Conformance Test of Federated Choreographies

J. Eder, M. Lehmann, A. Tahamtan

University of Vienna, Dept. of Knowledge and Business Engineering,
Rathausstrasse 19/9, A-1010 Vienna, Austria
johann.eder@univie.ac.at, marek.lehmann@univie.ac.at,
amirreza.tahamtan@univie.ac.at

Abstract

Web Services technology is constantly gaining importance for automation of business processes. A major contribution of this technology is its integration capability, i.e. compositions allowing several autonomous but cooperating web services to implement a business process going beyond the boundaries of a single organization. *Federated choreographies* provide a framework for modular modeling complex collections of choreographies and orchestrations. In this paper we present a conformance test to check the structural conformance of the choreographies and orchestration of the proposed model in order to hold the model structurally consistent. We can formally check whether an orchestration realizing (one part of) a choreography, resp. two related choreographies fit together.

1 Introduction

Web services are seen as a key enabling technology for integrating applications in interorganizational business processes. Web services enable application development and integration over the web by supporting interactions within and across the boundaries of cooperating partner organizations. Web services technology offers composition through choreographies and orchestrations. A web service choreography specifies a communication protocol among involved partners. The outcome of the choreography is a virtual process definition viewed from the global perspective where all partners are treated equally [1]. On the other hand, web service orchestration refers to an executable process run by a single partner [2]. Each of the partners involved in a choreography realizes its own parts of this choreography by its internal orchestration.

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

Several partially overlapping and competing standards for Web Service composition like BPEL4WS [3], WS-CDL [4], WSCI [5] have been proposed. They address some aspects of both choreography and orchestration. 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. Moreover, a single partner may be involved in several different choreographies and may have to implement all of its parts in one orchestration. Therefore, it is important to check whether these choreographies and orchestrations that implement them conform to each other.

In [6] we have proposed a consistent and integrated view on both choreographies and orchestrations and their mutual relationships. Several choreographies may be combined into a more complex federated choreography. Our federated choreographies and orchestrations are more flexible than typical compositional approaches used in proposals like WS-CDL. In this work we present a procedure to check the structural conformance of federated choreographies and orchestrations. The main contribution of this paper is introduction of an algorithm for checking the inter- and intra-layer conformance of the proposed model which ensures the structural conformance of choreographies and orchestrations.

This paper is structured as follows. We start by presenting the metamodel we have proposed to model choreographies and orchestrations (Section 2). We then discuss and explain the need for structural conformance of the model. After formal representation of the concepts, we propose an algorithm for checking the conformance (Section 3). Finally, we draw some conclusions.

2 Federated Choreographies

A typical scenario for web service composition assumes one choreography shared among several partners and a set of private orchestrations (one for each partner) [1, 2, 7]. Each partner sees the same global choreography and no partner has control over the choreography which defines only the message exchange protocol. On the other hand, private orchestration of each partner realizes only those parts of the protocol which belong to this partner in the choreography.

This picture misses an important facet that a partner involved in one choreography may also take part in other choreographies which is not visible to the other participants of the first choreography, however essential for the overall process. 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 credit card company are involved in a process of handling payment. Furthermore, the seller and the shipper realize another protocol they agreed upon containing other tasks like money

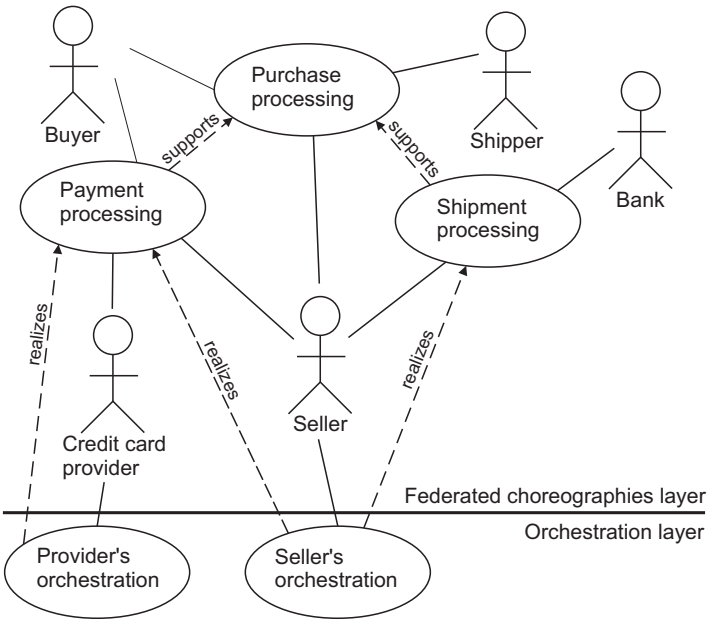

Fig. 1. Federated choreographies

transfer from seller's bank to the shipper for balancing shipment charges. 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 different interaction protocols with the buyer, shipper and the credit card company.

