Choreography and Orchestration: A Synergic Approach for System Design*

Nadia Busi, Roberto Gorrieri, Claudio Guidi,
Roberto Lucchi, and Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy
{busi, gorrieri, cguidi, lucchi, zavattar}@cs.unibo.it

Abstract. Choreography and orchestration languages deal with business processes design and specification. Referring to Web Services technology, the most credited proposals are WS-CDL about choreography and WS-BPEL about orchestration. A closer look to such a kind of languages highlights two distinct approaches for system representation and management. Choreography describes the system in a top view manner whereas orchestration focuses on single peers description. In this paper we define a notion of conformance between choreography and orchestration which allows to state when an orchestrated system is conformant to a given choreography. Choreography and orchestration are formalized by using two process algebras and conformance takes the form of a bisimulation-like relation.

1 Introduction

In the design and deployment of service oriented applications two different and opposite features should be taken into account. On the one hand, it is important to program single peers services, which could be involved in different systems at different times, preserving their composionality, and on the other hand, it is fundamental to guarantee overall systems functionalities. Orchestration and choreography deal with such a kind of issues where the former focuses on single peers description whereas the latter describes a system in a top view manner.

In this paper we present a notion of conformance between two simple formal languages we have developed for representing choreography and orchestration. They are inspired by Web Services technical specifications as WS-CDL [W3C] and WS-BPEL [OAS]. Here we intend to give a mathematical *liaison* between the two different approaches of choreography and orchestration in order to give a powerful mechanism for designing systems where peers behaviours and systems functionalities are developed together in a complementary fashion.

The choreography language has been presented in our previous works [BGG$^+$] [GGL05] whereas the orchestration one is introduced in this work. They are based both on basic Web Services interaction mechanisms, the so called *operations*, defined in WSDL specifications. Briefly, we remind that an operation contains

* Research partially funded by EU Integrated Project Sensoria, contract n. 016004.

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 228–240, 2005.

the definition of an incoming message for a service and, when used, the definition of the response one.

The orchestration language takes inspiration from WS-BPEL. More precisely we have been inspired by the abstract non-executable fragment of WS-BPEL used to specify the observable behaviour of services abstracting away from internal details. Abstract WS-BPEL exploits opaque variables (i.e. variables without a specified content) to indicate which state variables could influence a choice taken from a service, without specifying the internal decision procedure of the service. In our language, all variables are opaque; this permits us to focus on interaction mechanisms and data flow abstracting away from variable values.

The choreography language is named CL_P and is used to describe the behaviour of a system by defining the involved roles and their interactions whereas the orchestration language is called OL and it allows to program communicating behaviours of the single peers which can be composed for obtaining an orchestrated system. We define the semantics of such languages by using two different labelled transition systems and we present a notion of conformance based on a bisimulation between them with some similarities with branching bisimulation [vGW96]. In particular, we define a so called *joining function* in order to associate the orchestrators of the orchestrated system to the roles of the choreography and we construct the definition of conformance on a bisimulation which exploits such a function in order to compare the labels of the two different labelled transition systems.

In section 2 we present the choreography language whereas in section 3 we present the orchestration one. Then in section 4 we present the joining function and we give the notion of conformance. Furthermore, in section 5, an example is reported and in section 6 conclusions are presented.

2 A Formal Model for Choreography

In this section we introduce the formal model for representing choreography which is based on a declarative part and on a conversational one. The former deals with roles, operations and variables whereas the latter deals with a language for describing the conversations (interactions composition) among the roles.

Declarative Part. Here we explain the declarative part of our choreography formal model which is based on the concept of *role*. A role represents the behaviour that a participant has to exhibit in order to fulfill the activity defined by the choreography. Each role can store variables and exhibit operations.

As far as variables are concerned, we associate to each role a set of variables which represent the information managed by the role and which will be used in the interactions between roles. In this model we abstract away from the values and we consider variables as names which are exploited for representing the data flow among the roles.

As far as operations are concerned, each role is equipped with a set of operations it has to exhibit which essentially represent the access points that will be

used by the other roles to interact with the owner one. Operations can have one of the following interaction modalities: *One-Way* or *Request-Response*. Indeed, in WSDL specifications, the most significant types of operations are the *One-Way*, where only the incoming message is defined, and the *Request-Response*, where both the incoming message and the response one are defined.

Let us now introduce the formalization of *roles*, *variables* and *operations*.

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 where each operation is univocally identified by its name.

