The P2P Approach to Interorganizational Workflows

Wil M.P. van der Aalst and Mathias Weske

Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{w.m.p.v.d.aalst, m.weske}@tm.tue.nl

Abstract. This paper describes in an informal way the Public-To-Private (P2P) approach to interorganizational workflows, which is based on a notion of inheritance. The approach consists of three steps: (1) create a common understanding of the interorganizational workflow by specifying a shared public workflow, (2) partition the public workflow over the organizations involved, and (3) for each organization, create a private workflow which is a subclass of the respective part of the public workflow. Using an example, we explain that the P2P approach yields an interorganizational workflow which is guaranteed to realize the behavior specified in the public workflow.

1 Introduction

In today's corporations, products and services are typically created by business processes, and workflow technology can be used for enhancing the flexibility and efficiency of these processes [14,19]. Corporations often operate across organizational boundaries, for example in E-commerce and extended enterprises [11,20,27]. Consequently, workflows between organizations – interorganizational workflows – are becoming increasingly important [21,12]. Interorganizational workflows are typically subject to conflicting constraints of the organizations involved. On the one hand, there is a strong need for coordination to optimize the flow of work in and between organizations. On the other hand, the organizations involved are essentially autonomous and have the freedom to create or modify workflows at any point in time. Some of the issues resulting from these conflicting goals will be tackled in this paper: We introduce the Public-To-Private (P2P) approach to interorganizational workflows which provides the means to specify a common public workflow, to partition it according to the organizations involved and to allow for private refinement of the parts by the organizations, based on a notion of inheritance. The P2P approach guarantees that the private workflows of the participating organizations (or, as we prefer to say, the domains) satisfy the public workflow as agreed upon; it consists of the following steps:

– *Step 1:* The organizations involved agree on a common public workflow, which serves as a contract between these organizations.

K.R. Dittrich, A. Geppert, M.C. Norrie (Eds.): CAiSE 2001, LNCS 2068, pp. 140–156, 2001.

- *Step 2:* Each task of the public workflow is mapped onto one of the domains. Each domain is responsible for a part of the public workflow, referred to as its public part.
- *Step 3:* Each domain can now make use of its autonomy to create a private workflow. To satisfy the correctness of the overall interorganizational workflow, however, each domain may only choose a private workflow which is a subclass of its public part.

This paper introduces the P2P approach in an informal way, guided by an example of an electronic bookstore. The paper is structured according to the steps mentioned, and for each step concepts and notations are introduced when required; the complete definitions and the technical details of the proofs can be found in [4]. Sections 2 through 4 present the phases of the P2P approach, and Section 5 summarizes the main results. A discussion of related work and concluding remarks complete this paper.

2 Designing the Public Workflow (Step 1)

The example used throughout this paper is inspired by electronic bookstores such as Amazon [8] and Barnes and Noble [9]. In this section, we design the public workflow for ordering books. The scope of the workflow process includes the ordering, billing and shipping of books, involving the customer, the bookstore, the publisher, and the shipper.

The P2P approach uses *workflow nets* (WF-nets) [2] for modeling workflows, which are a specific form of Petri nets. In WF-nets, tasks are modeled by transitions, and causal dependencies are modeled by places and arcs. In fact, a place corresponds to a condition which can be used as pre- and/or post-condition for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs. A WF-net has one source place and one sink place because any case (i.e., workflow instance) represented by the WF-net is created when it enters the workflow management system and is deleted once it is completely handled. An additional requirement is that there should be no dangling tasks or conditions, i.e., tasks and conditions which do not contribute to the processing of cases. Therefore, all the nodes of the workflow should be on some path from source to sink. WF-nets with these properties are called *sound* [1,2].

Figure 1 shows the public workflow N^{publ} of the electronic bookstore. This workflow can be regarded as a contract between the domains, i.e., the customer, the bookstore, the publisher, and the shipper. We stress that the public workflow does not necessarily show the way the tasks are actually executed; the real process may be much more detailed, and it may involve much more tasks. The public workflow only contains the tasks which are of interest to all parties. The public workflow shown in Fig. 1 is defined as a WF-net. While the mapping of the tasks to domains is only done in the next step, one can think of the tasks in the left column as performed by the customer, for instance the *place_c_order* task.

