Inheritance of Interorganizational Workflows: How to agree to disagree without loosing control?

W.M.P. van der Aalst[1,2]

[1] Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands. w.m.p.v.d.aalst@tm.tue.nl
[2] Department of Computer Science, University of Colorado at Boulder, Campus Box 430, Boulder, CO 80309-0430, USA

Abstract. Internet-based technology, E-commerce, and the rise of networked virtual enterprises have fueled the need for interorganizational workflows. Although XML allows trading partners to exchange information, it cannot be used to coordinate activities in different organizational entities. Business-to-business processes are hindered by the lack of a common language to support collaboration. This paper describes the P2P (Public-To-Private) approach which addresses some of the problems using a notion of inheritance. The approach consists of three steps: (1) create a common understanding of the interorganizational workflow by specifying the shared public workflow, (2) partition the public workflow over the organizational entities involved, and (3) for each organizational entity: create a private workflow which is a subclass of the relevant part of the public workflow. This paper shows that this approach avoids typical anomalies in business-to-business collaboration (e.g., deadlocks and livelocks) and yields an interorganizational workflow which is guaranteed to realize the behavior specified in the public workflow.

Keywords: Information Systems (H), Electronic Commerce (K.4.4), Workflow Management (H.4.1), Petri Nets (D.2.2), Inheritance of Dynamic Behavior, Verification.

1 Introduction

Today's corporations often must operate across organizational boundaries. Phenomena such as E-commerce, extended enterprises, and the Internet stimulate cooperation between organizations. Therefore, the importance of workflows distributed over a number of organizations is increasing. Interorganizational workflow offers companies the opportunity to re-shape business processes beyond the boundaries of their own organizations. However, interorganizational workflows are typically subject to conflicting constraints. On the one hand, there is a strong need for coordination to optimize the flow of work in and between the different 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. As the subtitle of this paper suggests ("How to agree to disagree without loosing control?"), this is exactly the problem that will be tackled in this paper. To motivate the approach described in this paper, we first discuss some of the developments in the field of E-commerce and Internet-based technologies.

E-commerce refers to the enabling of purchasing and selling of goods and services through a communications network [6, 16, 26, 34, 35, 47, 51]. The ability to conduct business activities involved in marketing, finance, manufacturing, selling, and negotiation, electronically, is what E-commerce is all about. One major objective of adopting E-commerce strategies is to reduce costs and improve the efficiency of business processes by replacing paper business with electronic alternatives. E-commerce, in its earliest incarnation known as Electronic Data Interchange (EDI), has been traditionally used by larger corporations to share and exchange information between business partners and suppliers using private networks [19, 31, 32]. EDI enables the exchange of business data from one computer to another computer. It eliminates the need to re-key information from documents or messages by supporting the creation of electronic versions of documents or messages using public standard formats, which can then be transmitted, received, and interpreted by other systems. Typical applications were (and still are) supply-chain management processes like order placement and processing. However, with the explosive growth of the Internet in the last couple of years, E-commerce is now able to offer solutions for a much broader range of business processes than EDI previously addressed. Also, the extensive availability of the Internet has enabled smaller companies, hindered previously by the large financial investment required for these private networks, to conduct business electronically. Technologies like bar coding, automatic teller machines, e-mail, fax, video-conferencing, workflow, and the World-Wide-Web have continued to impact the success of E-commerce. Although the term E-commerce frequently refers to on-line retailing involving businesses and consumers, experts predict that as E-commerce continues to grow, business-to-business E-commerce will continue to enjoy the lion share of the revenue.

The *Internet* and the *World-Wide-Web* (WWW) have become the de facto standard for E-commerce. The Internet has evolved from a primitive medium to exchange data to the backbone of today's information society. In [30], Kumar and Zhao identify five stages in the development of the WWW. In the first phase, primitive text-based tools such as Gopher and Archie are used primarily for knowledge discovery. In the second phase, hypertext-based graphical browsers are used for knowledge discovery. The third phase is marked by connecting applications and databases to the WWW using gateways based on technologies such as CGI. As a result, the WWW can be used to process transactions in a synchronous manner and present up-to-date information. The fourth phase is the phase where asynchronous mode interaction between series of trading partners is enabled using semantic languages such as XML, ontologies, etc. In the fifth phase procedural information is attached to the information exchanged, i.e., the workflow processes are made explicit (in a common language) and WWW-based applications become "process-aware". Today, we are in-between phase three and phase four. Kumar and Zhao [30] envision that the emphasis will shift from short-lived transactions of a synchronous nature to long-lived transactions which require a complex asynchronous exchange of information. Clearly, the main focus of WWW-based tools and the associated research has been on information, communication, and presentation [44]. As a result, problems related to collaboration, coordination, and business process support have been neglected. Satisfactory concepts and products to support interorganizational workflows are still missing. Business-to-business E-commerce will be hindered by these

problems. Therefore, we focus on some of the problems related to interorganizational workflow.

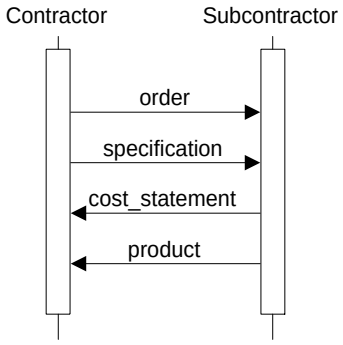

Fig. 1. The interactions between contractor and subcontractor.

