Choreographies as Federations of Choreographies and Orchestrations*

Johann Eder, Marek Lehmann, and Amirreza Tahamtan

University of Vienna
Dept. of Knowledge and Business Engineering
first_name.last_name@univie.ac.at

Abstract. We propose a new conceptual model for choreographies of web-services. Choreographies are seen as virtual workflow models shared among participants. Subsets of these participants might have more refined models known only to them. So we see choreographies actually as federations of process models (choreographies as well as orchestrations). In this paper we discuss this layered concept, and present a metamodel with the following distinguishing features: It is fully distributed and does not require a central or global authority. It captures the control flow and the data flow aspects of the processes. Choreography models can be (re)used in several other choreographies. Additionally, we provide a procedure which checks whether choreographies fit together, i.e. the conformance of the federation relationship between models.

1 Introduction

Web Service composition has received recently a lot of interest, both in academia and industry. It enables software integration within and across boundaries of cooperating organizations. The real added value of Web Services lies in their composition and this is the place where several overlapping and competing standards have been proposed. Proposals like BPEL4WS [3], WS-CDL [8], WSCI [7] address some aspects of both choreography and orchestration, but the distinction between the two is not very clear. Moreover, there is still a big gap between choreography and orchestration models, i.e. it may be very difficult to answer whether a given orchestration really realizes a given choreography. A coherent and integrated view on both choreography and orchestration is still missing.

Web Services are loosely-coupled and autonomous. From this perspective it is essential that an organization is able to define its own business process described as a composed Web Service and control the execution of this process or at least the execution of parts of it. This is the task of the orchestration. On the other hand, the same organization must adjust its internal executable processes to protocols used in cooperative and interorganizational processes described by choreographies. Furthermore, this organization may take part in several choreographies and several choreographies may be reflected as a single orchestration.

* Work partly supported by the Commission of the European Union within the project WS-Diamond in FP6.STREP.

In this paper we propose a new metamodel which gives a consistent and integrated view on both choreographies and orchestrations and their mutual relationships. We use an extension of our workflow metamodel [6,9] which enables convenient and flexible representation and analysis of choreographies and orchestrations. Moreover, we propose a novel approach to combine several choreographies into more complex federated choreographies. Our federated choreographies are more flexible than typical compositional approaches.

2 Federated Choreographies

A Web Service choreography specifies a communication protocol for all involved partners. The outcome of the choreography is a virtual process definition viewed from the global perspective where all partners are treated equally [4]. On the other hand, Web Service orchestration refers to an executable process run by a single partner. It contains information about message exchanges, definition of business logic, execution order of activities and internal actions (e.g. data transformations) [11]. Each of the partners involved in a global choreography realizes the in the internal orchestration only its own parts of this choreography.

This typical scenario [4,11] assumes one choreography shared among all partners and a set of private orchestrations (one for each partner). Besides, two partners involved in one choreography may also take part in another choreography that is not visible to other partners in the first choreography, however essential for the realization of its goal. For example, when shopping online we take part in a choreography, where we know the following partners and steps: we order something at a seller, pay by a credit card and expect to receive the items from a shipper. At the same time the seller takes part in several other choreographies which are not visible to us, e.g. the seller and the shipper realize another protocol they agreed upon containing other actions like money transfer from seller's bank to the shipper for balancing shipment charges. These choreographies overlap in some parts but cannot be composed into a single 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, shipper and the bank.

We propose a new approach where existing choreographies can be combined into a *federated choreography* and extended according to the need. We treat both choreographies and orchestrations as workflows. A choreography coordinates several orchestrations owned by different partners and a single orchestration may realize parts of several choreographies. Choreographies can be federated into more complex ones. Moreover, as all choreographies and orchestrations are workflows, they can be composed out of other choreographies by means of complex activities and control structures available in our workflow models.