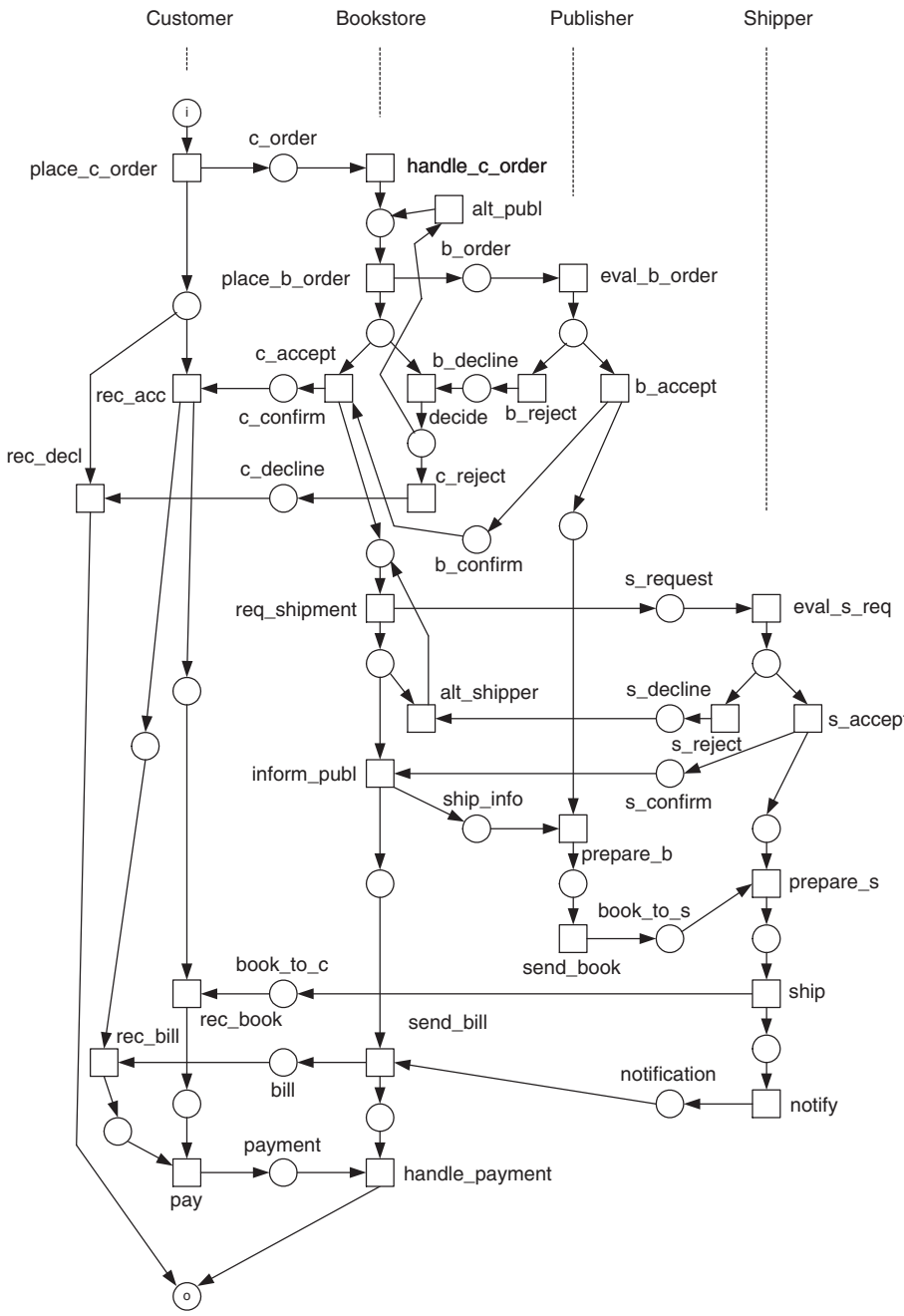

Fig. 1. The public workflow N^{publ}.

The next columns to the right belong to the bookstore (containing, e.g., the *handle_c_order* task to handle the customer order), the publisher (e.g., *eval_b_order*), and the shipper (e.g., *eval_s_req*), respectively.

The workflow process is initiated by a customer placing an order (represented by the task *place_c_order*). This customer order is sent to and is handled by the bookstore (*handle_c_order*). The electronic bookstore is a virtual company which has no books in stock. Therefore, the bookstore transfers the order of the desired book to a publisher (*place_b_order*). The bookstore order is evaluated by the publisher (*eval_b_order*) and either accepted (*b_accept*) or rejected (*b_reject*). In both cases an appropriate signal is sent to the bookstore. If the bookstore receives a negative answer, it decides (*decide*) to either search for an alternative publisher (*alt_publ*) or to reject the customer order (*c_reject*). If the bookstore searches for an alternative publisher, a new bookstore order is sent to another publisher, etc. If the customer receives a negative answer (*rec_decl*), then the workflow terminates. If the bookstore receives a positive answer (*c_accept*), the customer is informed (*rec_acc*), and the bookstore continues processing the customer order.

Once the order is confirmed, the bookstore sends a request to a shipper (*req_shipment*), the shipper evaluates the request (*eval_s_req*) and either accepts (*s_accept*) or rejects (*b_reject*) the shipping request. If the bookstore receives a negative answer, it searches for another shipper. This process is repeated until a shipper accepts. Note that, unlike the unavailability of the book, the unavailability of a shipper can not lead to a cancellation of the order. After a shipper is found, the publisher is informed (*inform_publ*), the publisher prepares the book for shipment (*prepare_b*), and the book is sent from the publisher to the shipper (*send_book*). The shipper prepares the shipment to the customer (*prepare_s*) and actually ships the book to the customer (*ship*). The customer receives the book (*rec_book*) and the shipper notifies the bookstore (*notify*). The bookstore sends the bill to the customer (*send_bill*). After receiving both the book and the bill (*rec_bill*), the customer makes a payment (*pay*). Then the bookstore processes the payment (*handle_payment*) and the interorganizational workflow terminates.

The public workflow shown in Fig. 1 is indeed a sound WF-net, since it has exactly one input place and one output place, at the moment when the workflow reaches the output place, all tasks have completed, and there are no dead transitions, i.e., all tasks of the WF-net are in fact reachable during workflow executions.

3 Partitioning the Public Workflow (Step 2)

In the second step of the P2P approach, the public workflow is partitioned according to the domains, and the public parts are related to each other, making up an interorganizational workflow. An interorganizational workflows is defined by an *interorganizational workflow net* (IOWF-net). An IOWF-net consists of a set of WF-nets, a set of channels, a set of methods, and a channel flow relation.

In our example, the public workflow is partitioned over four domains: the customer domain, the bookstore domain, the publisher domain, and the shipper

domain, as shown in Fig. 2. Methods of the domains are represented by shaded boxes, and they are linked to channels by the channel flow relation, which is represented by arrows. In Fig. 2, the public parts of the customer, the bookstore, the publisher and the shipper are represented by boxes N_C^{part}, N_B^{part}, N_P^{part}, and N_S^{part}, respectively. Channels are represented by icons, and the channel flow relation represented by arrows specifies the linkage of the domains. For example, the *c_order* channel and the attached arrows represent the fact that customer order information flows from the customer domain to the bookstore domain, while the confirmation of the order flows in opposite direction, making use of channel *c_confirm*.

Based on this description it is clear how the public workflow needs to be partitioned. The public part of the customer domain is quite simple (cf. Fig. 3): The customer first places an order, using the method *place_c_order*. Then either the order is accepted, the book and the bill are received and the bill is paid, or the order is declined. Notice that for each transition in the WF-net, there is a method linked to it by a dotted line, representing the actual function which is invoked when the task is executed.