We have proposed a new approach where existing choreographies can be combined into a *federated choreography* and extended according to the need. The idea of federated choreographies is presented in Fig. 1. It has two layers. The first layer consists 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 means the former contributes to the latter and partially elaborates it. The second layer consists of orchestrations that realize the choreographies. Each partner provides its own internal realization of relevant parts of the according choreographies, e.g. the Seller has an orchestration which realizes its part in all three choreographies.

We treat both choreographies and orchestrations as workflows. This is reflected in our metamodel in Fig. 2. Here we describe only the most important parts of the model. For the complete metamodel and a detailed discussion the reader is referred to [6, 8, 9]. The metamodel provides a coherent view on both choreographies and orchestrations and their mutual relationships, bridg-

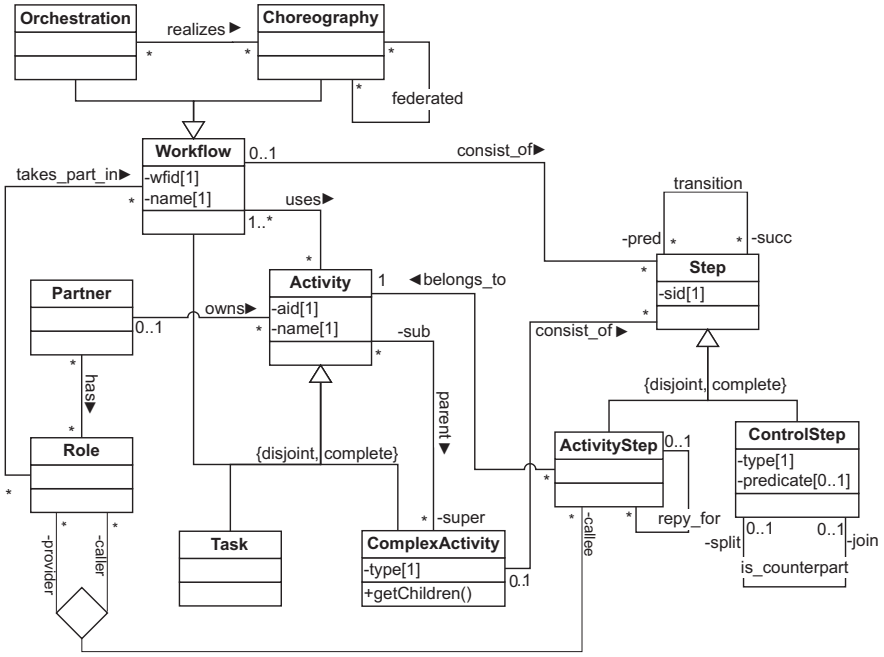

Fig. 2. Metamodel for Federated Choreographies

ing the gap between abstract and executable processes. The activities of one choreography are solely visible to the participatig partners of this choregraphy. Choreographies orchestrations can share the same activities. An activity visible in one choreography can be extended by its relationships with other activities in a federated choreography and have a complex implementation described in an orchestration. Thus, choreographies and orchestrations can be viewed on different levels of detail and in context of different relationships.

Choreographies and orchestrations can be composed out of activities by means of typical workflow control structures like parallel or conditional complex activities. A complex activity is used to abstract and hide process strcutures. A choreography is realized by several , at least two, orchestrations owned by involved partners and a single orchestration realizes parts of one or more choreographies. Choreographies can be federated into more complex ones. An activity is represented in a given workflow definition by an activity step. Control steps represent control structures like parallel or conditional split/join.

A control flow of a workflow model can graphically be represented as a directed graph with two kinds of nodes corresponding to activity steps and control steps. Edges represent the transitions between steps. In the rest of this work Workflow-Nets (WF-nets) [10] as the modeling language for both

choreographies and orchestrations are used. For a discussion on how a graph representation can be mapped to WF-net refer to [6].