The idea of federated choreographies is presented in Fig. 1. It consists of two layers. The first layer consist of federated choreographies shared between different partners, e.g. a *Purchase processing* choreography shared between Buyer, Seller, and Shipper. A choreography may support another choreography. This

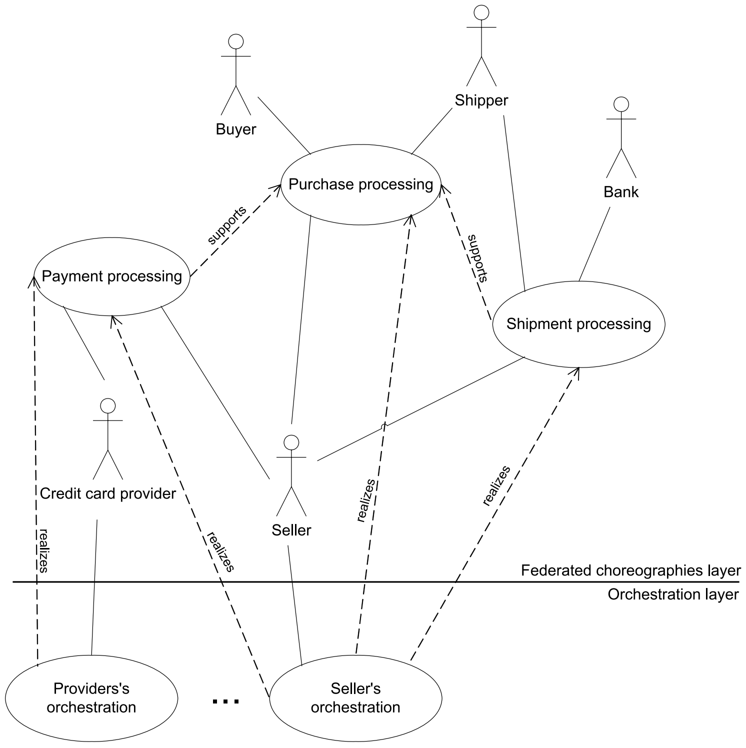

Fig. 1. Federated choreographies

means the former contributes to the latter and partially elaborates it. The second layer consist of orchestrations that realize the choreographies. Each partner provides its own realization of relevant parts of the according choreographies, e.g. the Seller has an orchestration which realizes its part in all three choreographies.

The presented approach is fully distributed. Each partner has local models of all choreographies in which it participates. All local models of the same choreography are identical. Additionally, each partner holds and runs its own model of the orchestration. There is no need for a centralized coordination.

In order to ensure the compatibility of the system, a conformance test between orchestrations and the choreographies that they realize as well as an intra-layer conformance test in the choreography layer is necessary.

The choreography in Fig. 2 describes the "Purchase Processing" choreography in Fig. 1. Three partners participate: a Buyer, a Seller and a Shipper. The Seller upon receiving a request quote from the Buyer answers if the requested item is deliverable or not. If the item is deliverable, the buyer places an order, the seller processes the order and forwards the shipment details to the shipper. The shipper ships the products to the buyer and informs the seller about the details. The seller upon receipt of information sends the bill to the buyer. When the buyer has received both the bill and the ordered goods, makes the payment to the seller. The process terminates upon receipt of the payment by the seller.