To introduce the problems tackled in this paper we use a small example of an interorganizational workflow involving two business partners: a contractor and a subcontractor. The interaction between these two business partners is shown in Figure 1 using a so-called sequence diagram. First, the contractor sends an order to the subcontractor. Then, the contractor sends a detailed specification to the subcontractor and the subcontractor sends a cost statement to the contractor. Based on the specification the subcontractor manufactures the desired product and sends it to the contractor. For this very simple business-to-business protocol a sequence diagram is suitable. However, sequence diagrams are typically used to describe scenarios rather than a complete specification of a business-to-business protocol or trade procedure. Sequence diagram have problems expressing a mixture of choice and synchronization and subtle aspects such as the moment of choice. In Section 2.3, we will address these subtle, but crucial, issues using the workflow model shown in Figure 12. Given these limitations, we use Petri nets [39, 40] to model such business-to-business protocols.

Figure 2 specifies the process described earlier in terms of a Petri net. The *transitions*, represented by squares, correspond to *tasks* and the *places*, represented by circles, correspond to the causal relations between the tasks. The places *order*, *specification*, *cost_statement*, and *product* are used to exchange the messages shown in Figure 1. Places may contain *tokens* and, at any time, the distribution of tokens over places specifies the current state of the process. Initially, place i (i.e., the source place) contains one token corresponding to a new *case* also called *workflow instance*. Transitions are *enabled* if each input place contains a token, i.e., initially transition *send_order* is enabled. Enabled transitions can *fire* by removing a token from each input place and producing a token for each output place, i.e., firing *send_order* results in the consumption of the token in i and the production of three new tokens. After firing *send_order*, transition *receive_order* is the only transition enabled. It is easy to see that starting with a token in place i all transitions are executed (i.e., fired) in a predefined order thus resulting in the state with just one token in the sink place o. Moreover, the exchange of tokens

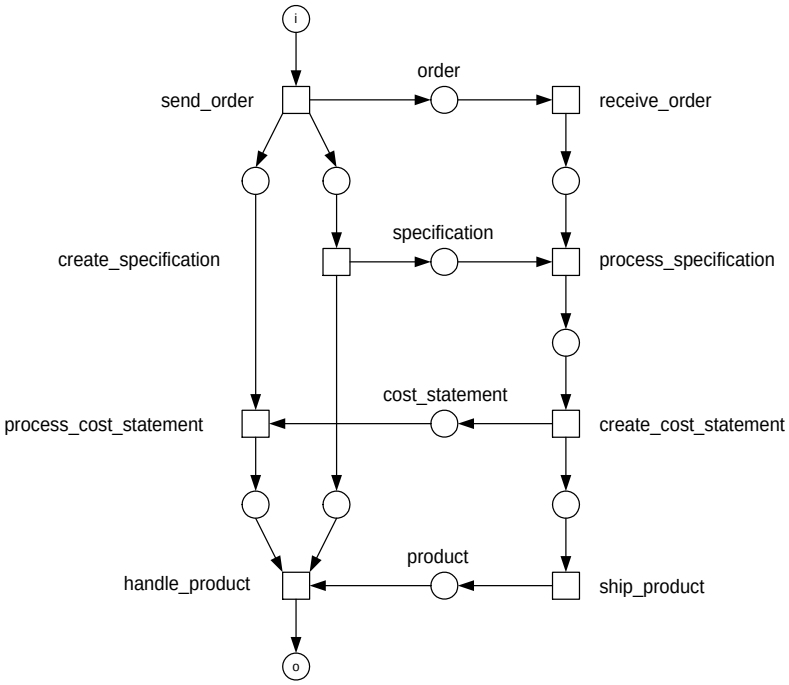

Fig. 2. The public workflow N^{publ}.

via the places *order*, *specification*, *cost_statement*, and *product* matches the interaction pattern shown in Figure 1. Since this Petri net exhibits no choices and the degree of parallelism is limited, the model may seem unnecessary complex. However, for more complex business-to-business protocols, the more advanced constructs offered by the Petri-net formalism are indispensable. Moreover, in contrast to Figure 1, Figure 2 also shows the tasks.

We will use the term *public workflow* for the Petri net shown in Figure 2. One can think of this Petri net as the *contract* between the contractor and the subcontractor, i.e., Figure 2 does not necessarily show the way the tasks are actually executed. The real process may be much more detailed and involving much more tasks. The public workflow only contains the tasks which are of interest to both parties.

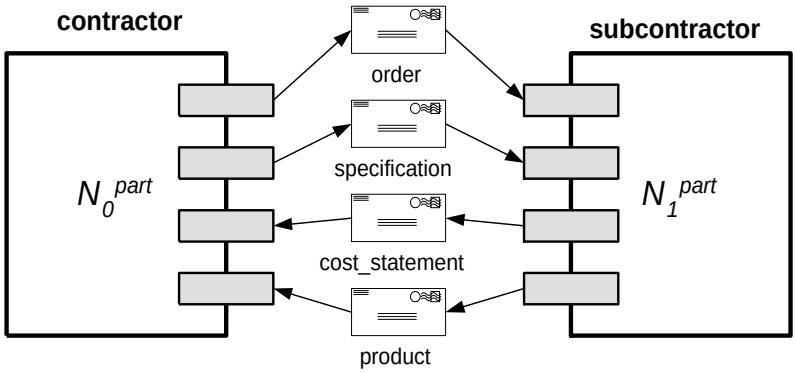

Fig. 3. The interorganizational workflow Q^{part}.