The public part of the bookstore workflow is slightly more complex (cf. Fig. 4): After the order arrives, the bookstore checks for a publisher ready for providing the ordered book. If no publisher can be found, the order is rejected. Otherwise, shipment is requested from a shipper, and payment is handled. The public parts of the publisher and shipper workflow are shown in Fig. 5.

The IOWF-net is a high-level representation of the domains and their dependencies; its semantics are given in terms of a labeled P/T net. A IOWF-net is transformed into a labeled P/T net by taking the union of all WF-nets, adding a place for each channel, connecting transitions to these newly added places, and removing superfluous source and sink places. We call this the flattening of the interorganizational workflow. As shown in [4], we can easily make sure that the partitioning is valid, i.e., all public parts are sound WF-nets and there is no multiple activation. We mention that the flattened IOWF-net equals the public workflow. Hence, flattening the interorganizational workflow shown in Fig. 2 results in the public workflow shown in Fig. 1.

4 Designing the Private Workflows (Step 3)

After partitioning the public workflow, each domain can realize the corresponding public part of the interorganizational workflow in any way they want, as long as they make sure that their private workflow is a subclass of their public part.

The subclass relationship between WF-nets is based on a specific notion of inheritance, called *projection inheritance*. Projection inheritance has been defined in [6,10] and uses encapsulation as a mechanism to establish subclass-superclass relationships. The basic idea of projection inheritance can be characterized as follows:

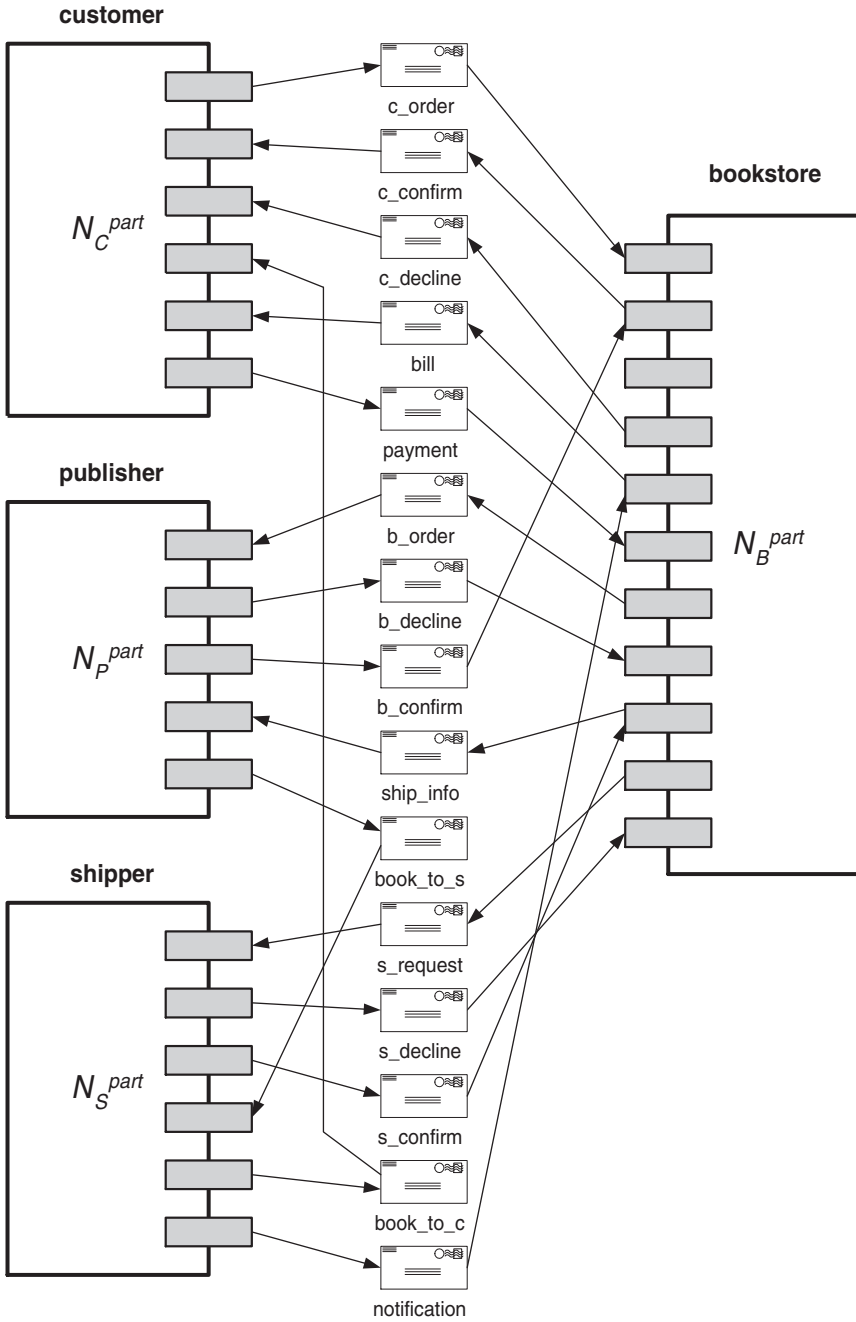

Fig. 2. The interorganizational workflow Q^{part}.

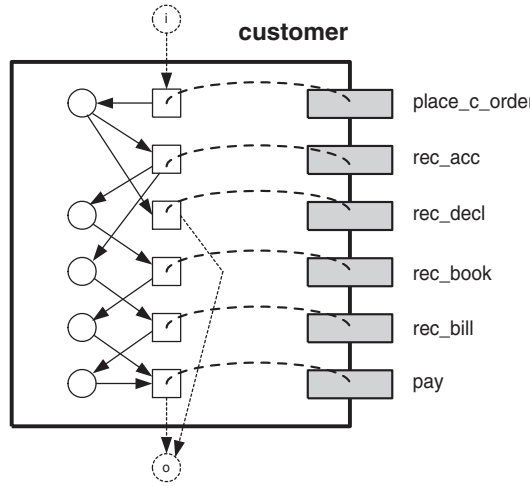

Fig. 3. The WF-net N_C^{part} (public part of the customer domain).