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

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 by ρ. The set $Role$, containing all the possible roles, is defined as follows:

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

Conversational Part. The conversations among the roles are defined by using the following grammar:

$$C_P ::= \mathbf{0} \mid \mu \mid C_P; C_P \mid C_P | C_P \mid C_P + C_P$$
$$\mu ::= (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir)$$

In the following we use CL_P, ranged over by Con, to denote the set of conversations of such a language. C_P denotes a conversation which can be the null one ($\mathbf{0}$), the interaction μ, the sequential composition ($C_P; C_P$), the parallel composition ($C_P \mid C_P$) or the choice one ($C_P + C_P$). $(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir)$ 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. They represent that after the interaction the information 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 ($\downarrow$) one. Choreography well-formedness rules can be found in [BGG$^+$].

Semantics of CL_P. The semantics of CL_P is defined in terms of a labelled transition system [Kel76] which describes the evolution of a conversation. Let Act_C be a set of parameterized actions $(\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir)$ ranged over by μ. $C_P \xrightarrow{\mu} C'_P$ means that the conversation C_P evolves in one step in a configuration C'_P performing the action μ. We define $\rightarrow \subseteq \mathtt{CL}_P \times Act_C \times \mathtt{CL}_P$ as the least relation which satisfies the axioms and rules of Table 1 and closed w.r.t. $\equiv$, where $\equiv$ is the least congruence relation satisfying the axioms at the end of Table 1. The structural congruence $\equiv$, which equates the conversations whose

Table 1. Semantics of CL_P conversations

$$\text{(Interaction)}\quad \mu \xrightarrow{\mu} \mathbf{0},\ \mu = (\rho_A, \rho_B, o, \widetilde{x}, \widetilde{y}, dir) \qquad \text{(Sequence)}\quad \frac{C_P \xrightarrow{\mu} C'_P}{C_P; D_P \xrightarrow{\mu} C'_P; D_P}$$

$$\text{(Parallel)}\quad \frac{C_P \xrightarrow{\mu} C'_P}{C_P \mid D_P \xrightarrow{\mu} C'_P \mid D_P} \qquad \text{(Choice)}\quad \frac{C_P \xrightarrow{\mu} C'_P}{C_P + D_P \xrightarrow{\mu} C'_P} \qquad \text{(Struct)}\quad \frac{C'_P \equiv C_P, C_P \xrightarrow{\mu} D_P, D_P \equiv D'_P}{C'_P \xrightarrow{\mu} D'_P}$$

$$\text{(Structural Congruenge)}$$
$$\mathbf{0}; C_P \equiv C_P \qquad C_P \mid \mathbf{0} \equiv C_P \qquad C_P + \mathbf{0} = C_P$$
$$C_P + D_P \equiv D_P + C_P \quad C_P \mid D_P \equiv D_P \mid C_P$$
$$(C_P + D_P) + E_P \equiv C_P + (D_P + E_P) \qquad (C_P \mid D_P) \mid E_P \equiv C_P \mid (D_P \mid E_P)$$

behavior cannot be distinguished, expresses that $(C_P, +)$ and $(C_P, \mid)$ are abelian monoids where $\mathbf{0}$ is the null element. Furthermore, the rule $\mathbf{0}; C_P \equiv C_P$ means that when a conversation completes then the other one which follows in sequence can be performed.

The description of axioms and rules follows. The axiom Interaction describes that an interaction μ, which is a request or a response one depending on the value of dir, is performed. When a request is performed ($dir = \uparrow$) the information contained in the variables $\widetilde{x}$ within the sender role ρ_A are passed to the variables $\widetilde{y}$ within the receiver role ρ_B exploiting the operation o of the role ρ_B. When a response is performed ($dir = \downarrow$) the information contained in the variables $\widetilde{y}$ within the request receiver role ρ_B are passed to the variables $\widetilde{x}$ within the request sender role ρ_A exploiting the operation o of the role ρ_B. Response must be performed always after a request interaction from ρ_A to ρ_B. The rules Sequence, Parallel, Choice and Struct are standard.

Now we are ready to define a choreography. A choreography, denoted by C, is defined by a pair (Con, Σ) where $Con \in CL_P$ and $\Sigma \subseteq Role$ with Σ finite.