Figure 2 does not show who is executing the tasks. Therefore, we extend the Petri net with a notion of hierarchy as shown in figures 3, 4, and 5. Figure 3 shows the top-level of the interorganizational workflow, i.e., the two business partners involved and the messages exchanged. The two large squares in Figure 3 are called *domains*. In this case there are two domains: one for the contractor (left) and one for the subcontractor (right). The two domains are connected via the *channels order*, *specification*, *cost_statement*, and *product*. The shaded rectangles corresponds to *methods*, i.e., services offered by the domains.

In the interorganizational workflow shown in figures 3, 4, and 5, the public workflow is partitioned over the two domains. Figure 4 shows the contractor's share of the public workflow and Figure 5 shows the subcontractor's share of the public workflow. Transitions in the two domains are mapped onto methods, i.e., the execution of a transition provides the corresponding service offered by the domain in Figure 3. In this particular example there is a one-to-one correspondence between transitions and methods. However, in general several transitions may offer the same service (i.e., are mapped onto the same method). Moreover, there may be transitions which are just added for routing purposes and do not correspond to relevant tasks. These tasks are not mapped onto methods.

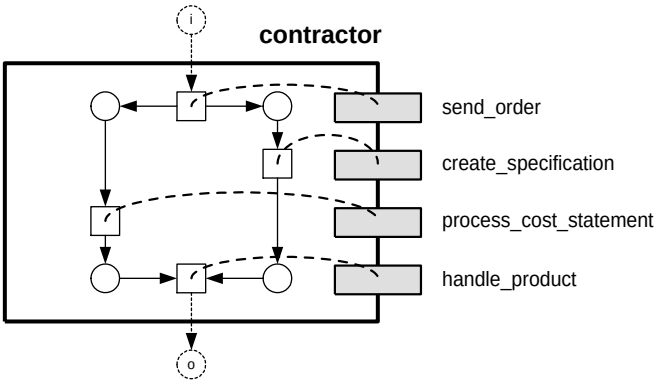

Fig. 4. The WF-net N_0^{part} (public part of contractor).

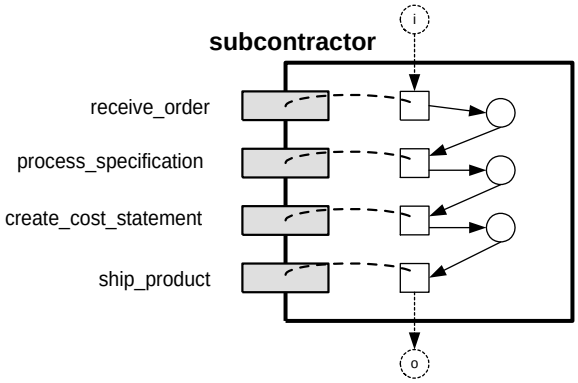

Fig. 5. The WF-net N_1^{part} (public part of subcontractor).

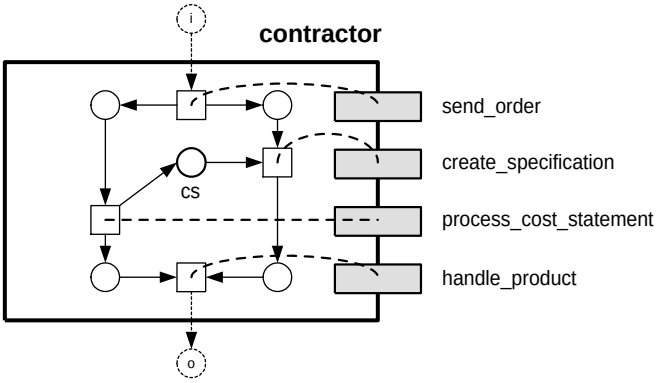

Fig. 6. The WF-net N_0^{priv} (private part of contractor).

The interorganizational workflow corresponding to the partitioned public workflow (i.e., figures 3, 4, and 5) serves only as an *agreement*, i.e., it is the business-to-business protocol the business partners agreed upon and not the real workflow as it is executed. The workflow description which is used to actually execute the workflow within one of the domains is called the *private workflow*. The private workflow typically contains several tasks which are only of local interest. Figure 6 shows a rather a-typical private workflow. This is the private workflow of the contractor and contains no additional tasks. The only thing that has been added is the place *cs* connecting the task which processes the cost statement to the task creating the specification. This place may have been added because the contractor thinks that it is more efficient to create the specification after the cost statement has been processed. From a local point of view, such a change is quite acceptable. If the contractor is only interested in the part of the public workflow shown in Figure 4, the change may seem harmless. However, if the subcontractor executes its local workflow as specified in Figure 5, then the process (i.e., the overall interorganizational workflow) will deadlock after the execution of *send_order* and *receive_order*. This example shows that local changes may have dramatic effects.

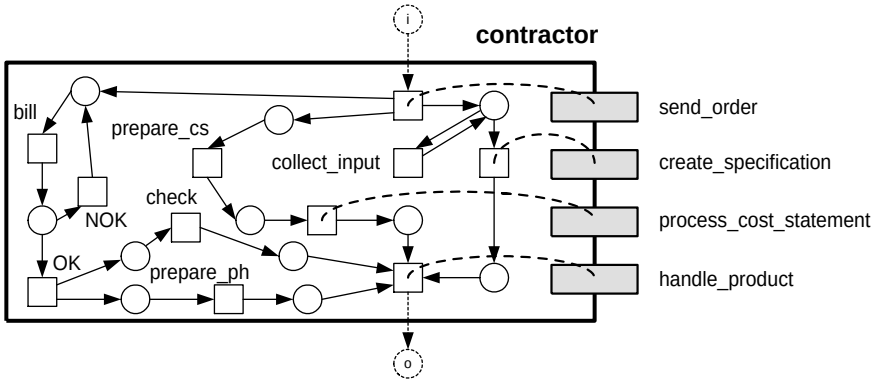

Fig. 7. An alternative WF-net N_0^{priv} (private part of contractor).