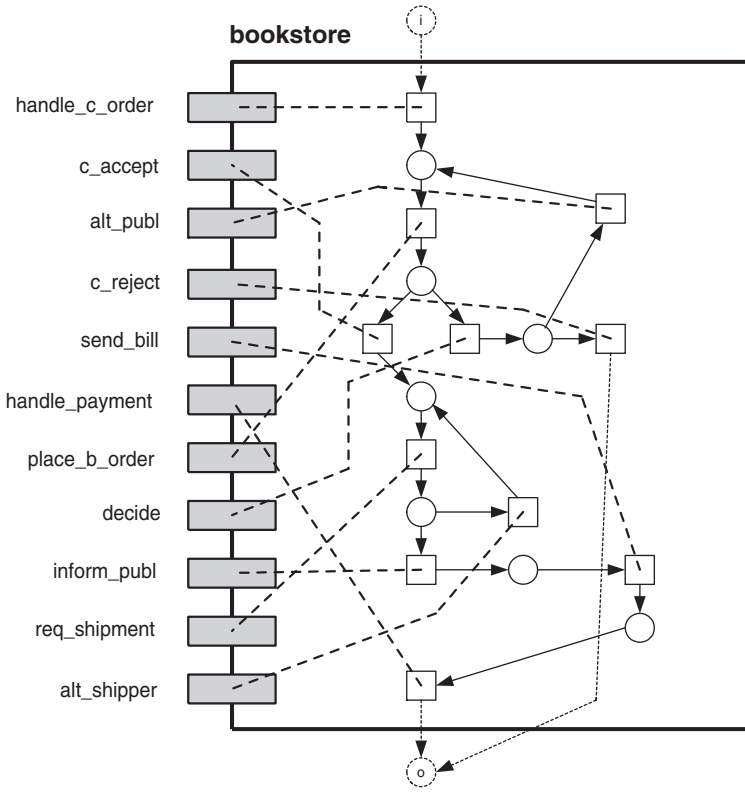

Fig. 4. The WF-net N_B^{part} (public part of the bookstore domain).

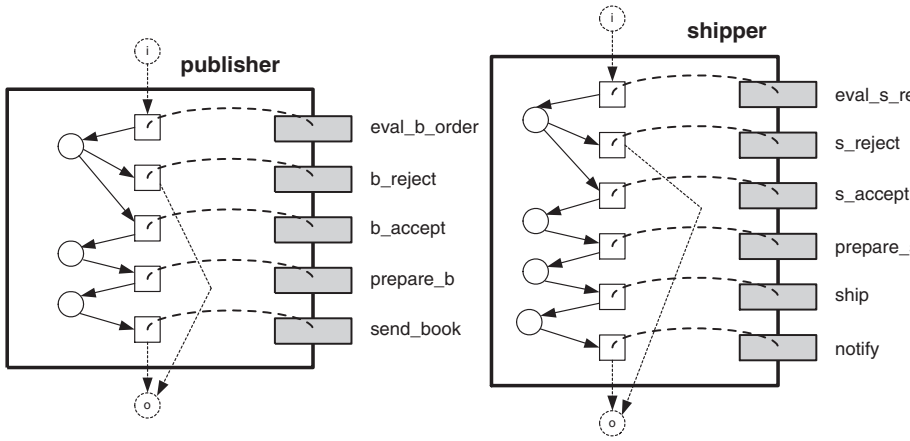

Fig. 5. The WF-net N_P^{part} (public parts of the publisher and shipper domains).

If it is not possible to distinguish the behaviors of x and y when arbitrary methods of x are executed, but when only the effects of methods that are also present in y are considered, then x is a subclass of y.

Projection inheritance is based on branching bisimilarity as the standard equivalence relation on marked, labeled P/T-nets [15]. For projection inheritance, all new methods (i.e., methods added in the subclass) are hidden; an abstraction operator τ is used to hide methods.

For any two sound WF-nets N and N', N' is a subclass of N under projection inheritance if and only if the externally visible behavior of N' is branching bisimilar to N. Let us consider the five WF-nets shown in Fig. 6. N_1 is not a subclass of N_0 because hiding of the new task d results in a potential execution where a is followed by c without executing b, i.e., the WF-net where d is hidden is not branching bisimilar. N_2 is a subclass of N_0 because hiding e in N_2 results in a behavior equivalent to the behavior of N_0, i.e., the addition of e only postpones the execution of b and does not allow for a bypass such as the one in N_1. N_3 is also a subclass of N_0: Hiding the parallel branch containing f yields the original behavior. Finally, N_4 is also a subclass of N_0.

Based on the notion of projection inheritance we have defined three *inheritance-preserving transformation rules*. These rules correspond to design patterns when extending a superclass to incorporate new behavior: (1) adding a loop (rule *PPS*), (2) inserting methods in-between existing methods (rule *PJS*), and (3) putting new methods in parallel with existing methods (rule *PJ3S*). The formal definitions of these transformation rules, their preconditions, and the proofs that these rules actually preserve projection inheritance are given in [6,10].