3 A Formal Model for Orchestration

In this section we introduce a formal language for orchestration called OL. An orchestrator can be seen as a process, associated to an identifier, that can exchange information, represented by variables, with other processes. Our model takes inspiration from the abstract non-executable fragment of WS-BPEL and abstracts away from variables values focusing on data-flow. Let Var be the set of variables used for choreography ranged over by x, y, z, k and ID be the set of possible identifiers ranged over by id. We denote with $\tilde{x}$ tuples of variables. The language syntax follows:

$$P ::= \mathbf{0} \mid o \mid \bar{o} \mid o(\tilde{x}) \mid \bar{o}(\tilde{y}) \mid o(\tilde{x}, \tilde{y}, P) \mid \bar{o}(\tilde{x}, \tilde{y}) \mid P; P \mid P + P \mid P \mid P$$
$$E ::= [P]_{id} \mid E \parallel E$$

An orchestrated system E is a pool of named processes. An orchestrator $[P]_{id}$ is a process P identified by id. Informally the idea is that orchestrators are executed on different locations, thus they can be composed by using only the parallel operator ($\|$). Processes can be composed in parallel ($|$), sequence ($;$) and alternative composition ($+$). $\mathbf{0}$ represents the null process. Communication mechanisms model Web Services One-Way and Request-Response operations. In particular, we have three kinds of primitives for synchronization, one for the internal synchronization and two for the external one. The former simply consists of a channel o that different threads of the process running in parallel, can use to coordinate their activities. In this case no message is exchanged; this is because the orchestrator variables are shared by any processes running on that orchestrator. The primitives for external synchronization, that is between different orchestrators, are the following ones: $o(\tilde{x})$ and $\bar{o}(\tilde{y})$ represent the input and the output of a single message whereas the primitives $o(\tilde{x}, \tilde{z}, P)$ and $\bar{o}(\tilde{y}, \tilde{k})$ represent coupled messages exchanges. In particular we have that $o(\tilde{x})$ represents a One-Way operation whose name is o where the received information are stored in the tuple of variable $\tilde{x}$ of the receiver. $\bar{o}(\tilde{y})$ represents a One-Way invocation whose name is o and the sent information are stored in the tuple $\tilde{y}$ of the sender. $o(\tilde{x}, \tilde{z}, P)$ represents a Request-Response operation whose name is o. In this case the process receives a message and stores the received information in $\tilde{x}$ then it executes the process P and, at the end, sends the information contained in $\tilde{z}$ as a response message to the invoker. Finally, $\bar{o}(\tilde{y}, \tilde{k})$ represents the invocation of a Request-Response operation whose name is o. The process sends the information contained in $\tilde{y}$ as a request message and stores the information of the response message in $\tilde{k}$.

Semantics of OL. The semantics of OL is defined in terms of a labelled transition systems which describes the evolution of an orchestrated system. We define $\rightarrow$ as the least relation which satisfies the axioms and rules of Table 2. Let $Act_{OL} = \left\{\bar{o}, o, \bar{o}(\tilde{y}), o(\tilde{x}), \bar{o}(\tilde{y}, \tilde{k})(n), o(\tilde{x}, \tilde{z})(n), \bar{o}_n(\tilde{y}), o_n(\tilde{x}), \sigma, \tau\right\}$, ranged over by γ, be the set of actions. σ is a parameterized action of the form $(id, id', o, \tilde{x}, \tilde{y}, dir)$ where id, id' are orchestrators ids, o is an operation name, $\tilde{x}, \tilde{y}$ are tuples of variables and $dir \in \{\uparrow, \downarrow\}$.

Table 2 is divided into two parts describing the rules and structural congruence for processes and orchestrated systems respectively. In, Out, One-WayOut, One-Way-In, Req-Out, Req-In, Resp-Out, Resp-In are axioms where it is important to note the behaviour of the Req-In one. It stands that after the reception of a request on a Request-Response operation the process P must be executed before sending the response. Rule Int-Sync deals with internal synchronization whereas Par-In and Choice ones deal with internal parallel and choice respectively. Seq describes the behaviour of sequentially composed processes. CongrP deals with internal structural congruence denoted by $\equiv_P$.

Rule One-WaySync deals with the synchronization on a One-Way operation between two orchestrators whereas the rules Req-Sync and Resp-Sync deal with that on a Request-Response one. Rule Req-Sync exploits a fresh label n which is generated in order to univocally link the response synchronization

Table 2. *OL* operational semantics

(Rules over P)

(In) $o \xrightarrow{o} \mathbf{0}$ (Out) $\bar{o} \xrightarrow{\bar{o}} \mathbf{0}$ (One-WayOut) $\bar{o}(\tilde{y}) \xrightarrow{\bar{o}(\tilde{y})} \mathbf{0}$ (One-WayIn) $o(\tilde{x}) \xrightarrow{o(\tilde{x})} \mathbf{0}$ (Req-Out) $\bar{o}(\tilde{x},\tilde{y}) \xrightarrow{\bar{o}(\tilde{x},\tilde{y})(n)} o_n(\tilde{y})$

