# View Driven Federation of Choreographies

Amirreza Tahamtan[1] and Johann Eder[2]

[1] Vienna University of Technology, Dept. of Software Technology & Interactive Systems, Information & Software Engineering Group
tahamtan@ifs.tuwien.ac.at
[2] Alpen-Adria University of Klagenfurt, Dept. of Informatic-Systems, Austria
eder@isys.uni-klu.ac.at

**Abstract.** We propose a layered architecture for choreographies and orchestrations of web services. The proposed architecture uses the concept of process views. The distributed nature of the model and the concept of views improve the privacy of business partners but do not limit their interaction capabilities, an essential feature in B2B and interorganizational applications. Our approach enables description of business processes in different levels of detail with a uniform modeling language and is fully distributed.

**Keywords:** Web Service Composition, Choreography, Orchestration, Process View, Interorganizational Process.

## 1 Introduction

Web Services enable application development and integration over the Web by supporting interactions within and across the boundaries of cooperating partner organizations. Two mostly used concepts in the realm of Web Service composition are choreographies and orchestrations.

An orchestration belongs to and is controlled by one partner and describes an executable process which is run by its owner. A partner's internal logic and business know-how are contained in his orchestration. An orchestration is solely visible to its owner and other external partners have no view on and knowledge about this orchestration. An orchestration is a process viewed only from the perspective of its owner. Different languages such as WS-BPEL executable process [1] or BPML [2] can be used for definition of orchestrations.

On the other hand, a choreography is a non-executable, abstract process that defines the message exchange protocol between partners. A choreography defines the collaboration among involved partners. Exchanged messages are visible to all participants of a choreography. External parties who are not part of a choreography are not able to view and monitor the messages and have no view on the choreography. A choreography has no owner or a super user in charge of control and all involved partners are treated equally. A choreography is a process definition from a global perspective shared among all involved partners [13].

N.T. Nguyen et al. (Eds.): Adv. in Intelligent Inform. and Database Systems, SCI 283, pp. 145–156.
springerlink.com © Springer-Verlag Berlin Heidelberg 2010

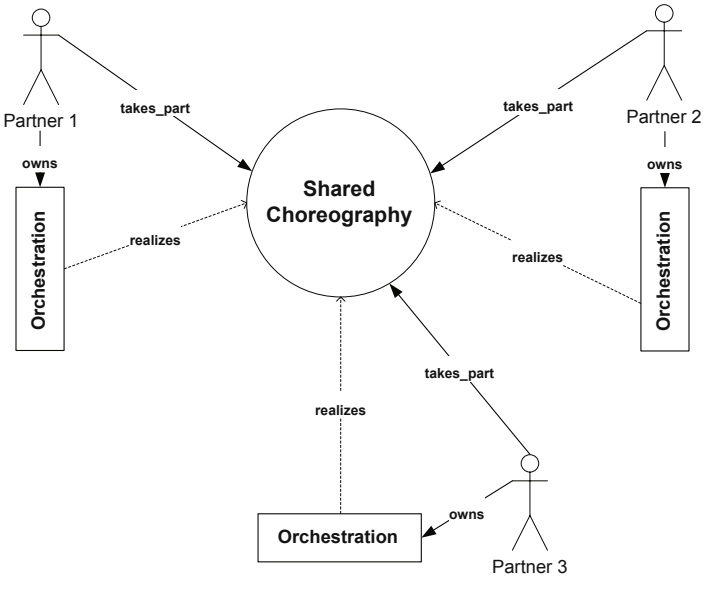


**Fig. 1.** A typical scenario of Web Service composition

A typical scenario [3,5,6] of web service composition assumes one choreography shared among several partners where each partner realizes its parts of the choreography in its orchestration. The shared choreography defines the communication among orchestrations. This scenario is depicted in figure 1.

Imagine a procurement scenario, whose participants are a buyer, a seller and a shipper. Figure 2 shows such a choreography. Note that in this work choreohraphies and orchestrations are modeled as workflows, where nodes represent activities and edges the dependencies between activities. For the meta model refer to section 3. This process represents a simple scenario. A real life business process is more complex including e.g exception handling mechanisms. The partners' orchestrations have additional activities which are not contained in the shared choreography. The buyer's orchestration is depicted in figure 3. The buyer before making a request for quote, searches for available sellers for his requested item and consequently selects one, activities *Search sellers* and *Select seller* in the buyer's orchestration.