Figure 7 shows an alternative private workflow. In this workflow many tasks have been added which are only of local interest, e.g., in-between the sending of the order and the creation of the specification task *collect_input* may be executed multiple times. Although the private workflow shown in Figure 7 adds many tasks to its part of the public workflow, the original order of the key tasks is not changed. In contrast with Figure 6, Figure 7 is consistent with Figure 4. If the subcontractor executes its local workflow as specified in Figure 4 and the contractor executes its local workflow as specified in Figure 7, then the overall interorganizational workflow will run smoothly without deadlocks or similar anomalies.

Figure 8 shows the private workflow of the subcontractor. This workflow contains three tasks not present in the Figure 5: *decide*, *procedure_1*, and *procedure_2*. After the order is received, a decision is made. Based on this decision one of two possible procedures is executed. In one procedure, the specification is processed before the

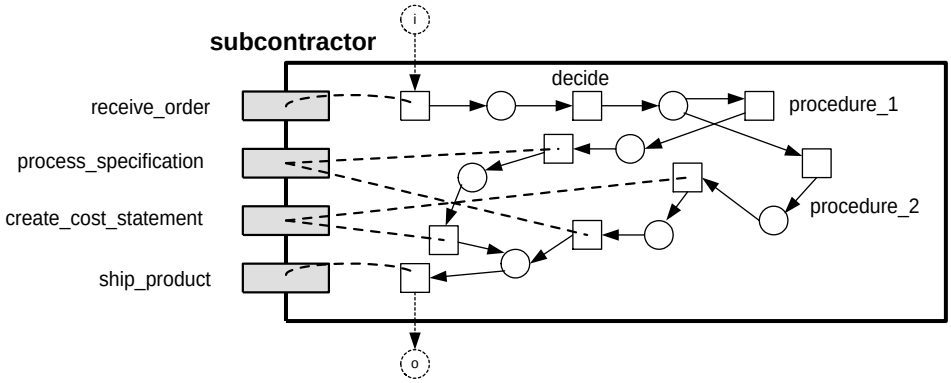

Fig. 8. The WF-net N_1^{priv} (private part of subcontractor).

cost statement is created. In the other procedure, the cost statement is created before the specification is processed. Again, from a local perspective, there is no apparent reason why the private workflow cannot be extended in this way. The first procedure corresponds to the order specified in the public workflow. The second procedure only offers the opportunity to reverse the order of the tasks *process_specification* and *create_cost_statement*. If the private workflow of the subcontractor shown in Figure 8 is combined with the private workflow of the contractor shown in Figure 6, then there is still a potential deadlock. However, executing the alternative procedure can actually help to avoid the deadlock mentioned earlier. If the subcontractor uses the second procedure, the addition of the place cs in the contractor's workflow does not result in a deadlock. This illustrates that the contractor can detect the presence of the alternative procedure and indicates that the private workflow shown in Figure 8 is not consistent with Figure 5. If the private workflow of the subcontractor shown in Figure 8 is combined with the private workflow of the contractor shown in Figure 7, then there are no potential anomalies such deadlocks and livelocks. Nevertheless, something essential has changed. Based on the public workflow, the contractor may assume that the specification has been processed by the subcontractor when the cost statement is processed. However, if the workflow shown in Figure 8 is used, this is no longer guaranteed. Therefore, we consider the workflow shown in Figure 8 not a suitable candidate to realize the subcontractor's share of the public workflow.

Figure 9 shows an alternative private workflow. This workflow is consistent with the subcontractor's part of the public workflow. If the private workflows shown in figures 6 and 9 are combined, the key tasks are executed in the order specified in the public workflow.

This paper addresses the problems illustrated by this small example: How to make sure that the local implementation of a workflow does not create all kinds of anomalies over organizational borders? To solve these problems we propose the *P2P* (Public-To-Private) approach which is based on *projection inheritance*. Projection inheritance has been defined in [7, 8, 15] and uses encapsulation as a mechanism to establish subclass-superclass relationships. In contrast to many other notions of inheritance, it primarily

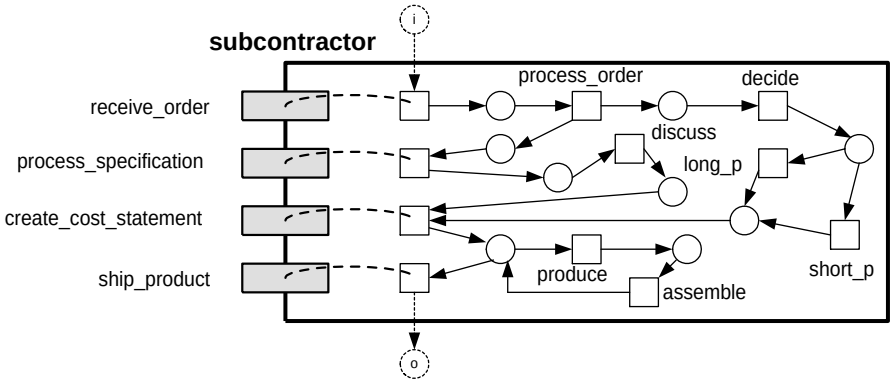

Fig. 9. An alternative WF-net N_1^{priv} (private part of subcontractor).

addresses the dynamic behavior rather than data types or method signatures. The P2P approach consists of three steps. In the first step, the public workflow is created. In the second step, the public workflow is partitioned over a number of domains. Finally, a private workflow is created for each domain such that the private workflow is a subclass of the corresponding part of the public workflow. If the P2P approach is followed, it is guaranteed that the overall workflow (i.e., the workflow obtained by combining all private workflows) is free of deadlocks and other similar anomalies. Moreover, the overall workflow is a subclass of the public workflow. Therefore, it is guaranteed that the business-to-business protocol specified in the public workflow is actually realized. To demonstrate the validity of our approach we show that the Greatest Common Denominator (GCD) of all local views (i.e., the part of the process visible from one domain) is the public workflow. We also show that the Least Common Multiple (LCM) of all local views is the overall workflow.