(Req-In) $o(\tilde{x},\tilde{y},P) \xrightarrow{o(\tilde{x},\tilde{y})(n)} P;\bar{o}_n(\tilde{y})$ (Resp-Out) $\bar{o}_n(\tilde{y}) \xrightarrow{\bar{o}_n(\tilde{y})} \mathbf{0}$ (Resp-In) $o_n(\tilde{x}) \xrightarrow{o_n(\tilde{x})} \mathbf{0}$

(Int-Sync) $$\frac{P \xrightarrow{o} P' \ , \ Q \xrightarrow{\bar{o}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$ (Par-Int) $$\frac{P \xrightarrow{\gamma} P'}{P \mid Q \xrightarrow{\gamma} P' \mid Q}$$ (Seq) $$\frac{P \xrightarrow{\gamma} P'}{P;Q \xrightarrow{\gamma} P';Q}$$

(Choice) $$\frac{P \xrightarrow{\gamma} P'}{P + Q \xrightarrow{\gamma} P'}$$ (CongrP) $$\frac{P \equiv_P P' \ , \ Q' \equiv_P Q \ , \ P' \xrightarrow{\gamma} Q'}{P \xrightarrow{\gamma} Q}$$

(Structural Congruenge over P)

$$P + \mathbf{0} \equiv_P P \qquad P \mid \mathbf{0} \equiv_P P \qquad \mathbf{0};P \equiv_P P \qquad (P + Q) \equiv_P (Q + P)$$
$$(P \mid Q) \equiv_P (Q \mid P) \qquad (P + Q) + R \equiv_P P + (Q + R) \qquad (P \mid Q) \mid R \equiv_P P \mid (Q \mid R)$$

(Rules over E)

(One-WaySync) $$\frac{[P]_{id} \xrightarrow{\bar{o}(\tilde{x})} [P']_{id} \ , \ [Q]_{id'} \xrightarrow{o(\tilde{y})} [Q']_{id'} \ , \sigma = (id, id', o, \tilde{x}, \tilde{y}, \uparrow)}{[P]_{id} \parallel [Q]_{id'} \xrightarrow{\sigma} [P']_{id} \parallel [Q']_{id'}}$$

(Req-Sync) $$\frac{[P]_{id} \xrightarrow{\bar{o}(\tilde{z},\tilde{k})(n)} [P']_{id} \ , \ [Q]_{id'} \xrightarrow{o(\tilde{x},\tilde{y})(n)} [Q']_{id'} \ , n \ fresh, \sigma = (id, id', o, \tilde{z}, \tilde{x}, \uparrow)}{[P]_{id} \parallel [Q]_{id'} \xrightarrow{\sigma} [P']_{id} \parallel [Q']_{id'}}$$

(Resp-Sync) $$\frac{[P]_{id} \xrightarrow{o_n(\tilde{k})} [P']_{id} \ , \ [Q]_{id'} \xrightarrow{\bar{o}_n(\tilde{y})} [Q']_{id'} \ , \sigma = (id, id', o, \tilde{k}, \tilde{y}, \downarrow)}{[P]_{id} \parallel [Q]_{id'} \xrightarrow{\sigma} [P']_{id} \parallel [Q']_{id'}}$$

(Par-Ext) $$\frac{E_1 \xrightarrow{\gamma} E_1'}{E_1 \parallel E_2 \xrightarrow{\gamma} E_1' \parallel E_2}$$ (CongrE) $$\frac{E_1 \equiv E_1' \ , \ E_2' \equiv E_2 \ , \ E_1' \xrightarrow{\gamma} E_2'}{E_1 \xrightarrow{\gamma} E_2}$$ (Int-Ext) $$\frac{P \xrightarrow{\gamma} P'}{[P]_{id} \xrightarrow{\gamma} [P']_{id}}$$

(Structural Congruence over E)

$$\frac{P \equiv_P Q}{[P]_{id} \equiv [Q]_{id}} \qquad E_1 \parallel E_2 \equiv E_2 \parallel E_1 \qquad E_1 \parallel (E_2 \parallel E_3) \equiv (E_1 \parallel E_2) \parallel E_3$$

defined in rule Req-Resp. Considering the axiom Req-Out and Req-In indeed, the Request-Response primitives will be transformed into two One-Way (invocation and reception) identified by the label n which is unique and univocally determined during the synchronization. It is worth noting that all the synchronizations which are performed between different orchestrators are labelled with an action σ. This fact will be fundamental for the definition of the conformance notion presented in the next section. Par-Ext deals with external parallel composition and CongrE is for external structural congruence denoted by $\equiv$. Int-Ext expresses the fact that an orchestrator behaves accordingly with its internal processes.