## 2 View Driven Federated Choreographies

The typical scenario explained above, one shared choreography and a set of private orchestrations, misses an important facet. Presence of only one choreography is not fully adequate for all real life applications. Imagine a web shopping scenario. When shopping online a buyer takes part in a choreography whose partners are a buyer, a seller company like Amazon , a credit card

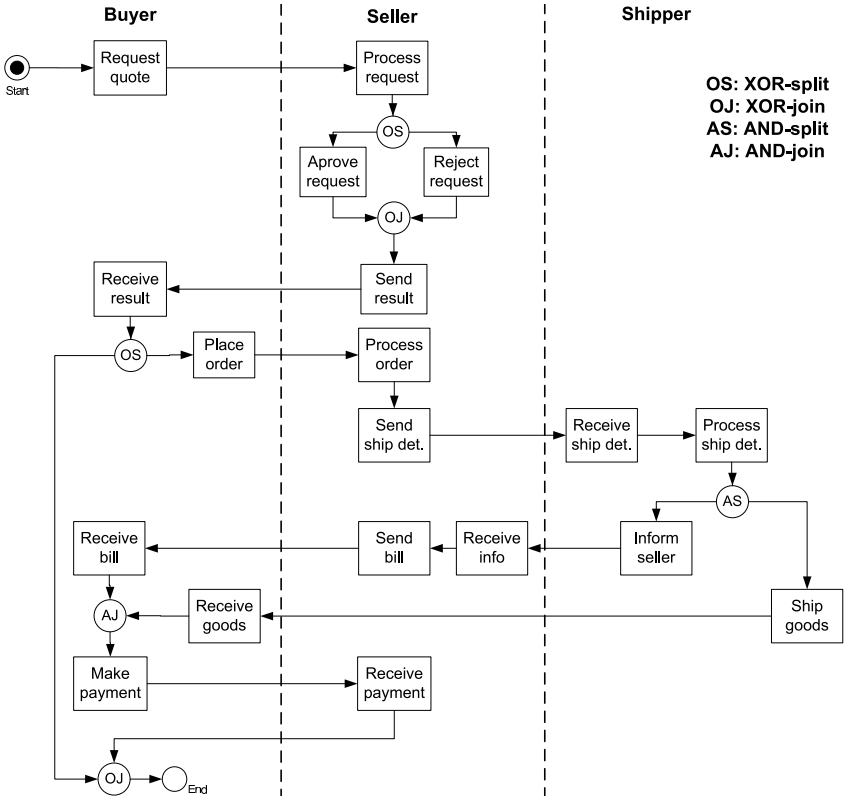


**Fig. 2.** The shared choreography between buyer, seller and shipper

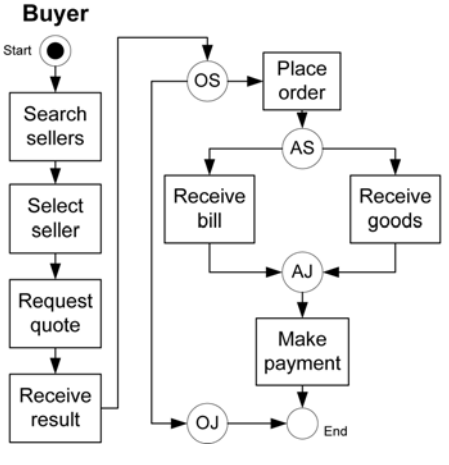


**Fig. 3.** The buyer's orchestration

provider like Visa and a shipper company like FedEx. The buyer knows the following partners and steps: the buyer orders something at the seller, pays by a credit card and expects to receive the items from a shipper. At the same time the seller takes part in several other choreographies which are not visible to the buyer, e.g. the seller and the credit card company are involved in a process for handling payment through a bank. Furthermore, the seller and the shipper realize another protocol they agreed upon containing other actions such as money transfer from the seller's bank to the shipper for balancing shipment charges. As this example shows, more than one choreography may be needed for reaching the goals of a business process. Besides, two partners involved in one choreography may also take part in another choreography that is not visible to other partners of the choreography, however essential for the realization of business goals. All these choreographies overlap in some parts but cannot be composed into a single global choreography. Moreover, such choreographies must be realized by orchestrations of partners that take part in them. In the above example the seller implements an orchestration enacting the different interaction protocols with the buyer, the shipper and the credit card company. Even if the combination of all choreographies into one choreography would be possible, the separation offers obvious advantages. To overcome these restrictions, a new architecture and a nouvelle approach called *view driven federated choreographies*, is proposed. This architecture is an extension of our previous work [7] by the concept of views. Consideration of Views has several advantages:

**Improvement of Privacy:** Views improve the privacy of partners in a business process. By using views, partners can decide which parts of their internal private process can be exposed to external partners and there is no need for exposing the whole internal logic and thus preventing cooperating partners to become competitors.