The remainder of this paper is organized as follows. First, we introduce the notations, techniques, and theoretical results used in this paper. Unfortunately the proofs are quite complex and require a lot of preliminaries. The paper builds on Petri nets [39, 40], sound WF-nets [3], branching bisimilarity [24], projection inheritance [7, 15], and GCD/LCM of processes [8]. The corresponding concepts are all introduced in Section 2. Readers familiar with these concepts can pass over selected parts of this section. Section 3 introduces the framework used to model interorganizational workflows. The P2P approach is described in Section 4. Section 5 demonstrates that the overall workflow realizes the public workflow if the P2P approach is used. Section 6 introduces the notion of views and shows that the GCD and LCM of these views coincide with the public respectively overall workflow. To illustrate the relevance of the results we use the P2P approach to design an interorganizational workflow for a fictive electronic bookstore similar to amazon.com or bn.com in Section 7. Finally, we conclude the paper by relating the results to existing work, describing our tool Woflan which partly supports the P2P approach, and summarizing our plans for future work.

2 Preliminaries

2.1 Place/Transition nets

In this section, we define a variant of the classic Petri-net model, namely labeled Place/-Transition nets. For a more elaborate introduction to Petri nets, the reader is referred to [21, 37, 39]. Let U be some universe of identifiers; let L be some set of *action labels*. $L_v = L \backslash \{\tau\}$ is the set of all visible labels. (The role of τ, the silent action, will be explained later.)

Definition 1 (Labeled P/T-net). *A labeled Place/Transition net is a tuple* (P, T, M, F, ℓ) *where:*

1. $P \subseteq U$ *is a finite set of* places,
2. $T \subseteq U$ *is a finite set of* transitions *such that* $P \cap T = \emptyset$,
3. $M \subseteq L_v$ *is a finite set of* methods *such that* $M \cap (P \cup T) = \emptyset$,
4. $F \subseteq (P \times T) \cup (T \times P)$ *is a set of directed arcs, called the* flow relation, *and*
5. $\ell : T \rightarrow M \cup \{\tau\}$ *is a* labeling function.

Each transition has a label which refers to the *method* or *operation* that is executed if the transition fires. However, if the transition bears a τ label, then no method is executed. Note that there can be many transitions with the same label, i.e., executing the same method. Figure 2 shows a P/T net with 13 places and 8 transitions. The figure only shows the labels of the transitions, i.e., the identifiers of the transitions and some of the places are not shown. In this particular example where there are no two transitions with the same label. Therefore, we can use the transition labels to identify transitions.

Let (P, T, M, F, ℓ) be a labeled P/T-net. Elements of $P \cup T$ are referred to as *nodes*. A node $x \in P \cup T$ is called an *input node* of another node $y \in P \cup T$ if and only if there exists a directed arc from x to y; that is, if and only if xFy. Node x is called an *output node* of y if and only if there exists a directed arc from y to x. If x is a place in P, it is called an input place or an output place; if it is a transition, it is called an input or an output transition. The set of all input nodes of some node x is called the *preset* of x; its set of output nodes is called the *postset*, e.g., the preset of the transition labeled *receive_order* in Figure 2 is the singleton containing *order* and the preset of the transition labeled *handle_product* contains three places. Two auxiliary functions $\bullet\_, \_\bullet : (P \cup T) \rightarrow \mathcal{P}(P \cup T)$ are defined that assign to each node its preset and postset, respectively. For any node $x \in P \cup T$, $\bullet x = \{y \mid yFx\}$ and $x\bullet = \{y \mid xFy\}$. Note that the preset and postset functions depend on the context, i.e., the P/T-net the function applies to. If a node is used in several nets, it is not always clear to which P/T-net the preset/postset functions refer. Therefore, we augment the preset and postset notation with the name of the net whenever confusion is possible: ${}^{N}\bullet x$ is the preset of node x in net N and $x{}^{N}\bullet$ is the postset of node x in net N.

Definition 2 (Marked, labeled P/T-net). *A* marked, *labeled P/T-net is a pair* (N, s), *where* $N = (P, T, M, F, \ell)$ *is a labeled P/T-net and where* s *is a bag over* P *denoting the marking (also called state) of the net. The set of all marked, labeled P/T-nets is denoted* $\mathcal{N}$.

For some bag X over alphabet A and $a \in A$, $X(a)$ denotes the number of occurrences of a in X, often called the cardinality of a in X. The set of all bags over A is denoted $\mathcal{B}(A)$. The empty bag, which is the function yielding 0 for any element in A, is denoted $\mathbf{0}$. For the explicit enumeration of a bag we use square brackets and superscripts to denote the cardinality of the elements. For example, $[a^2, b, c^3]$ denotes the bag with two elements a, one b, and three elements c. In this paper, we allow the use of sets as bags.