The WF-Net of the *Purchase processing* choreography in Fig. 1 is depicted in Fig. 3. The Seller upon receiving a request quote from the Buyer answers if the requested item is deliverable. If the Buyer's request is approved by the Seller, 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.

This choreography involves interactions between the Buyer and the Seller, the Seller and the Shipper and the Buyer and the Shipper. Each of them can be described by a separate choreography. Existing proposals allow to combine several choreographies only by means of composition [4, 11] 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 complex than that. In our example the interaction between the Seller and the Shipper includes 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 is described by a separate choreography between Seller, Shipper and Bank. This choreography, illustrated in Fig. 4, has additional activities and partners which are not visible to the Buyer. This choreography supports the *Purchase processing* choreography and partially elaborates the interaction between the Seller and Shipper. The advantage of our approach is that each partner only knows those parts of the interaction which address this partner, hence avoiding unnecessary information. Besides partners can keep their interaction with other organizations private, invisible to other non-involved partners, which improves protection of business secrecy.

3 Conformance Test

The key requirement of the model is the inter- and intra-layer conformance and consistency. This requires conformance of realizing orchestrations with the choreographies (inter-layer), as well as conformance of supporting and supported choreographies with each other in the choreography layer (intra-layer). Intuitively, the notion of conformance indicates that supporting choreographies and realizing orchestrations must violate none of the requirements of the choreographies they support or realize. Such choreographies and orchestrations are an extended subset of the supported choreography. This means they can not change the order of execution of the activities defined in the supported choreography nor define any alternative for activities as at run time the alternative activity and not the originally defined activity can be