In the P2P approach, projection inheritance is used as a formal link between the public parts of the domains and the private workflows which are actually executed. Transformation rules are the key mechanism to create specializations of a given WF-net, making use of the fact that applying these rules to a given

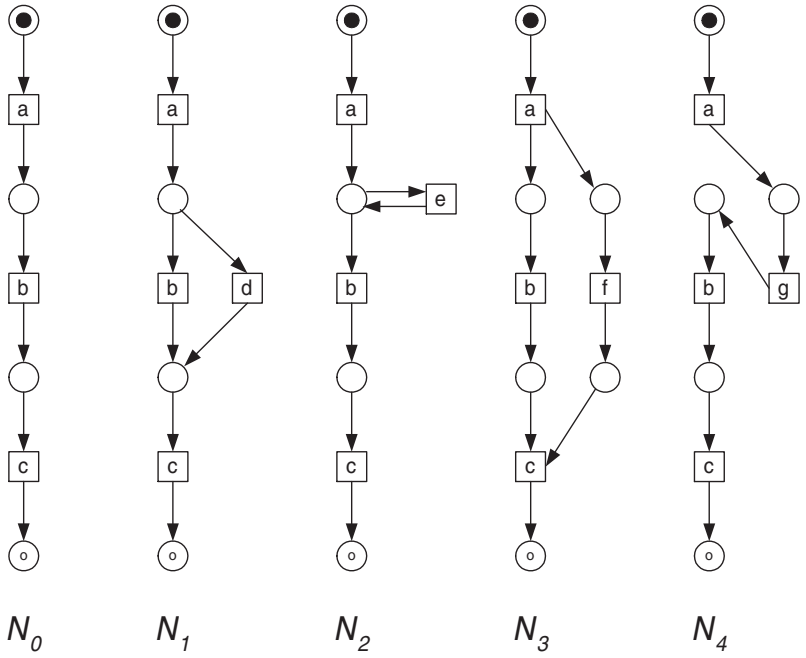

Fig. 6. N_2, N_3, and N_4 are subclasses of N_0 under projection inheritance.

WF-net is guaranteed to create a subclass of that WF-net. Hence, the P2P approach is constructive in the sense that any modification applied to a WF-net via transformation rules *PPS*, *PJS*, and *PJ3S* yields a subclass of the WF-net.

Figure 7 shows the private workflow of the bookstore. Five new tasks, i.e., tasks not present in the public workflow, have been added. After the customer order is handled, the customer profile (information about the interests of the customer) is updated (*update_customer_profile*). This task is executed in parallel with the placement of the bookstore order. After both tasks have been executed, the marketing department is informed (*inform_marketing*). The tasks *monitor_order*, *monitor_shipment*, and *monitor_payment* have been added to monitor the behavior of the publisher, shipper, and customer. The task *monitor_order* can be executed as long as the bookstore is waiting for a response of the publisher. The task *monitor_shipment* can be executed between the moment the publisher is informed and the moment the shipper sends a notification. The task *monitor_payment* can be executed after the bill is sent to the customer. Note that each of the monitor tasks can be executed multiple times. For example, the bookstore checks every week whether the customer has paid and if needed takes action, e.g., sending a bailiff.

We now show by construction that the private workflow N_B^{priv} (Fig. 7) is indeed a subclass of N_B^{part} (Fig. 4): tasks *monitor_order*, *monitor_shipment*, and *monitor_payment* can be added by applying transformation rule *PPS* three times;

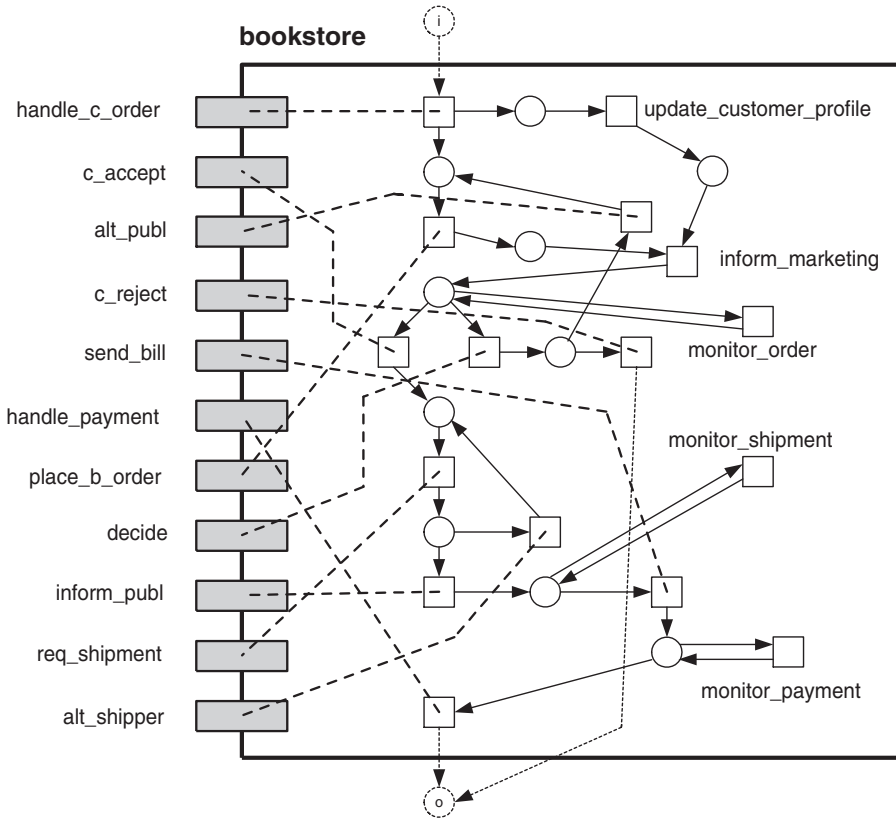

Fig. 7. The WF-net N_B^{priv} (private workflow of the bookstore domain).

task *inform_marketing* can be added using transformation rule *PJS*. To complete the construction, the task *update_customer_profile* can be added using transformation rule *PJ3S*.

Similarly, the private workflow of the publisher (Fig. 8) has been created by applying transformation rule *PJS* to the public part: The task *check_warehouse* has been added in-between the receipt of the order and the decision. In fact, the decision is based on the result of *check_warehouse*. After accepting the order of the bookstore, the corresponding inventory item is locked (*lock_inventory*), the stock is replenished (if possible) (*replenish*), and the book is moved to the part of the warehouse reserved for books which are waiting for shipment (*move_book_to_release_buffer*). It is easy to verify that the private workflow N_P^{priv} is a subclass of N_P^{part}, using the transformation rule *PJS*.