Definition 3 (Transition enabling). *Let (N, s) be a marked, labeled P/T-net in $\mathcal{N}$, where $N = (P, T, M, F, \ell)$. A transition $t \in T$ is* enabled*, denoted $(N, s)[t\rangle$, if and only if each of its input places p contains a token. That is, $(N, s)[t\rangle \Leftrightarrow \bullet t \leq s$.*

If a transition t is enabled in marking s (notation: $(N, s)[t\rangle$), then t can fire. If, in addition, t has label a (i.e., $a = \ell(t)$ is the associated method, operation, or observable action) and firing t results is marking s', then $(N, s)\ [a\rangle\ (N, s')$ is used to denote the potential firing.

Definition 4 (Firing rule). *The firing rule $\_[\_\rangle\_ \subseteq \mathcal{N} \times L \times \mathcal{N}$ is the smallest relation satisfying for any (N, s) in $\mathcal{N}$, with $N = (P, T, M, F, \ell)$, and any $t \in T$,*
$(N, s)[t\rangle \Rightarrow (N, s)\ [\ell(t)\rangle\ (N, s - \bullet t + t\bullet)$.

Consider the labeled P/T-net shown in Figure 2. In the marking with just a token in i (i.e., marking $[i]$) only transition *send_order* is enabled. Firing this transition in marking $[i]$ results in a state where the three output places of *send_order* are marked, i.e., the transition consumes one token and produces three tokens. Note that if transition *handle_product* is fired, three tokens are consumed and only one token is produced.

Definition 5 (Firing sequence). *Let (N, s_0) with $N = (P, T, M, F, \ell)$ be a marked, labeled P/T-net in $\mathcal{N}$. A sequence $\sigma \in T^*$ is called a* firing sequence *of (N, s_0) if and only if $\sigma = \varepsilon$ or, for some positive natural number $n \in \mathbb{N}$, there exist markings $s_1, \ldots, s_n \in \mathcal{B}(P)$ and transitions $t_1, \ldots, t_n \in T$ such that $\sigma = t_1 \ldots t_n$ and, for all i with $0 \leq i < n$, $(N, s_i)[t_{i+1}\rangle$ and $s_{i+1} = s_i - \bullet t_{i+1} + t_{i+1}\bullet$. Sequence σ is said to be* enabled *in marking s_0, denoted $(N, s_0)[\sigma\rangle$. Firing the sequence σ results in the unique marking s, denoted $(N, s_0)\ [\sigma\rangle\ (N, s)$, where $s = s_0$ if $\sigma = \varepsilon$ and $s = s_n$ otherwise.*

In Figure 2 the following firing sequence is enabled in marking $[i]$: *send_order, receive_order, create_specification, process_specification, create_cost_statement, process_cost_statement, ship_product, handle_product*. Execution of this firing sequence starting in $[i]$ yields marking $[o]$, i.e., the state with just one token in the sink place.

Definition 6 (Reachable markings). *The set of* reachable markings *of a marked, labeled P/T-net $(N, s) \in \mathcal{N}$ with $N = (P, T, M, F, \ell)$, denoted $[N, s\rangle$, is defined as the set $\{s' \in \mathcal{B}(P) \mid (\exists\, \sigma : \sigma \in T^* : (N, s)\ [\sigma\rangle\ (N, s'))\}$.*

In Figure 2, eleven markings are reachable from $[i]$.

Definition 7 (Connectedness). *A labeled P/T-net $N = (P, T, M, F, \ell)$ is* weakly connected*, or simply* connected*, if and only if, for every two nodes x and y in $P \cup T$, $x(F \cup F^{-1})^* y$. Net N is* strongly connected *if and only if, for every two nodes x and y in $P \cup T$, xF^*y.*

The P/T-net shown in Figure 2 is connected but not strongly connected, e.g., there is no directed path from place o to place i.

Definition 8 (Directed path). *Let (P, T, M, F, ℓ) be a labeled P/T-net. A (directed) path C from a node n_1 to a node n_k is a sequence $\langle n_1, n_2, \ldots, n_k \rangle$ such that $n_i F n_{i+1}$ for $1 \leq i \leq k-1$. C is elementary if and only if for any two nodes n_i and n_j on C, $i \neq j \Rightarrow n_i \neq n_j$. C is non-trivial iff it contains at least two nodes.*

Since the P/T-net shown in Figure 2 is acyclic all directed paths are elementary.

Definition 9 (Union of labeled P/T-nets). *Let $N_0 = (P_0, T_0, M_0, F_0, \ell_0)$ and $N_1 = (P_1, T_1, M_1, F_1, \ell_1)$ be two labeled P/T-nets such that $(P_0 \cup P_1) \cap (T_0 \cup T_1) = \emptyset$ and such that, for all $t \in T_0 \cap T_1$, $\ell_0(t) = \ell_1(t)$. The union $N_0 \cup N_1$ of N_0 and N_1 is the labeled P/T-net $(P_0 \cup P_1, T_0 \cup T_1, F_0 \cup F_1, \ell_0 \cup \ell_1)$. If two P/T-nets satisfy the abovementioned two conditions, their union is said to be* well defined.