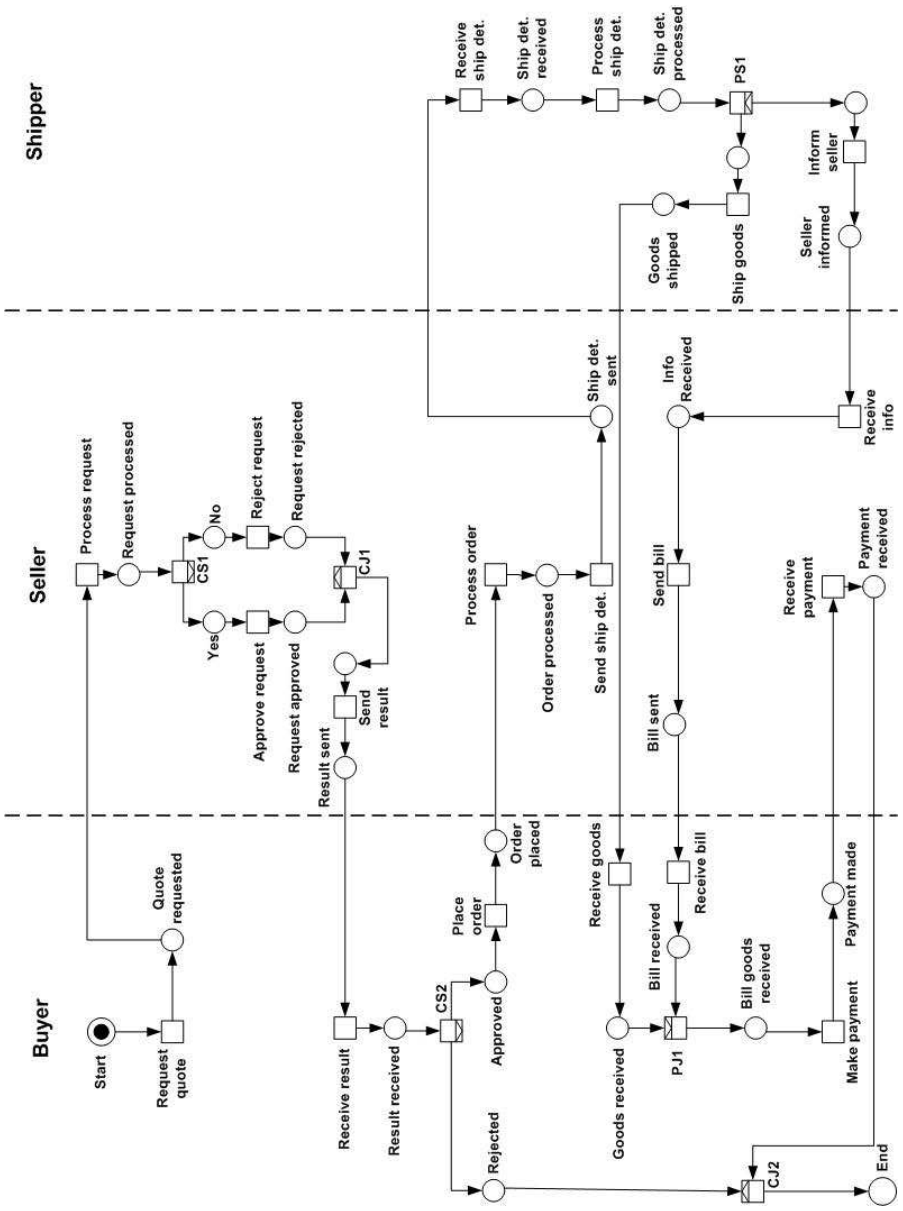

Fig. 3. The WF-Net of *Purchase processing* choreography in Fig. 1

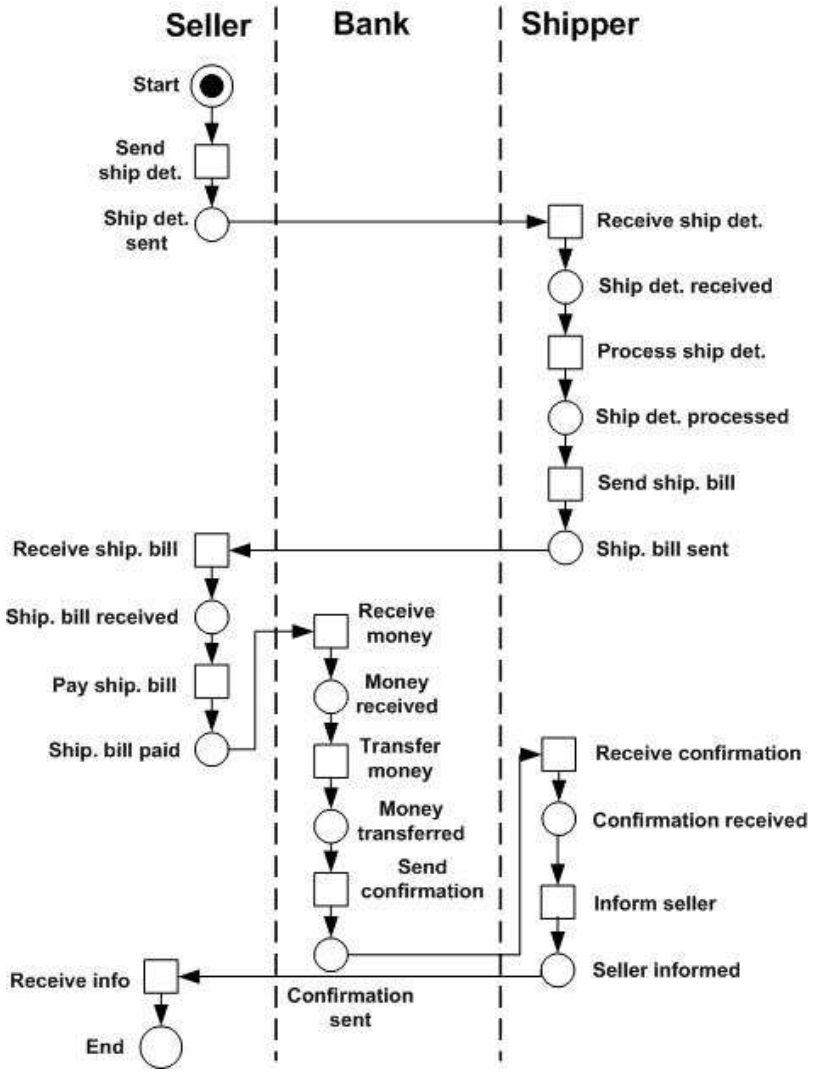

Fig. 4. The *Shipment processing* choreography between Seller, Bank and the Shipper

executed which is an obvious violation of the requirements of the supported choreography.