**Serving different (groups of) partners:** A single private process can have many views and each view can be used for interaction with a partner or a group of partners. By application of views, one single underlying private process can serve different groups of partners.

**Interaction compatibility:** Partners can change and modify their private processes without any effect on the interaction with other partners, as long as the views remain the same, i.e. they can be defined on the new processes as well. In this way the interaction can be performed in a consistent way and there is no need to inform other partners about changes in the private process.

Each partner has an orchestration for realization of its tasks and takes part in one or more choreographies. Each partner has a view on his orchestration. A view is not only a view on an orchestration but also a view on the shared choreography and identifies the parts that belong to a specific partner of the shared choreography which this partner is in charge of its realization. By realization of the activities that belong to a partner, the partner has a


**Fig. 4.** The main idea of the view driven federated choreographies

conformant behavior with respect to the agreed upon choreography. A view shows a single partner's perspective on the choreography and can be used as a skeleton for designing the partner's orchestrations by adding other internal tasks. In other words they show the minimum amount of task as well as the structure of the tasks that a partner's orchestration must contain in order to be conformant with the shared choreography. For a more detailed discussion on views and how correct views can be constructed refer to [14].

The main idea of the view driven federated choreographies is presented in figure 4. It consists of two layers. The upper layer consists of the federated choreographies shared between different partners, e.g. in figure 4 *Purchase processing choreography* is shared between buyer, seller and shipper. A choreography is composed of views of the orchestrations by which the choreography is (partially) realized. In other words, the activities contained in a choreography are only those in the views. A choreography may support another

choreography. This means the former, the supporting choreography, contributes to the latter, the supported choreography, and partially elaborates it. E.g. *Shipment processing choreography* is the supporting choreography and *Purchase processing choreography* is the supported choreography. The set of activities contained in a supporting choreography is an extended subset of the activities of the supported choreography. The supporting choreography describes parts of the supported choreography in more detail. The choreography which supports no other choreography and is only supported by other choreographies is called the *global choreography*, in our running example *Purchase processing choreography* is the global choreography. Informally, the global choreography captures the core of a business process and other choreographies which support the global choreography describe parts of the global choreography in the needed detail for implementation. In figure 4 the global choreography, *Purchase processing choreography*, describes how an item is sold and shipped to the buyer. It contains the activities and steps which are interesting for the buyer and the buyer needs to know them in order to take part in or initiate the business process. How shipping of the items and debiting the buyer's credit card is handled in reality are described in the *Shipment processing choreography* and the *Payment processing choreography* respectively.

The bottom layer consists of orchestrations that realize the choreographies in the upper layer. Each orchestration provides several views for different interactions with other partners. The interactions with other partners are reflected in the choreographies. Hence, an orchestration needs to provide as many views as the number of choreographies this orchestration (partially) realizes. Let figure 2 be the *Purchase processing choreography*. The buyer's orchestration and its shared view with the *Purchase processing choreography* are presented in figures 3 and 5 respectively. Each partner provides its own internal realization of relevant parts of the according choreographies, e.g. buyer has an orchestration which realizes its part in all three choreographies.

The presented approach is fully distributed and there is no need for a centralized coordination. Each partner has local models of all choreographies in which it participates. All local models of the same choreography are identical. By having the identical local models of choreographies, partners know to which activities they have access, which activities they have to execute and in which order. In addition, partners are aware when to expect messages and in which interval they can send messages. In other words, the knowledge about execution of the model is distributed among involved partners and each partner is aware of its duties in the course of process execution. Hence, there is no need for a super-user or a central role that possesses the whole knowledge about execution of the process. Rather this knowledge is distributed among participants and each partner knows what he needs to know. Additionally, each partner holds and runs its own model of the orchestration. Note that if there is a link between two chorographies and/or orchestrations, either a support link between two choreographies or a realize link between a

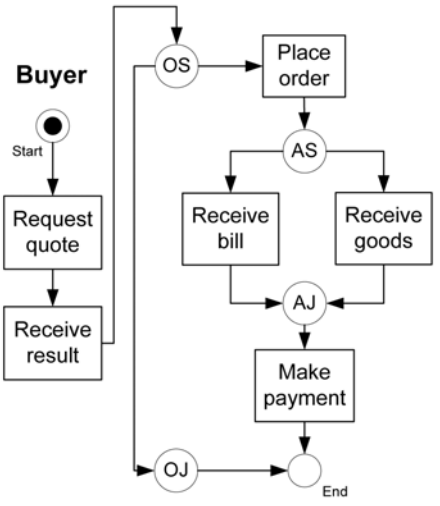


**Fig. 5.** The view on the buyer's orchestration

choreography and an orchestration, it implies that these two choreographies and orchestrations have at least one activity in common. That means that the greatest common divisor of these two choreographies and/or orchestrations is not empty.