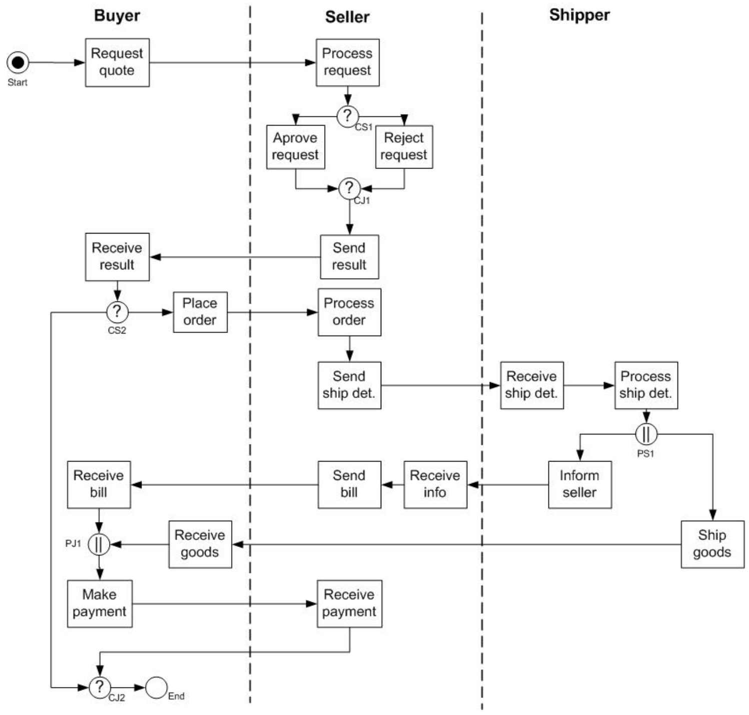

Fig. 2. *Purchase processing* choreography between Buyer, Seller and the Shipper

This choreography involves interactions between the Buyer and the Seller, the Seller and the Shipper and between the Buyer and the Shipper. Each of these interactions can be represented as a separate choreography. These choreographies may be combined by the means of composition as described in [8] where existing choreography definitions can be reused and recursively combined into more complex choreographies. But we claim that a relationship between choreographies can be more sophisticated than merely a composition. In our example the relationship between the Seller and the Shipper includes not only the passing of shipment details from the Seller to the Shipper but it also involves payment of shipment charges through the seller's bank. This is described by a separate choreography between the Seller, Shipper and Bank. This choreography shown in Fig. 3 has additional activities and partners which are not visible in the previous one. This choreography contributes to the "Purchase Processing" and elaborates the interaction between the Seller and Shipper.

Choreographies describe just abstract processes, whereas an orchestration describes an executable process run by one of the partners. Each partner involved in choreographies must provide their realization. The orchestration of the Seller is presented in Fig. 4. It implements the behavior of the Seller in both presented choreographies. Notice that the Shipper and other partners must provide their own private orchestrations which we omit here for the aim of brevity.

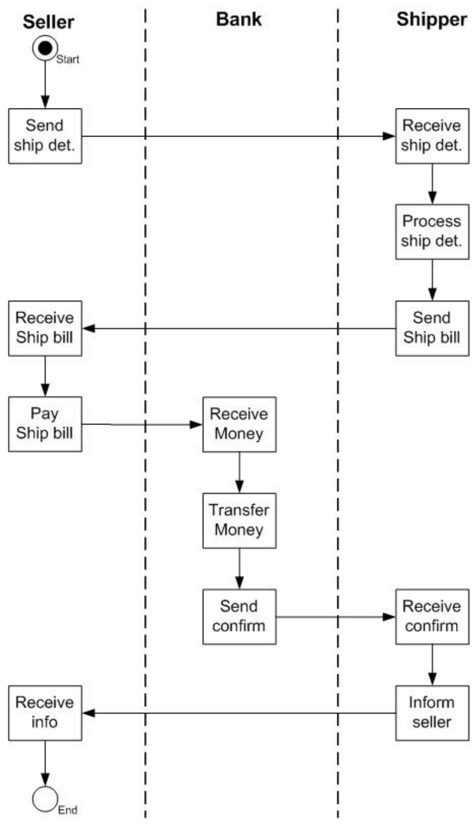

Fig. 3. The *Shipment processing* choreography between Seller, Bank and the Shipper

3 Metamodel

Our metamodel enables the representation of both choreographies and orchestrations which are described as workflow models. Therefore, choreographies and orchestrations can be modeled using typical workflow control flow structures. Moreover, it provides a coherent view on both choreographies and orchestrations and their mutual relationships, thus bridging the gap between abstract and executable processes. The metamodel allows to describe several choreographies on different levels of detail and an orchestration responsible for a private implementation of these choreographies. Choreographies and the orchestration can share the same activities. 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. Our metamodel presented in Fig. 5