For example as depicted in Fig. 1 the *Shipment processing* choreography supports the choreography responsible for *Purchase processing*. As illustrated in Fig. 3, the Seller orders the shipment after the Buyer has placed an order. The choreography between Buyer and Seller in the *Shipment processing* choreography (Fig. 4) and their orchestrations must be designed in such a way

that their executions do not lead to skip of the tasks defined in the *Purchase processing* choreography e.g. no shipment order or shipment details are sent to the Shipper after receipt of the order by the Seller. In the following subsection we formalize the notion of conformance using *projection inheritance* [12, 13] based on *branching bisimulation* [14, 15] as equivalence relation.

3.1 Basic Definitions

Petri nets [10, 16, 17] and their subclasses are a convenient modeling language for workflows and workflow based applications and are widely used and vastly studied in the literature. It is supported by many tools and applications as a highly expressive language with a well-defined structure.

Definition 1. *A WF-Net is a Petri Net N with the following additional properties:*

- *N has a special place i whose preset is empty. We call this place* input place.
- *N has a special place o whose postset is empty. We call this place* output place.
- *Any node* $x \in (P \bigcup T)$ *is on a path from input place to output place, i.e. there is no dangling node. Where* P *is the set of places and* T *the set of transitions.*

For comparison of behavior of two WF-Nets, we need an equivalence relation stating if two WF-Nets possess behavioral equivalence i.e. they exhibit the same observable behavior. For this aim we use *branching bisimulation*. The notion of branching bisimulation has been introduced in [14, 15]. Other equivalence relations, e.g. observational equivalence, are also defined. For a detailed discussion on other notions of equivalence please refer to [18, 19, 20, 21, 22]. Branching bisimulation as a generalization of the theory of bisimulation has gained popularity in the computer science community. In process algebra the notion of abstraction [23] provides a mean for making actions unobservable or hiding them. The abstraction operator renames the label of actions to the label τ. A τ-labeled action is called a *silent action* or synonymously *hidden action* or *internal action* in the literature. A silent action is invisible from outside and cannot be recognized since it is hidden from the external observer and has no external effect. All other actions are external actions and visible. Assume A is the set of actions, we define $A_\tau = A \bigcup \{\tau\}$

Before formalizing the branching bisimulation we need one auxiliary definition. The notation p $\xrightarrow{a}$q means that state p evolves to the state q by performing the action a.

Definition 2. *A binary relation* $\Re \subseteq \Pi \times \Pi$ *is a branching bisimulation:*
$\forall p, p\prime, q, q\prime \in \Pi, \alpha \in A_\tau \Leftrightarrow$
$p \Re q \bigwedge p \xrightarrow{\alpha} p\prime \Rightarrow ((\alpha = \tau \bigwedge p\prime \Re q) \bigvee (\exists q\prime, q\prime\prime \in \Pi : q \mapsto q\prime\prime \xrightarrow{\alpha} q\prime \bigwedge p \Re q\prime\prime \bigwedge p\prime \Re q\prime),$
$p \Re q \bigwedge q \xrightarrow{\alpha} q\prime \Rightarrow ((\alpha = \tau \bigwedge p \Re q\prime) \bigvee (\exists p\prime, p\prime\prime \in \Pi : p \mapsto p\prime\prime \xrightarrow{\alpha} p\prime \bigwedge p\prime\prime \Re q \bigwedge p\prime \Re q\prime),$

Two processes p and q are branching bisimilar, denoted $p \approx_b q$, iff there exists a branching bisimulation relationship between p and q.

In [24] has been shown that branching bisimulation is an equivalence relation. We use this equivalence relation to test if the different layers of our federated choreographies comply and fit together.

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. In order to check if two workflows, choreographies or orchestrations, are conformant we use the notion of *projection inheritance*. This concept and other notions of inheritance and the relationship among them and to branching bisimulation have been defined in [12, 13].

Definition 3. Projection inheritance states that *"If it is not possible to distinguish the behaviors of x and y when arbitrary tasks of x are executed, but when only the effects of tasks that are also present in y are considered, then x is a subclass of y."*

Assume W_a and W_b are two choreographies, modeled as WF-Nets, and W_b has a link to W_a i.e. W_b supports or realizes W_a. W_b comprises a set of methods some of which are internal and not included in W_a and W_a includes actions that are not interesting for W_b. These actions can be made invisible using the abstraction operator. In order to decide which actions of W_a shall be made invisible we need to define the Greatest Common Divisor of W_a and W_b.