Figure 9 shows the private workflow of the shipper. Using the transformation rules, six new tasks have been added: Task *check_availability_trucks* is executed after the request by the bookstore is received. Based on this task the request is accepted or rejected. Tasks *update_file* and *quality_control* are executed in

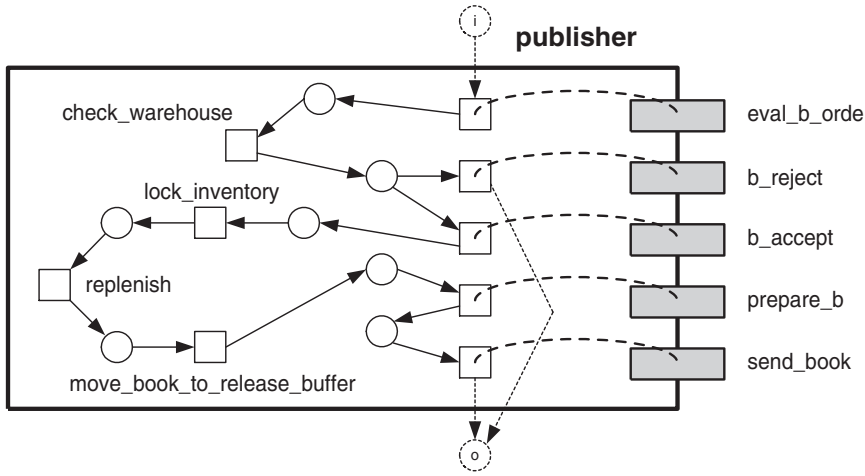

Fig. 8. The WF-net N_P^{priv} (private workflow of the publisher domain).

parallel with the preparation and shipment tasks. After preparation, shipments are assigned to trucks (*assignment*). Based on the assignment, the routing of the truck is determined (*routing*). In-between tasks *assignment* and *routing* the task *re-assignment* can be executed multiple times. Again it is easy to verify that the WF-net shown in Fig. 9 is indeed a subclass of the one shown in Fig. 5. Task *check_availability_trucks* can be added using transformation rule *PJS*. Tasks *update_file* and *quality_control* can be added using transformation rule *PJ3S*. Tasks *assignment*, *re-assignment*, and *routing* can be added using transformation rule *PJS*. Note that it is also possible to first add tasks *assignment* and *routing* using *PJS*, and then add task *re-assignment* using transformation rule *PPS*.

The design of the interorganizational workflow involving a customer, bookstore, publisher, and shipper presented in this paper is a simplification of the real process. In the real process customers can order multiple books at the same time, the customer can return books, the customer can refuse to pay, etc. One can imagine that for realistic interorganizational workflows where the public part consists of more than fifty tasks and the overall workflow consists of hundreds of tasks, a structured approach is needed to avoid all kinds of anomalies. In our opinion, the P2P approach could be used as a starting point for a more comprehensive approach which also deals with other aspects such as data and security.

5 Summary and Main Results

To summarize the P2P approach, in the first step the public workflow is specified in terms of a sound WF-net; it serves as a contract between the business partners involved. In the second step, the public workflow is partitioned over the set of domains. Note that each domain corresponds to an organizational entity. As a

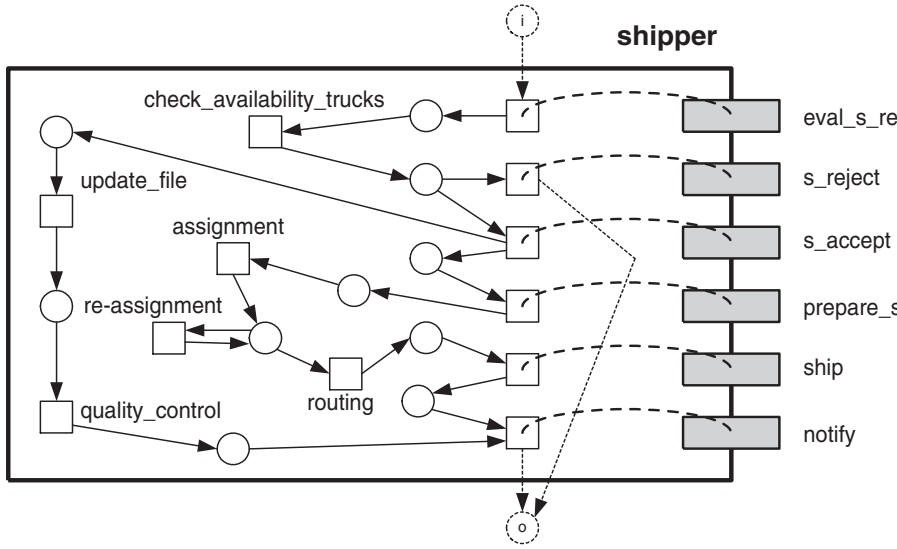

Fig. 9. The WF-net N_S^{priv} (private workflow of the shipper domain).

result of the partitioning, each fragment of the partitioned workflow corresponds to one of the domains and is represented by a sound WF-net, called public part. In the final step, the public parts are replaced by private workflows. Each private workflow corresponds to an actual workflow as it is executed in one of the domains. The P2P approach guarantees that each private workflow is a subclass of the corresponding public part under projection inheritance. It is important to note that the P2P approach is constructive: By applying the three transformation rules introduced above, the design is guaranteed to be correct without the need to check whether each private workflow is actually a subclass of the corresponding public part.

Following the general tone of this paper, we explain the main results informally and introduce concepts if and when required. Please refer to [4] for a detailed theoretical discussion. The first result concerns the *overall workflow*, which consists of all private workflows of the participating domains.