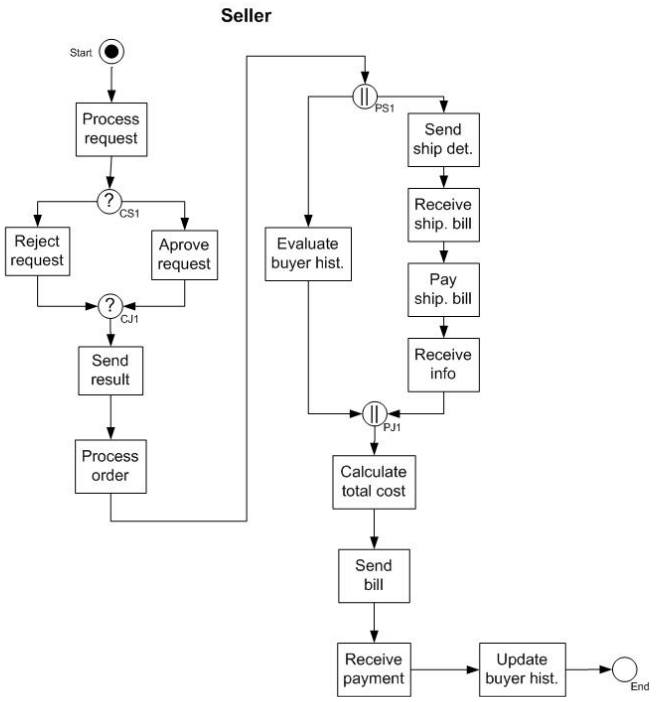

Fig. 4. Seller's Orchestration

focuses only on the control flow aspects of a workflow definition. Because of lack of space we had to omit the data aspects.

A workflow is either a *choreography* or an *orchestration*. A workflow uses *activities*. An activity is either a *task*, a *complex activity* or a *(sub-)workflow*. An activity can be used to compose complex activities and workflows. 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 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 provided by another role. Thus a single parter can use different roles to participate in a 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.

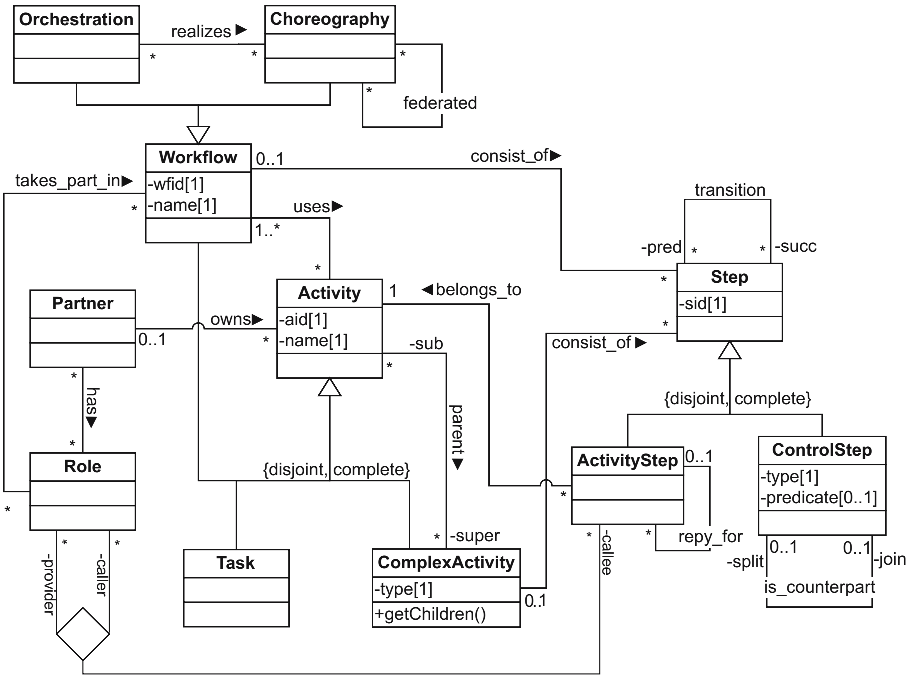

Fig. 5. Metamodel for Federated Choreographies

A complex activity may be decomposed in a given workflow into steps that constitute this complex activity only if all 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. Currently we allow conditional and parallel structures, i.e. the type of a control step is one of the following: *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 may have a corresponding join control step what is represented by the recursive relation *is_counterpart*. This relation is used to represent well structured workflows [6] where each split node has a corresponding join node of the same type and vice versa.