4 Conformance Between Choreography and Orchestration

Our proposal defines a conformance notion based on a bisimulation-like relation between the labelled transition system of choreography and the labelled transition system of the orchestrated system where orchestrators are associated to roles. Such a kind of machinery allows us to test if all the interactions performed by the orchestrated system are coherent with the conversations expressed in the choreography. Furthermore, it guarantees, by exploiting the fact that the name of the variables must be the same, that the data flow of the orchestrated system is conformant with that expressed by choreography conversations. In particular, let C be a choreography where Con represents the conversation rules and let E be an orchestrated system. We define a *joining function*, named Ψ, for associating the orchestrators of E to the roles of C and we test the conformance, up to Ψ, of E and C by using a bisimulation-like relation where the σ labels of the former are compared with the μ ones of the latter.

Definition 1 (joining function). *A joining function is an element of the set* $\{\Psi \mid \Psi : ID \to RName \cup \{\bot\}\}$ *containing functions which associate to each orchestrator identifier id a choreography role* ρ *or the* $\bot$ *value.*

Given a joining function Ψ and an action $\sigma = (id, id', o, \tilde{x}, \tilde{y}, dir)$ of a given orchestrated system where id and id' are orchestrator identifiers, o is an operation, $\tilde{x}$ and $\tilde{y}$ are tuples of variables and $dir \in \{\uparrow, \downarrow\}$, we denote with $\Psi[\sigma] = (\Psi(id), \Psi(id'), o, \tilde{x}, \tilde{y}, dir)$ the renaming of the orchestrator identifiers with the joined roles.

Now we introduce the conformance notion, namely *conformability bisimulation*, between an OL system and a CL_P one. It is based on a relation which resembles branching bisimulation and it tests that, given a joining function Ψ, all the conversations σ produced by the OL system are equal to the μ produced by the CL_P one excluding τ actions.

Definition 2 (Conformability bisimulation). *Let* Ψ *be a joining function. A relation* $\mathcal{R}_\Psi \subseteq (CL_P \times OL)$ *is a conformability bisimulation if* $(C, E) \in \mathcal{R}_\Psi$ *implies that for all* $\mu \in Act_C$ *and for all* $\sigma \in Act_{OL}$*:*

1. $C \xrightarrow{\mu} C' \Rightarrow E \xrightarrow{\tau}{}^{*} E' \wedge E' \xrightarrow{\sigma} E'' \wedge (C', E'') \in \mathcal{R}_\Psi \wedge \Psi[\sigma] = \mu$
2. $E \xrightarrow{\tau} E' \Rightarrow (C, E') \in \mathcal{R}_\Psi$
3. $E \xrightarrow{\sigma} E' \Rightarrow C \xrightarrow{\mu} C' \wedge (C', E') \in \mathcal{R}_\Psi \wedge \Psi[\sigma] = \mu$

We write $C \triangleright_\Psi E$ if there exists a conformability bisimulation $\mathcal{R}_\Psi$ such that $(C, E) \in \mathcal{R}_\Psi$.

Such a kind of notion is not enough for defining conformance because it is possible that there exist synchronizations between orchestrators on operations which are not considered in choreography. Furthermore, there could be orchestrators which are not joined with the roles of the choreography (this case corresponds to those identifiers that Ψ maps in $\perp$) and which are used for coordinating the others. To the end of conformance, only the interactions which are performed on the operations within the choreography roles and which involve only orchestrators joined with roles must be considered relevant (i.e. observable). The following definition defines the notion of conformance between a choreography and an OL system exploiting a hiding operator which makes observable those interactions which contain operations included in the choreography and orchestrators joined with roles.

Definition 3 (Conformance Notion). *Given a choreography $C = (Con, \Sigma)$, an orchestrated system $E \in OL$ and a joining function Ψ such that $Im(\Psi) = \Sigma_\rho \cup \{\perp\}$[1] where Σ_ρ is the set of role names contained in Σ, let ω_C be the set of the operations involved within the choreography C, let ω_o be the set of operations exhibited by the processes of E and let $E_{OP} = \omega_o \backslash \omega_C$ be the set of operations exhibited by E and which do not appear within the roles of C. Let $E_\perp$ be the set of orchestrator identifiers id of E for which $\Psi(id) = \perp$. We say that E is conformant to C if the following condition holds:*

$$C \triangleright_\Psi E/E_{OP}//E_\perp$$

where $/E_{OP}$ is a hiding operator which hides (replaces with τ moves) all the transitions which contain operations contained in E_{OP} and $//E_\perp$ is a hiding operator which hides all the transitions which contain orchestrators not joined with any role.

