchestration and choreography. *Web Services Journal*, July 2003.

[8] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. Business Process Execution Language for Web Services (BPEL4WS 1.0). [http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/], 2002.

[9] C. Laneve and G. Zavattaro. Foundations of Web Transactions. In *In Proc. of International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*. LNCS. To appear.

[10] F. Leymann. Web Services Flow Language (WSFL 1.0). [http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf], Member IBM Academy of Technology, IBM Software Group, 2001.

[11] R. Lucchi and G. Zavattaro. WSSecSpaces: a Secure Data-Driven Coordination Service for Web Services Applications. In *Proc. of ACM Symposium on Applied Computing (SAC'04)*, pages 487–491. ACM Press, 2004.

[12] M. Mazzara and R. Lucchi. A Framework for Generic Error Handling in Business Processes. In M. Bravetti and G. Zavattaro, editors, *Proc. of 1st International Workshop on Web Services and Formal Methods (WS-FM 2004)*, volume 105 of *ENTCS*. Elsevier, 2004.

[13] Microsoft,IBM, Siebel Systems, BEA. *Business Process Execution Language for Web Services Version 1.1*. [http://www-106.ibm.com/developerworks/library/ws-bpel/].

[14] OASIS. *Web Services Security (WS-Security)*. [http://www-106.ibm.com/developerworks/webservices/library/ws-secure/].

[15] S. Thatte. XLANG: Web Services for Business Process Design. [http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm], Microsoft Corporation, 2001.

[16] M. Viroli. Towards a Formal Foundation to Orchestration Languages. In M. Bravetti and G. Zavattaro, editors, *Proc. of 1st International Workshop on Web Services and Formal Methods (WS-FM 2004)*, volume 105 of *ENTCS*. Elsevier, 2004.

[17] W3C. *SOAP Version 1.2 Part 1: Messaging Framework*. [http://www.w3.org/TR/soap12-part1/].

[18] W3C. *Web Services Choreography Description Language Version 1.0. Working draft 27 April 2004*. [http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/].

[19] W3C. *Web Services Description Language (WSDL) 1.1*. [http://www.w3.org/TR/wsdl].

[20] World Wide Web Consortium (W3C). Web service choreography interface (wsci) 1.0. [http://www.w3.org/TR/wsci], 2002.