Definition 4. *The Greatest Common Divisor of two WF-Nets are parts of the nets that two nets have in common, denoted GCD_{W_a,W_b}.*

We define $W\prime_a = W_a - GCD_{W_a,W_b}$, this is the methods of W_a that are not included in GCD_{W_a,W_b}. Let $\tau_1(W\prime_a)$ be the corresponding net after application of abstraction operator on methods of $W\prime_a$. W_b is conformant with W_a, if and only if its visible behavior has a branching bisimulation relation to the $\tau_1(W\prime_a)$. Groote and Vaandrager in [25] have introduced an algorithm by which in polynomial time is decidable if two processes are branching bisimilar. This algorithm has time complexity $O(n.(n+m))$ and space complexity $O(n+m)$, where n is the number of states and m the number of transitions.

Conformance Algorithm

Let W_a, W_b be two WF-Nets such that there is a link between W_a, W_b and W_b supports or realizes W_a. In order to decide if federated choreographies are

conformant it must be checked that all choreographies and orchestrations that are linked to another choreography are conformant.

1: For all links in the model s.t. W_b supports or realizes W_a {
2: compute GCD_{W_a,W_b} and subsequently $W\prime_a$
3: compute $\tau_1(W\prime_a)$, i.e. apply abstraction operator to all methods of $W\prime_a$ of W_a
4: compute if W_b is a subclass of $\tau_1(W\prime_a)$ under projection inheritance. If such a relationship does not exist the federated choreographies are not conformant and the algorithm terminates otherwise it is conformant.
5: endfor }

The algorithm iteratively takes two supporting and supported choreographies (Intra-layer conformance) or realizing orchestration and realized choreography (inter-layer conformance) and computes their GCD by comparing the name keys of the activities. It then computes the relevant parts of the supported or realized choreography by computing which of its activities are not contained in the GCD and applying the abstraction operator on them. Finally the algorithm decides the subclass relationship between the relevant parts of the supported or realized choreography with the supporting choreography or realizing orchestration. There are some tools that based on the algorithm in [25] by an enumerative approach can decide if a WF-Net is a subclass of another WF-net under projection inheritance among other inheritance definitions. For example Woflan [26] can be used for deciding the subclass relationship between two WF-Nets. Fig. 3 depicts the *Purchase processing* choreography and Fig. 4 the *Shipment processing* choreography. As illustrated in Fig. 1 the *Shipment processing* choreography supports the *Purchase processing* choreography. In order to check if *Shipment processing* choreography is conformant with the *Purchase processing* choreography the relevant parts of the *Purchase processing* choreography (the supported choreography) as described in the algorithm shall be extracted and be checked if the *Shipment processing* choreography (the supporting choreography) is its subclass under projection inheritance. Fig. 5 illustrates the relevant parts of the *Purchase processing* choreography for the *Shipment processing* choreography. The inter-layer conformance of the model, e.g. between the *Purchase processing* Choreography and the Seller's orchestration can be done in the same manner. For this the relevant parts of the *Purchase processing* choreography this time for the Seller's orchestration must be computed and checked if the subclass relationship under projection inheritance exists. Note that this algorithm checks the structural conformance of the model and other consistency issues e.g. data flow consistency and messaging consistency are out of scope of this work.

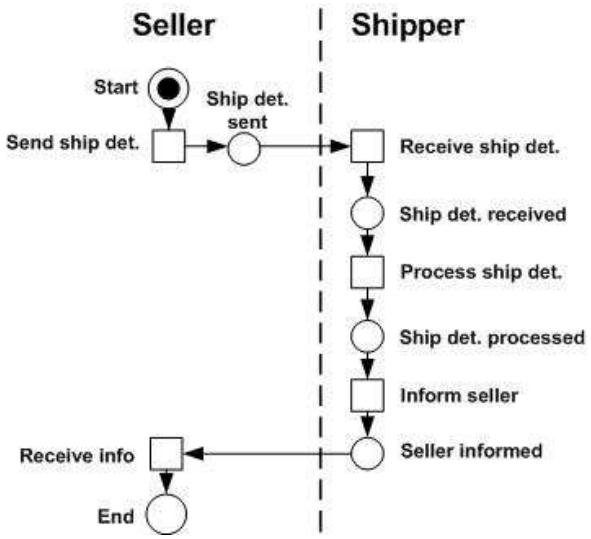

Fig. 5. Relevant parts of the *Purchase processing* choreography for the *Shipment processing* choreography