5 Example

Here we reason about the meaning of conformance by using an example. Let us now consider a business scenario where a customer invokes a store service in order to buy a good and where, depending on the customer's credit card type, the store service will invoke the respective payment service. In order to define the choreography let us consider four roles: ρ_C which represents the customer behaviour, ρ_S which represents the store service, ρ_V which represents the VISA

[1] $Im(\Psi) = \{\Psi(id) \mid id \in ID\}$.

payment service and ρ_{AE} which represents the American Express payment service. For each role we define the following operations:

$$\omega_C = \{(\text{RECEIPT}, ow)\}, \quad \omega_S = \{(\text{BUY}, rr)\},$$
$$\omega_V = \{(\text{PAY-VISA}, rr)\}, \quad \omega_{AE} = \{(\text{PAY-AE}, rr)\}.$$

Referring to the role definitions described in Section 2 the role ρ_C exhibits a One-Way operation named RECEIPT whereas the other roles exhibit a Request-Response operations (respectively BUY, PAY-VISA, PAY-AE).

Let $type_C$ and $type_S$ be the variables which hold the credit card type of role ρ_C and ρ_S respectively, let $\tilde{d}_C$, $\tilde{d}_S$, $\tilde{d}_V$ and $\tilde{d}_{AE}$ be the tuples of variables representing the customer data respectively owned by each role and let ack_C, ack_S, ack_V and ack_{AE} be the variables used for modelling the response information into the Request-Response operations. We denote with $\circ$ (e.g. $\tilde{d}_C \circ type_C$) the concatenation of tuples. We define the variable sets of each role in the following way:

$$V_C = \left\{type_C, \tilde{d}_C, ack_C\right\}, V_S = \left\{type_S, \tilde{d}_S, ack_S\right\},$$
$$V_V = \left\{\tilde{d}_V, ack_V\right\}, V_{AE} = \left\{\tilde{d}_{AE}, ack_{AE}\right\}$$

Let Σ be the set of roles defined in the following way:

$$\Sigma = \{(\rho_C, \omega_C, V_C), (\rho_S, \omega_S, V_S), (\rho_V, \omega_V, V_V), (\rho_{AE}, \omega_{AE}, V_{AE})\}.$$

Let Con be the following conversation:

$Con ::= CustBuyReq;$
$\quad (VisaPay; CustBuyResp\ ; VisaReceipt$
$\quad +$
$\quad AEPay; CustBuyResp\ ; AEReceipt)$
$CustBuyReq ::= (\rho_C, \rho_S, \text{BUY}, \tilde{d}_C \circ type_C, \tilde{d}_S \circ type_S, \uparrow)$
$CustBuyResp ::= (\rho_C, \rho_S, \text{BUY}, ack_C, ack_S, \downarrow)$
$VisaPay ::=$
$\quad (\rho_S, \rho_V, \text{PAY-VISA}, \tilde{d}_S, \tilde{d}_V, \uparrow); (\rho_S, \rho_V, \text{PAY-VISA}, ack_S, ack_V, \downarrow)$
$VisaReceipt ::= (\rho_V, \rho_C, \text{RECEIPT}, \tilde{d}_V, \tilde{d}_C, \uparrow)$
$AEPay ::=$
$\quad (\rho_S, \rho_{AE}, \text{PAY-AE}, \tilde{d}_S, \tilde{d}_{AE}, \uparrow); (\rho_S, \rho_{AE}, \text{PAY-AE}, ack_S, ack_{AE}, \downarrow)$
$AEReceipt ::= (\rho_{AE}, \rho_C, \text{RECEIPT}, \tilde{d}_{AE}, \tilde{d}_C, \uparrow)$

In Fig. 1 are graphically represented the interactions among the roles set by Con without showing the order they are performed. The circles are the roles with their own variables depicted inside, the bold segments are the operations and the arrows are the interactions. $CustBuyReq$ is the request interaction from the customer to the store service for purchasing a good. Depending on the information held by the variable $type_S$ the Visa payment interaction $VisaPay$ or the American Express one $AEPay$, will be enabled. After the payment is performed the store service can send the response to the customer ($CustBuyResp$).