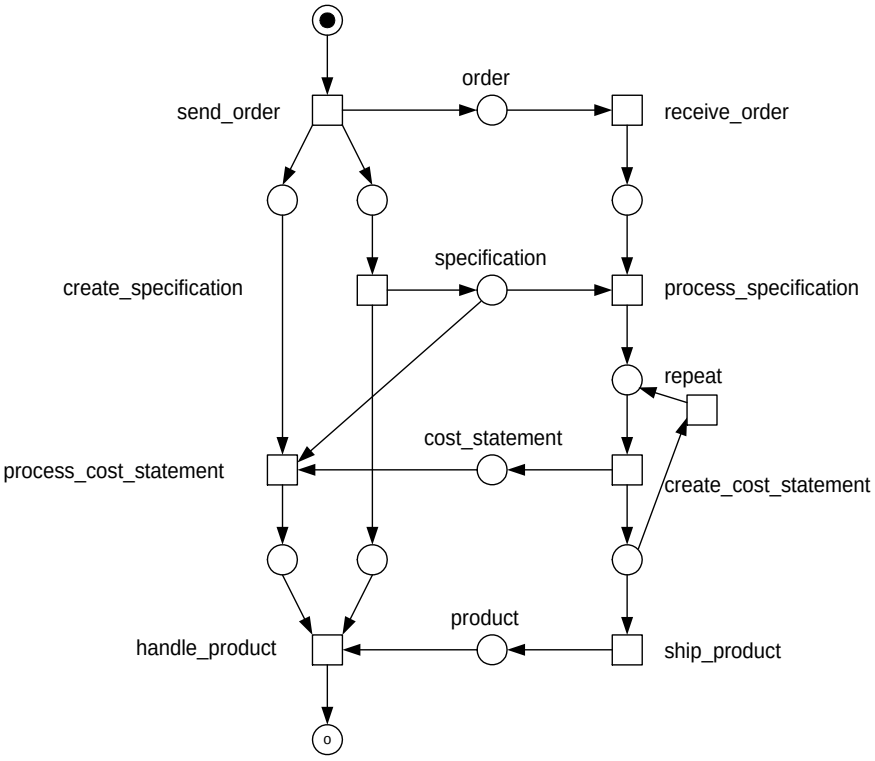

Fig. 10. A non-bounded, non-safe, non-live marked P/T-net with two dead transitions.

Definition 10 (Boundedness). *A marked, labeled P/T-net $(N, s) \in \mathcal{N}$ is* bounded *if and only if the set of reachable markings $[N, s\rangle$ is finite.*

The labeled P/T-net shown in Figure 2 is bounded for any initial marking. The marked P/T-net shown in Figure 10 is not bounded because place *cost_statement* can be marked with any number of tokens by executing the transitions *create_cost_statement* and *repeat* alternatingly.

Definition 11 (Safeness). *A marked, labeled P/T-net* $(N, s) \in \mathcal{N}$ *with* $N = (P, T, M, F, \ell)$ *is* safe *if and only if, for any reachable marking* $s' \in [N, s\rangle$ *and any place* $p \in P$, $s'(p) \leq 1$.

Safeness implies boundedness. Therefore, the unbounded marked P/T-net shown in Figure 10 cannot be safe.

Definition 12 (Dead transition). *Let* (N, s) *be a marked, labeled P/T-net in* $\mathcal{N}$. *A transition* $t \in T$ *is* dead *in* (N, s) *if and only if there is no reachable marking* $s' \in [N, s\rangle$ *such that* $(N, s')[t\rangle$.

The transitions *process_cost_statement* and *handle_product* are dead in the marked P/T-net shown in Figure 10. None of the transitions in Figure 2 is dead given the initial marking $[i]$.

Definition 13 (Liveness). *A marked, labeled P/T-net* $(N, s) \in \mathcal{N}$ *with* $N = (P, T, M, F, \ell)$ *is* live *if and only if, for every reachable marking* $s' \in [N, s\rangle$ *and transition* $t \in T$, *there is a reachable marking* $s'' \in [N, s'\rangle$ *such that* $(N, s'')[t\rangle$.

A marked P/T-net containing dead transitions is not live, e.g., the marked P/T-net shown in Figure 10 cannot be live because it contains two dead transitions. However, there are non-live marked P/T-nets without any dead transitions. Consider for example the P/T-net shown in Figure 2 with initial marking $[i]$ which is not live and has no dead transitions. Note that if we fuse i and o in Figure 2, then the corresponding net is live in $[i]$.

2.2 Workflow nets

For the modeling of workflow processes we use labeled P/T-nets with a specific structure. We will name these nets *workflow nets* (WF-nets).

Definition 14 (WF-net). *Let* $N = (P, T, M, F, \ell)$ *be a labeled P/T-net. Net* N *is a* workflow net *(WF-net) if and only if the following conditions are satisfied:*

1. *instance creation:* P *contains an input (source) place* $i \in U$ *such that* $\bullet i = \emptyset$,
2. *instance completion:* P *contains an output (sink) place* $o \in U$ *such that* $o\bullet = \emptyset$,
3. *connectedness:* $\bar{N} = (P, T \cup \{\bar{t}\}, M, F \cup \{(o, \bar{t}), (\bar{t}, i)\}, \ell \cup \{(\bar{t}, \tau)\})$ *is strongly connected* ($\bar{t} \notin T$),
4. *method use:* $M = rng(\ell) \backslash \{\tau\}$,
5. *visible start: for any* $t \in T$ *such that* $t \in i\bullet$: $\ell(t) \in L_v$, *and*
6. *visible end: for any* $t \in T$ *such that* $t \in \bullet o$: $\ell(t) \in L_v$.

Note that the connectedness requirement implies that there is one unique source and one unique sink place. For the readers familiar with the work presented in [1, 3, 5]: the WF-nets defined in this paper are extended with the latter three requirements, i.e., all methods are actually used in the network, and the start transitions $i\bullet$ and stop transitions $\bullet o$ have non-τ labels. The P/T-nets shown in figures 2 and 10 are WF-nets. The structure of a WF-net allows us to define the following functions.