A workflow model can be represented as a directed graph with two kinds of nodes corresponding to activity steps and control steps. The edges correspond to transitions between steps and determine the execution sequence. A complex activity can be decomposed into a subgraph with one input node and one output node. In the graphical notation we use additionally two control nodes: start node and end node places before the start activity and after the end activity of the workflow, respectively. A sample workflow graph is presented in Fig. 2.

The workflow graphs are a convenient and human readable representation of workflow models. For conformance tests we use workflow nets (WF-Nets) [2] which are an extension of classical Petri nets [10] and can be analyzed with all the techniques developed for Petri nets. A WF-Net contains exactly one place without any predecessors (source) and exactly one place without any successors (sink). Moreover, the net graph extended with an additional transition from the sink place to the source place is strongly connected, i.e. for each node n there exists a path from the source place to n and from n to the sink place. We use WF-nets with a set of special transitions added to express branching decisions in a more human readable form: AND-split, AND-join, XOR-split and XOR-join.

The mapping from the presented workflow graphs to WF-Nets is straightforward: start and end control nodes are mapped onto source and sink place respectively; each activity steps is mapped onto a labeled transition with a single input and output place; each split control node is mapped onto a corresponding split transition with a single input place and at least two output places and each join control node is mapped onto a corresponding join transition with at least two input places and exactly one output place. Edges in the original graph are mapped onto dummy transitions connecting subsequent transformed nodes. Finally, dummy transitions can be reduced by Fusion of Series Places (FSP) [10]. A sample mapping of a graph in Fig. 2 is presented in Fig. 6.

4 Conformance Test

Essentially the participating partners are autonomous organizations that may have existing workflows for their orchestrations as well as for their interactions with other organizations and they favor to use the existing workflows instead of designing new ones from scratch and integrate them in the organization. It is pivotal to ensure that utilization of these orchestrations and choreographies lead to no conflict with other choreographies and orchestrations. The basic requirement of the model is the inter and intra-layer conformance of orchestrations and choreographies. This means the orchestration and choreographies must maintain all the assumptions made by the choreography they support or realize. It is possible to check the structural conformance of the system based on the *projection inheritance* [1,5] and notion of *branching bisimulation* [12] as equivalence relation of workflows. The question if two workflows are branching bisimilar is a decidable problem. Because of space limitations we are not able to provide the full description of conformance tests in this paper.