Result 1: The overall workflow is a sound WF-net.

This property is based on the observation that a part of a WF-net (called subflow) can be replaced by a specialization (i.e., a subclass subflow) without endangering soundness of the overall workflow. This result is proven in [4], based on a theorem which shows the compositionality of projection inheritance. From an application point of view, Result 1 makes sure that the P2P approach guarantees that the overall workflow is free of deadlocks and other anomalies.

Result 2: The overall workflow is a subclass of the public workflow.

This result shows that the dynamic behavior of the interorganizational workflow which the business partners agreed upon in the public workflow is in fact guaran-

teed to be satisfied by the execution of the interorganizational workflow, i.e., the overall workflow. From an application point of view, this is an important result, since it provides the business partners with the ability to perform any private modifications to their public workflow part, as long as the subclass relationship holds. Transformation rules are used for this purpose. Hence, an organization can be sure that its private workflow indeed satisfies the requirements specified in the contract, i.e., the public workflow.

The next result is based on the notion of *local views* of the domains. To introduce local views, we mention that each domain is aware of its private workflow and of the public parts of the other domains. The information which each domain has with respect to the overall workflow is called the local view of that domain. With respect to local views, the following interesting result can be obtained, which stresses the soundness of the P2P approach.

> *Result 3:* The overall workflow is a subclass of the local views of all domains, which in turn are subclasses of the public workflow.

For the final two properties we have to introduce some notation. Since projection inheritance is a partial ordering on the set of WF-nets, the Greatest Common Denominator (GCD) and the Least Common Multiple (LCM) can be defined. GCD and LCM are general concepts that apply to any ordering, and there are different applications of these concepts in the context of WF-nets, as described in more detail in [6]. In essence, the GCD of a set of WF-nets is a WF-net that captures the part these nets have in common, i.e., the part where they agree on. The LCM captures all possible behaviors. Note that projection inheritance is a partial order but not a lattice. Therefore, suitable definitions of GCD and LCM are far from trivial but can be defined as is shown in [6].

For an illustration of these concepts, consider the WF-nets N_0, N_2, N_3, and N_4 shown in Fig. 6. The GCD of these four nets is N_0, i.e., each of the four WF-nets is a subclass of this net and it is not possible to find a different WF-net which is also a superclass of N_2, N_3, and N_4 and at the same time a subclass of N_0. Figure 10 shows $N_{GCD} = N_0$ as the GCD of N_0, N_2, N_3, and N_4. Figure 10 also shows the WF-net N_{LCM}. N_{LCM} is a subclass of each of the four nets considered. Moreover, it is not possible to find a different WF-net which is also a subclass of N_0, N_2, N_3, and N_4 and at the same time a superclass of N_{LCM}. Any execution sequence generated by one of the four nets can also be generated by N_{LCM} after the appropriate abstraction. Based on the characterization of GCD and LCM we are now ready to present the following result:

> *Result 4:* The GCD of all local views is the public workflow.

The application specific interpretation of this result is as follows: The public workflow is the superclass of the local views of all domains, and it is minimal in the sense that no different WF-net can be found, which is a superclass of the local views and at the same time a subclass of the public workflow. This is an interesting, yet not surprising result. It shows that the local views of the domains have exactly the public workflow in common.

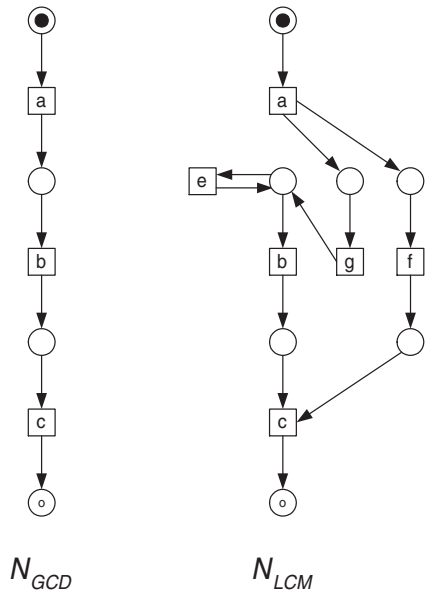

Fig. 10. The greatest common divisor N_{GCD} and least common multiple N_{LCM} of N_0, N_2, N_3, and N_4 shown in Fig. 6.

Analogously to the discussion of Result 4, the final result states a relationship between the local views of the domains and the overall workflow, as it is executed:

Result 5: The LCM of all local views is the overall workflow.

We interpret Result 5 as follows: The overall workflow is a specialization of all local views; conversely, the local views are superclasses of the overall workflow. The overall workflow is minimal in the sense that it is not possible to find a different WF-net which is also a subclass of all local views and which is a superclass of the overall workflow.

6 Related Work and Conclusions

Petri nets have been proposed for modeling workflow process definitions long before the term "workflow management" was coined and workflow management systems became readily available. Consider for example the work on Information Control Nets, a variant of the classical Petri nets, in the late seventies [13].

Only a few papers in the literature focus on the verification of workflow process definitions. In [16] some verification issues have been examined and the complexity of selected correctness issues has been identified, but no concrete verification procedures have been suggested. In [1] and [7] concrete verification procedures based on Petri nets have been proposed. This paper builds upon the work presented in [1] where the concept of a sound WF-net was introduced. The

technique presented in [7] has been developed for checking the consistency of transactional workflows including temporal constraints. However, the technique is restricted to acyclic workflows and only gives necessary conditions (i.e., not sufficient conditions) for consistency. In [23] a reduction technique has been proposed. This reduction technique uses a correctness criterion which corresponds to soundness and the class of workflow processes considered are in essence acyclic free-choice Petri nets.