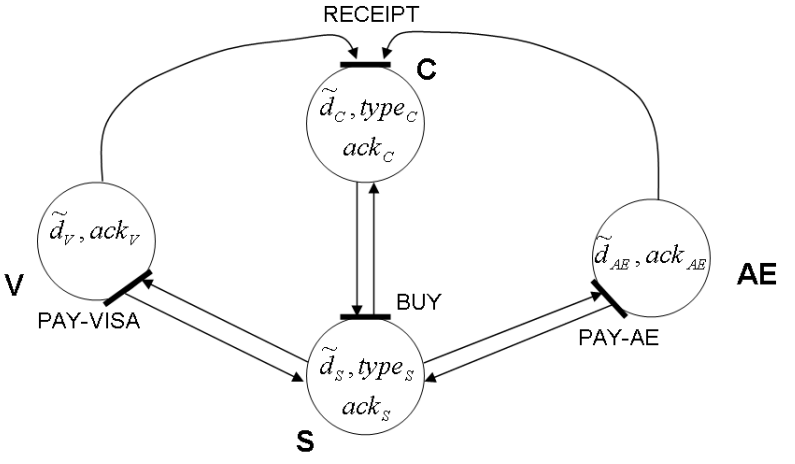

Fig. 1. Interactions among the roles

At the end the receipt from the credit card agency can be sent to the customer ($VisaReceipt$ or $AEReceipt$). We consider the choreography $C = (Con, \Sigma)$.

In the following we present two possible orchestrated systems where the former is not conformant with the choreography C and the latter satisfies the notion given in the previous section.

1. We consider an orchestrated system E_1 with four orchestrators: C, S, V and AE whose definition follows:

$$
\begin{aligned}
&E_1 ::= C \parallel S \parallel V \parallel AE\\
&C ::= [\overline{\text{BUY}}(\tilde{d}_C \circ type_C, ack_C) \mid \text{RECEIPT}(\tilde{d}_C)]_C\\
&S ::= [\text{BUY}(\tilde{d}_S \circ type_S, ack_S, Payment)]_S\\
&\quad Payment ::= (\overline{\text{PAY-VISA}}(\tilde{d}_S, ack_S) + \overline{\text{PAY-AE}}(\tilde{d}_S, ack_S))\\
&V ::= [\text{PAY-VISA}(\tilde{d}_V, ack_V); \overline{\text{RECEIPT}}(\tilde{d}_V)]_V\\
&AE ::= [\text{PAY-AE}(\tilde{d}_{AE}, ack_{AE}); \overline{\text{RECEIPT}}(\tilde{d}_{AE})]_{AE}
\end{aligned}
$$

We consider a joining function Ψ where C, S, V and AE embody roles ρ_C, ρ_S, ρ_V and ρ_{AE}, that is:

$$
\begin{aligned}
&\Psi(C) = \rho_C, \Psi(S) = \rho_S, \Psi(V) = \rho_V, \Psi(AE) = \rho_{AE},\\
&\Psi(id) = \perp \; for \; id \notin \{C, S, V, AE\}\,.
\end{aligned}
$$

Analysing system E_1 is possible to note that it is not conformant with the choreography C because of the RECEIPT operation. Indeed, there exist some paths where the interaction on the RECEIPT operation is performed before the response interaction on the BUY operation. Such a kind of behaviour is in contrast with that of choreography C where the receipt interaction must be performed after the response interaction on the BUY operation. For this reason the conformance notion is not satisfied. **2.** We consider a system E_2 where there

is an additional orchestrator whose identifier is B which is not joined with any role of the choreography and which is a bank service. The store service invokes the bank service when it has performed the credit card request (PAY-VISA or PAY-AE) for charging its transactions. The bank service will invoke the respective credit card service for charging the purchase order of the customer whose behaviour is the same of previous case E_1 as well as for the *Payment* process. At the end the credit card service will send the receipt to the customer.

$$
\begin{aligned}
&E_2 ::= C \parallel S \parallel V \parallel AE \parallel B\\
&S ::= [\mathrm{BUY}(\tilde{d}_S \circ type_S, ack_S, Payment); \overline{\mathrm{CHARGE}}(\tilde{d}_S)]_S\\
&V ::= [(\mathrm{PAY\text{-}VISA}(\tilde{d}_V, ack_V); \mathrm{CH\text{-}VISA}(\tilde{d}_V); \overline{\mathrm{RECEIPT}}(\tilde{d}_V)]_V\\
&AE ::= [\mathrm{PAY\text{-}AE}(\tilde{d}_{AE}, ack_{AE}); \mathrm{CH\text{-}AE}(\tilde{d}_{AE}); \overline{\mathrm{RECEIPT}}(\tilde{d}_{AE})]_{AE}\\
&B ::= [\mathrm{CHARGE}(\tilde{d}_B); (\overline{\mathrm{CH\text{-}VISA}}(\tilde{d}_B) + \overline{\mathrm{CH\text{-}AE}}(\tilde{d}_B))]_B
\end{aligned}
$$

We consider the same joining function Ψ of the example above. E_2 is conformant to the choreography C. Indeed, differently from E_1, the receipt interaction always follows the response on the BUY operation. This condition is guaranteed by the fact that the receipt interactions from the two credit card orchestrators are blocked by the CH-VISA and the CH-AE operations which are performed after the CHARGE operation in B. Indeed, the CHARGE operation in S is invoked after the response on the BUY operation.