In fact, one can argue that supporting choreographies may be combined by the means of composition as described in [11,4] where existing choreography definitions can be reused and recursively combined into more complex choreographies. But, in fact, the relationship between choreographies can be more sophisticated than merely a composition. For example the relationship between the seller and the shipper may include not only the passing of shipment details from the seller to the shipper but also payment of shipment charges through the seller's bank. This can be described by a separate choreography between seller, shipper and the bank (see *Shipment processing choreography* in figure 4). This choreography has additional activities and partners which are not visible in its supported choreography. This choreography contributes to the *Purchase processing choreography* and elaborates the interaction between the seller and the shipper. The *Shipment processing choreography* is illustrated in figure 6.

### 2.1 Advantages of the View Driven Federated Choreographies

View driven federated choreographies are more flexible than typical compositional approaches used in proposals like WS-CDL [11] and it closes the gap between choreographies and orchestrations by providing a coherent and integrated view on both choreographies and orchestrations. View driven federated choreographies offer obvious advantages such as:

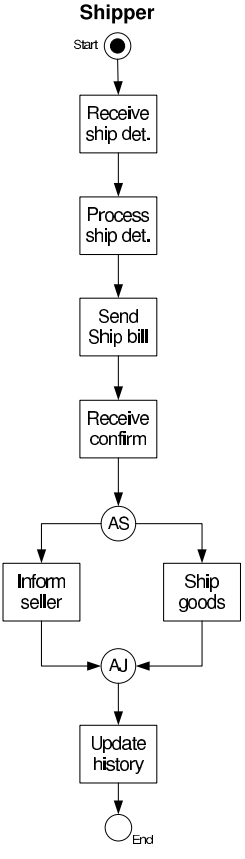


**Fig. 6.** The Shipment processing choreography

**Protection of business know-how:** View driven federated choreographies improve business secrecy and protect business know-how. If the whole business process including all involved partners are modeled as one single choreography, all message exchanges are visible to all partners. But if the interactions are separated into different choreographies, other external observers have no knowledge about the message exchanges and partners can keep the actual handling of their business private.

**Avoidance of unnecessary information:** The proposed approach avoids unnecessary information. Even if there is no need for protection of business know-how, it is desirable to separate choreographies and limit them only to the interested parties.

**Extendability:** The model is extendable, when such a need arises. As long as the conformance conditions are satisfied, the model can be extended and there is no need to interfere with the running process and notifying

the partners for setting up new choreographies. The conformance issues are discussed briefly in subsection 2.2.

**Uniform modeling:** Finally, View Driven federated choreographies use a coherent and uniform modeling for both choreographies and orchestrations and eliminates the need for different modeling languages and techniques for choreographies and orchestrations. The uniform modeling technique reduces the cost of process at design phase.

### 2.2 Conformance Issues

The central requirement of the proposed model is inter-layer conformance (between choreographies and orchestrations) as well as intra-layer conformance i.e. inside the choreographies and orchestrations layers itself. This includes structural conformance [8], temporal conformance [10,9], messaging conformance and dataflow conformance. Conformance issues are out of scope of this work. For structural and temporal conformance please refer to the references in this subsection as well as [14].

## 3 Metamodel of the View Driven Federated Choreographies

Choreographies, orchestrations and views are treated as workflows. Therefore, choreographies, orchestrations and views can be modeled using typical workflow control flow structures. Moreover, our approach provides a coherent view on both choreographies and orchestrations and their mutual relationships, thus bridging the gap between abstract and executable processes. An orchestration provides several views. Choreographies can be federated into more complex ones. Choreographies are composed of views. Moreover, as all choreographies are workflows, they can be composed out of other choreographies by means of complex activities and control structures available in the workflow models. The same applies to orchestrations. The metamodel allows to describe several choreographies and orchestrations on different levels of detail. Choreographies and orchestrations can share the same activities. These activities are contained in a view that is provided by the orchestration. Such a view identifies which activities of the choreography must be realized in the orchestration. An activity visible in one choreography can be extended by its relationships with other activities in a federated choreography. On the other hand, an activity visible in a choreography can have a complex implementation described in an orchestration. Thus, choreographies and orchestrations together with their activities can be viewed on different levels of detail and in context of different relationships. The metamodel of the view driven federated choreographies is represented in figure 7.