4 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 metamodel 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. The modular view is expected to care for the subtle visibility and information and process hiding vs. exposing needs characteristic for interoperability between enterprizes. The modeling concepts introduced are intended for an improved consideration of these complex distribution of information. The distributed (maybe even scattered) choreography models are difficult to comprehend. Therefore, the automatic control of conformance between the modules should be a major enabler for practically using the proposed federated architecture.

References

1. Barros AP, Dumas M, Oaks P (2005) Standards for web service choreography and orchestration: Status and perspectives. Proc. of the 1st International Workshop on

Web Service Choreography and Orchestration for Business Process Management, Nancy, France
2. Peltz C (2003) Web services orchestration and choreography. IEEE Computer. 36(10):46-52
3. Andrews T, et al (2003) Business process execution language for web services (bpel4ws). ver. 1.1, BEA, IBM, Microsoft, SAP, Siebel Systems
4. Kavantzas N, Burdett D, Ritzinger G (2004) Web services choreography description language (ws-cdl) 1.0. W3C Working Draft
5. Arkin A, et al (2002) Web Service Choreography Interface (wsci) 1.0. W3C
6. Eder J, Lehmann M, Tahamtan A (2006) Choreographies as Federations of Choreographies and Orchestrations. Proc. of CoSS, Tucson, AZ, USA
7. Dijkman RM, Dumas M (2004) Service-oriented design: A multi-viewpoint approach. Int. J. Cooperative Inf. Syst. 13(4):337-368
8. Eder J, Gruber W (2002) A meta model for structured workflows supporting workflow transformations. Proc. of ADBIS, Bratislava, Slovakia
9. Lehmann M (2006) Data Access in Workflow Management Systems. Ph.D. Thesis, University of Klagenfurt, Austria
10. Van der Aalst WMP (1998) The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers. 8(1):21-66
11. Burdett D, Kavantzas N (2004) WS choreography model overview. W3C
12. Van der Aalst WMP, Basten T (2002) Inheritance of workflows: an approach to tackling problems related to change. Theor. Comput. Sci. 270(1-2):125-203
13. Basten T (1998) In Terms of Nets: System Design with Petri Nets and Process Algebra. Ph.D thesis, TU Eindhoven, The Netherlands
14. Van Glabbeek RJ, Weijland WP (1998) Branching time and abstraction in bisimulation semantics (extended abstract). Proc. of IFIP 11th World Comput. Congr. San Francisco, CA, USA
15. Van Glabbeek RJ, Weijland WP (1996) Branching time and abstraction in bisimulation semantics. J. ACM. 43(3):555-600
16. Murata T (1989) Petri nets: Properties, analysis and applications. Proc. of IEEE. 77(4):541-580
17. Peterson JL (1977) Petri nets. ACM Comput. Surv. 9(3):223-252
18. Graf S (1986) A complete inference system for an algebra or regular acceptance models. Proc. of MFCS, Bratislava, Czechoslovakia
19. Hennessy M (1985) Acceptance trees. J. ACM. 32(4):896-928
20. Kanellakis PC, Smolka SA (1983) Ccs expressions, finite state processes, and three problems of equivalence. Inf. Comput. 86(1):43-86
21. Milner R (1982) A Calculus of Communicating Systems. Springer, New York
22. Olderog ER (1985) Specification-oriented programming in TCSP. In: Krzysztof R (ed) Logics and Models of Concurrent Systems, NATO ASI Series F, volume 13, Springer, New York
23. Bergstra JA, Klop JW (1985) Algebra of communicating processes with abstraction. Comput. Sci. 37:77-121
24. Basten T (1996) Branching bisimilarity is an equivalence indeed! Inf. Process. Lett. 58(3):141-147
25. Groote JF, Vaandrager FW (1990) An efficient algorithm for branching bisimulation and stuttering equivalence. proc. of ICALP, Warwick, England
26. Verbeek HMW, Van der Aalst WMP (2000) Woflan 2.0: A petri-net-based workflow diagnosis tool. Proc. of ICATPN, Aarhus, Denmark