This paper differs from the above approaches because the focus is on interorganizational workflows. Only a few papers explicitly focus on the problem of verifying the correctness of interorganizational workflows [3,17]. In [3] the interaction between domains is specified in terms of message sequence charts and the actual overall workflow is checked with respect to these message sequence charts. A similar, but more formal and complete, approach is presented by Kindler, Martens, and Reisig in [17]. The authors give local criteria, using the concept of scenarios (similar to runs or basic message sequence charts), to guarantee the absence of certain anomalies at the global level. Both approaches [3,17] are not constructive, i.e., they only specify criteria for various notions of correctness but do not provide concrete design rules such as the transformation rules.

In the last decade several researchers explored notions of behavioral inheritance (also named subtyping or substitutability), see [10] for an overview. Researchers in the domain of formal process models (e.g., Petri-nets and process algebras) have tackled similar questions based on the explicit representation of a process by using various notions of (bi)simulation . The inheritance notion used in this paper is characterized by the fact that it is equipped with both *inheritance-preserving transformation rules* to *construct* subclasses [10] and *transfer rules* to *migrate* instances from a superclass to a subclass and vice versa [6]. These features are very relevant for a both constructive and robust approach towards interorganizational workflows.

We have developed a tool named *Woflan* (WOrkFLow ANalyzer [2,28]). Woflan is an analysis tool which can be used to verify the correctness of a workflow process definition. The analysis tool uses state-of-the-art techniques to find potential errors in the definition of a workflow process. Woflan is designed as a workflow management system independent analysis tool. In principle it can interface with many workflow management systems. At the moment, Woflan can interface with the workflow management systems COSA (Software Ley [25]), METEOR (LSDIS [24]), Staffware (Staffware [26]), and with the business process re-engineering tool Protos (Pallas Athena [22]). Woflan has not been designed to analyze interorganizational workflows. However, it can be used to verify the soundness property used throughout this paper, and it can also check whether a given workflow is a subclass of another workflow.

In the future we hope to extend the P2P approach in several directions. First of all, we want to address local dynamic changes. The transfer rules presented in [6] can be used to migrate workflow instances from a superclass to a subclass and vice versa. Therefore, it is possible to change the workflows in each of the domains

on the fly, i.e., it is possible to automatically transfer each case to the latest version of the process. Other aspects of future work include the reconfiguration of interorganizational workflows (tasks move from one domain to another), the usage of alternative inheritance notions and the implementation of the concepts in prototypical workflow management systems, e.g., by using METEOR [5,24] or InterProcs [18].

References

1. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, Berlin, 1997.
2. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W.M.P. van der Aalst. Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
4. W.M.P. van der Aalst. Inheritance of Interorganizational Workflows: How to Agree to Disagree Without Loosing Control? BETA Working Paper Series, WP 46, Eindhoven University of Technology, Eindhoven, 2000.
5. W.M.P. van der Aalst and K. Anyanwu. Inheritance of Interorganizational Workflows to Enable Business-to-Business E-commerce. In *Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC'99)*, pages 141–157, Nashville, Tennessee, October 1999.
6. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An approach to tackling problems related to change. *Theoretical Computer Science*, 2001 (to appear).
7. N.R. Adam, V. Atluri, and W. Huang. Modeling and Analysis of Workflows using Petri Nets. *Journal of Intelligent Information Systems*, 10(2):131–158, 1998.
8. Amazon.com, Inc. Amazon.com. http://www.amazon.com, 1999.
9. Barnes and Noble. bn.com. http://www.bn.com, 1999.
10. T. Basten. *In Terms of Nets: System Design with Petri Nets and Process Algebra*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, December 1998.
11. R. Benjamin and R. Wigand. Electronic markets and virtual value chains on the information superhighway. *Sloan Management Review*, pages 62–72, 1995.
12. R.W.H. Bons, R.M. Lee, and R.W. Wagenaar. Designing trustworthy interorganizational trade procedures for open electronic commerce. *International Journal of Electronic Commerce*, 2(3):61–83, 1998.
13. C.A. Ellis. Information Control Nets: A Mathematical Model of Office Information Flow. In *Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems*, pages 225–240, Boulder, Colorado, 1979. ACM Press.
14. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
15. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
16. A.H.M. ter Hofstede, M.E. Orlowska, and J. Rajapakse. Verification Problems in Conceptual Workflow Specifications. *Data and Knowledge Engineering*, 24(3):239–256, 1998.

17. E. Kindler, A. Martens, and W. Reisig. Inter-Operability of Workflow Applications: Local Criteria for Global Soundness. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 235–253. Springer-Verlag, Berlin, 2000.
18. R.M. Lee. Distributed Electronic Trade Scenarios: Representation, Design, Prototyping. *International Journal of Electronic Commerce*, 3(2):105–120, 1999.
19. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
20. T.W. Malone, R.I. Benjamin, and J. Yates. Electronic Markets and Electronic Hierarchies: Effects of Information Technology on Market Structure and Corporate Strategies . *Communications of the ACM*, 30(6):484–497, 1987.
21. M. Merz, B. Liberman, K. Muller-Jones, and W. Lamersdorf. Interorganisational Workflow Management with Mobile Agents in COSM. In *Proceedings of PAAM96 Conference on the Practical Application of Agents and Multiagent Systems*, 1996.
22. Pallas Athena. *Protos User Manual*. Pallas Athena BV, Plasmolen, The Netherlands, 1999.
23. W. Sadiq and M.E. Orlowska. Applying Graph Reduction Techniques for Identifying Structural Conflicts in Process Models. In *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, Berlin, 1999.
24. A. Sheth, K. Kochut, and J. Miller. Large Scale Distributed Information Systems (LSDIS) laboratory, METEOR project page. http://lsdis.cs.uga.edu/proj/meteor/meteor.html.
25. Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, Germany, 1998.
26. Staffware. *Staffware 2000 / GWD User Manual*. Staffware plc, Berkshire, United Kingdom, 1999.
27. The White House. A Framework for Global Electronic Commerce. http://www.ecommerce.gov/framewrk.htm, 1997.
28. H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. Computing Science Report 99/02, Eindhoven University of Technology, Eindhoven, 1999.