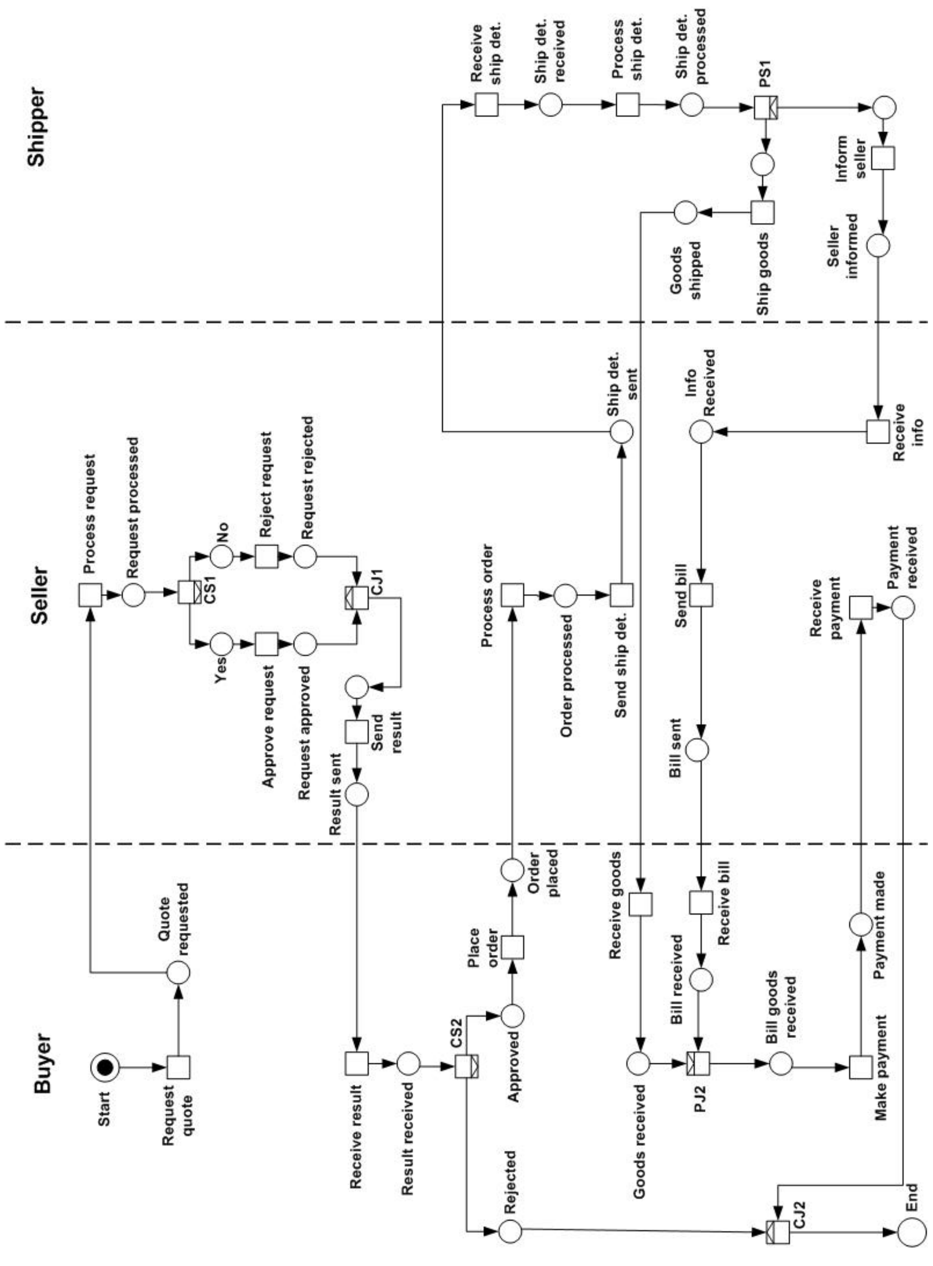

Fig. 6. The WF-Net of workflow graph in Fig. 2

5 Conclusions

Federated choreographies aim at modularizing web service choreographies. Federations can be built top-down - applying the famous "divide and conquer" paradigm - or they can be built bottom-up, re-using existing choreographies for the construction or negotiation of larger choreographies. The meta-model we propose supports all these usages. The automated checking of conformance between choreographies and also orchestrations can be used for federations built top-down as well as for those built bottom-up. Design principles and processes for both ways of constructing choreographies which guarantee conformance are subject of ongoing research.

References

1. Wil M. P. van der Aalst and Twan Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2), 2002.
2. Wil van der Aalst and Kees van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2002.
3. Tony Andrews et al. Business process execution language for web services (bpel4ws). ver. 1.1, BEA, IBM, Microsoft, SAP, Siebel Systems, 2003.
4. Alistair P. Barros, Marlon Dumas, and Phillipa Oaks. Standards for web service choreography and orchestration: Status and perspectives. In *BPM 2005 Workshops*, *LNCS* 3812, Springer, 2005.
5. Twan Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, TU Eindhoven 1998.
6. Johann Eder and Wolfgang Gruber. A meta model for structured workflows supporting workflow transformations. In *ADBIS 2002*, *LNCS* 2435, Springer, 2002.
7. Assaf Arkin et al. Web service choreography interface (wsci) 1.0. W3C, 2002.
8. N. Kavantzas et al. Web services choreography description language (ws-cdl) 1.0. W3C, 2004.
9. Marek Lehmann. *Data Access in Workflow Management Systems.* Number 94 in DISDBIS. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006.
10. Tadao Murata. Petri nets: Properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, April 1989.
11. Chris Peltz. Web services orchestration and choreography. *IEEE Computer*, 36(10):46–52, 2003.
12. Rob J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.