It is important to note that the operations CHARGE, CH-VISA and CH-AE are not described in the choreography C. Considering the conformance notion given at Section 4 all the interactions which involve such a kind of operations are hidden in τ actions by using the operator $/E_{2OP}$ where $E_{2OP} = \{\mathrm{CHARGE}, \mathrm{CH\text{-}VISA}, \mathrm{CH\text{-}AE}\}$. In the same way, all the interactions which involve the orchestrator B must be hidden by using the operator $/E_{2\perp}$ where $E_{2\perp} = \{B\}$. Thus we can write that: $C \triangleright_\Psi E_2/E_{2OP}/E_{2\perp}$.

6 Conclusions

In this work we have formalized a notion of conformance between choreography and orchestration. To this end we have exploited previous work where the choreography language CL_P were defined, and we have introduced here a language for orchestration equipped with a formal semantics. Choreography abstracts away from some aspects (e.g., coordination activities that should be performed by roles to preserve the conversation rules) while orchestrated systems can be seen as a further development step of systems described by choreography which must take into account also coordination activities between roles. We consider that the usage of a bisimulation-like technique, of the hiding operators and of the joining function between orchestrator identifiers and roles permits to capture such a relationship.

The notion of conformance between the choreography and the orchestration languages we introduce in this paper provides a methodology to approach both the system design and the development phase which allows us to verify whether

the implementation, that is the orchestrated system, behaves accordingly with the conversation rules of the choreography. As future work we intend on one hand to extend the two languages by introducing the notion of variables state and, on the other hand, to investigate how to capture the conformance between choreographies and more structured solutions at the level of orchestrated systems. Variables states can be used to express behaviours depending on the values stored in the variables and, on the orchestration side, to be closer to executable languages and in particular with the executable fragment of WS-BPEL. As for the conformance we intend to refine the notion in order to perform conformance tests also in cases where, for instance, more orchestration processes embody a certain role, or viceversa.

While there are several works which separately deal with services orchestration or choreography that we list below, to the best of our knowledge the only related work is [BBM+05] where, by means of automaton, defines a conformance notion which allows us to test whether interoperability is guaranteed. Such a notion is limited to systems involving only two peers.

Papers [BMM05, BHF04, LZ05, ML04] investigate mechanisms for supporting long running transactions in orchestration languages and security issues have been investigated in [BFG04]. Another aspect related with orchestration is the correlation sets mechanism (used, e.g., by WS-BPEL) which provides a mean for correlating some interactions between services; such a mechanism have been formalized in an orchestration language in [Vir04]. Besides our work on choreography we have mentioned in the paper, here we cite [BCPV04] where the Web Service Choreography Interface (WSCI) language is modeled.

References

[BBM+05] M. Baldoni, C. Badoglio, A. Martelli, V. Patti, and C. Schifanella. Verifying the conformance of web services to global interaction protocols: a first step. In *Proc. of Web Services and Formal Methods Workshop (WS-FM'05)*, volume 3670 of *LNCS*, pages 257–271. Springer-Verlag, 2005.

[BCPV04] 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.

[BFG04] 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.

[BGG+] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Towards a formal framework for Choreography. In *Proc. of 3rd International Workshop on Distributed and Mobile Collaboration (DMC 2005)*. IEEE Computer Society Press. To appear. [http://www.cs.unibo.it/%7Elucchi/papers/dmc.pdf].

[BHF04] Michael Butler, C. A. R. Hoare, and Carla Ferreira. A trace semantics for long-running transactions. In *25 Years Communicating Sequential Processes*, pages 133–150, 2004.

[BMM05] R. Bruni, H. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In *POPL '05: Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 209–220, New York, NY, USA, 2005. ACM Press.

[GGL05] R. Gorrieri, C. Guidi, and R. Lucchi. Reasoning on the interaction patterns in choreography. In *Proc. of Web Services and Formal Methods Workshop (WS-FM'05)*, volume 3670 of *LNCS*, pages 333–348. Springer-Verlag, 2005.

[Kel76] Robert M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, 1976.

[LZ05] C. Laneve and G. Zavattaro. Foundations of Web Transactions. In *Proc. of International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *LNCS*, pages 282–298, 2005.

[ML04] 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.

[OAS] OASIS. *Web Services Business Process Execution Language Version 2.0, Working Draft*. [http://www.oasis-open.org/committees/download.php/10347/wsbpel-specification-draft-120204.htm].

[vGW96] Rob J. van Gabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.

[Vir04] 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.

[W3C] W3C. *Web Services Choreography Description Language Version 1.0. Working draft 17 December 2004*. [http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/].