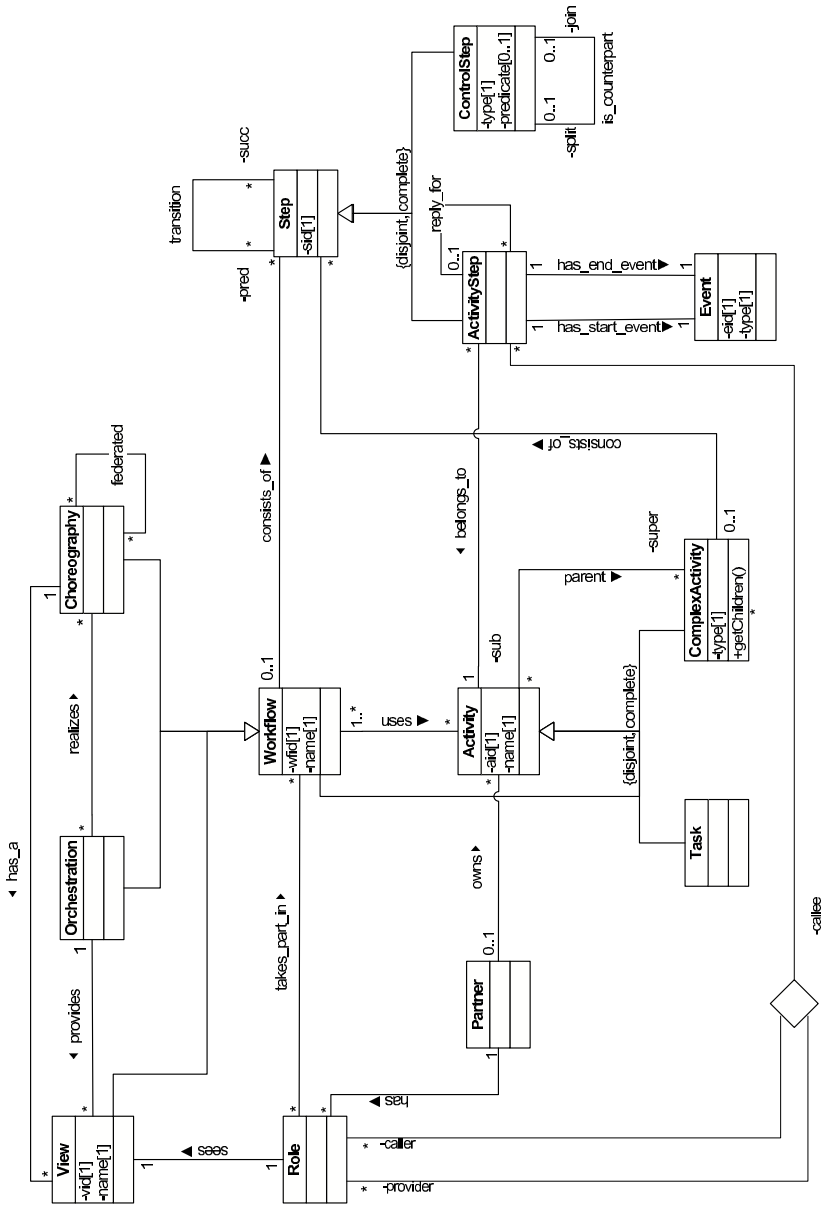


**Fig. 7.** Metamodel of view driven federated choreographies

A workflow is either a *choreography*, an *orchestration* or a *workflow view*. A workflow can have many *views*. A workflow defines views for different roles (of partners). Each role sees and accesses the workflow through the view. A workflow uses *activities*. An activity is either a *task* or a *complex activity* . An activity can be used to compose complex activities. An activity occurrence in such a composition is represented by an *activity step*. One activity can be represented by several activity steps in one or several workflows or complex activities and each activity step belongs to exactly one activity. In other words, activity steps are placeholders for reusable activities. The same activity can occur in different workflows. The control structure of a complex activity is described by its *type* (*seq* for sequence, *par* for parallel and *cond* for conditional).

An activity may be owned by a *partner*. Orchestrations and tasks must have an owner, whereas choreographies must not have an owner. A partner may have several *roles* and one role can be played by several partners. A role may take part in a workflow and call an activity step in this workflow. An activity step is provided by another role. Thus a single parter can use different roles to participate in a workflow and provide or call activity steps. A role sees and accesses a defined view on the workflow.

The notion of a *step* is very important for the presented metamodel. Both workflows and complex activities consist of steps. Between the subsequent steps there can be a *transition* from a predecessor to a successor which represents control flow dependencies between steps.

A complex activity may be decomposed in a given workflow into steps that constitute this complex activity only if all of the activities corresponding to these steps are also used and visible in this workflow. Therefore, a workflow can be decomposed and analyzed on different levels of detail with complex activities disclosing their content, but without revealing protected information on the implementation of these complex activities. To allow a correct decomposition, a complex activity must have only one activity without any predecessors and only one activity without any successors. The same applies to workflows.

A step can be either an *activity step* or a *control step*. As mentioned above, activity steps are placeholders for reusable activities and each activity step belongs to exactly one activity. Activity steps can be called in a workflow definition. An activity step may be used as a reply for a previous activity step. A single activity step may have several alternative replies.

A control step represents a control flow element such as a split or a join. Conditional and parallel structures are allowed, i.e. the type of a control step is one of the followings: *par-split*, *par-join*, *cond-split* or *cond-join*. An attribute *predicate* is specified only for steps corresponding to a conditional split and represents a conditional predicate. Conditional splits have XOR-semantics. A split control step have a corresponding join control step which is represented by the recursive relation *is_counterpart*. This relation is used to represent well structured workflows [12] where each split node has a corresponding join node of the same type and vice versa.

## 4 Conclusions

We introduced a layered, distributed architecture for web service composition composed of choreographies and orchestrations. The concept of views and distributed nature of this model allow business partners to interact and at the same time protect their business know-how and improve their privacy. Besides, business processes can be described in different levels of detail and with a uniform modeling language, as an executable process in orchestrations and as an abstract process in choreographies and views.

**Acknowledgments.** This work is supported by the European Project WS-Diamond in FP6.STREP and the Austrian projects GATiB and Secure 2.0.

## References

1. Andrews, T., et al.: Business process execution language for web services (bpel4ws), ver. 1.1. BEA, IBM, Microsoft, SAP, Siebel Systems (2003)
2. Arkin, A.: Business process modeling language (bpml), ver. 1.0. Technical report, BPMI (2002), http://www.bpmi.org/downloads/spec_down_bpml.htm
3. Barros, A., Dumas, M., Oaks, P.: A critical overview of the web services choreography description language(ws-cdl). Technical report, Business Process Trends (2005)
4. Burdett, D., Kavantzas, N.: Ws choreography model overview. Technical report, W3C (2004)
5. Decker, G., Overdick, H., Zaha, J.M.: On the suitability of ws-cdl for choreography modeling. In: Proc. of EMISA 2006 (2006)
6. Dijkman, R.M., Dumas, M.: Service-oriented design: A multi-viewpoint approach. Int. J. Cooperative Inf. Syst. 13(4), 337–368 (2004)
7. Eder, J., Lehmann, M., Tahamtan, A.: Choreographies as federations of choreographies and orchestrations. In: Proc. of CoSS 2006 (2006)
8. Eder, J., Lehmann, M., Tahamtan, A.: Conformance test of federated choreographies. In: Proc. of I-ESA 2007 (2007)
9. Eder, J., Pichler, H., Tahamtan, A.: Probabilistic time management of choreographies. In: Proc. of QSWS 2008 (2008)
10. Eder, J., Tahamtan, A.: Temporal conformance of federated choreographies. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 668–675. Springer, Heidelberg (2008)
11. Kavantzas, N.: et al. Web services choreography description language (ws-cdl) 1.0. Technical report, W3C (2004)
12. Kiepuszewski, B., ter Hofstede, A.H.M., Bussler, C.: On structured workflow modelling. In: Wangler, B., Bergman, L.D. (eds.) CAiSE 2000. LNCS, vol. 1789, p. 431. Springer, Heidelberg (2000)
13. Peltz, C.: Web services orchestration and choreography. IEEE Computer 36(10), 46–53 (2003)
14. Tahamtan, A.: Web Service Composition Based Interorganizational Workflows: Modeling and Verification. In: Suedwestdeutscher Verlag fuer Hochschulschriften (2009)