Definition 15 ($\underline{source}$, $\underline{sink}$, $\underline{start}$, $\underline{stop}$, $\underline{strip}$). *Let $N = (P, T, M, F, \ell)$ be a WF-net.*

1. $\underline{source}(N)$ *is the (unique) input place $i \in P$ such that $\bullet i = \emptyset$,*
2. $\underline{sink}(N)$ *is the (unique) output place $o \in P$ such that $o\bullet = \emptyset$,*
3. $\underline{start}(N) = \{t \in T \mid i \in \bullet t\}$ *is the set of start transitions,*
4. $\underline{stop}(N) = \{t \in T \mid o \in t\bullet\}$ *is the set of stop transitions, and*
5. $\underline{strip}(N) = (P', T, M, F \cap ((P' \times T) \cup (T \times P')), \ell)$ *with $P' = P \setminus \{\underline{source}(N), \underline{sink}(N)\}$ is the WF-net without source and sink place.*

Definition 14 only gives a static characterization of a WF-net. Workflows will have a life-cycle which satisfies the following requirements.

Definition 16 (Soundness). *A WF-net N with $\underline{source}(N) = i$ and $\underline{sink}(N) = o$ is said to be* sound *if and only if the following conditions are satisfied:* [1]

1. *safeness: $(N, [i])$ is safe,*
2. *proper completion: for any reachable marking $s \in [N, [i]\rangle$, $o \in s$ implies $s = [o]$,*
3. *completion option: for any reachable marking $s \in [N, [i]\rangle$, $[o] \in [N, s\rangle$, and*
4. *dead transitions: $(N, [i])$ contains no dead transitions.*

The set of all sound WF-nets is denoted $\mathcal{W}$. The first requirement states that a sound WF-net is safe. The second requirement states that the moment a token is put in place o all the other places should be empty, which corresponds to the termination of a workflow instance (i.e., a case) without leaving dangling references. The third requirement states that starting from the initial marking $[i]$, i.e., activation of the case, it is always possible to reach the marking with one token in place o, which means that it is always feasible to terminate successfully. The last requirement, which states that there are no dead transitions, corresponds to the requirement that for each transition there is an execution sequence activating this transition.

The WF-net shown in Figure 2 is sound. However, the WF-net shown in Figure 10 is not sound because it is not possible to mark the sink place o, i.e., the third requirement is violated. (In fact, all requirements except the second one are violated.)

Theorem 1 (Characterization of soundness). *Let $N = (P, T, M, F, \ell)$ be a WF-net and $\bar{N} = (P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\}, \ell \cup \{(\bar{t}, \tau)\})$ the short-circuited version of N ($\bar{t} \notin T$). N is sound if and only if $(\bar{N}, [i])$ is live and safe.*

Proof. The proof is similar to the proof of Theorem 11 in [1]. The only difference is that in this paper a stronger notion of soundness is used, which implies safeness rather than boundedness of the short-circuited net. □

[1] Note that $[i]$ and $[o]$ are *bags* containing the input respectively output place of N.

If we add a transition $\bar{t}$ to Figure 2 as described in Theorem 1, then the corresponding P/T-net is both live and safe. If we add a transition $\bar{t}$ to Figure 10, then the resulting net is not live and unbounded. Note that these observations support Theorem 1.

The fact that soundness coincides with standard properties such as liveness and safeness allows us to use existing tools and techniques to verify soundness of a given WF-net.

Lemma 1. *Let $N = (P, T, M, F, \ell)$ be a sound WF-net, i.e., $N \in \mathcal{W}$. For any $t \in T$, (i) if $i = \underline{source}(N)$ and $t \in \underline{start}(N)$, then $\bullet t = \{i\}$, and (ii) if $o = \underline{sink}(N)$ and $t \in \underline{stop}(N)$, then $t\bullet = \{o\}$.*

Proof. See [5]. □

The alphabet operator α is a function yielding the set of visible labels of all transitions of the net that are not dead.

Definition 17 (Alphabet operator α). *Let (N, s) be a marked, labeled P/T-net in $\mathcal{N}$, with $N = (P, T, M, F, \ell)$. $\alpha : \mathcal{N} \rightarrow \mathcal{P}(L_v)$ is a function such that $\alpha(N, s) = \{\ell(t) \mid t \in T \wedge \ell(t) \neq \tau \wedge t \text{ is not dead}\}$.*

Since sound WF-nets do not contain dead transitions, $\alpha(N, [i])$ equals $\{\ell(t) \mid t \in T \wedge \ell(t) \neq \tau\}$, which is denoted by $\alpha(N)$.

2.3 Branching bisimilarity

To formalize projection inheritance, we need to formalize a notion of equivalence. In this paper, we use *branching bisimilarity* [24] as the standard equivalence relation on marked, labeled P/T-nets in $\mathcal{N}$.

The notion of a *silent action* is pivotal to the definition of branching bisimilarity. Silent actions are actions (i.e., transition firings) that cannot be observed. Silent actions are denoted with the label τ, i.e., only transitions in a P/T-net with a label different from τ are observable. Note that we assume that τ is an element of L. The τ-labeled transitions are used to distinguish between external, or observable, and internal, or silent, behavior. A single label is sufficient, since all internal actions are equal in the sense that they do not have any visible effects.

In the context of workflow management, we want to distinguish *successful termination* from *deadlock*. A *termination predicate* defines in what states a marked P/T-net can terminate successfully. If a marked, labeled P/T-net is in a state where it cannot perform any actions or terminate successfully, then it is said to be in a *deadlock*. Based on the notion of soundness, successful termination corresponds to the state with one token in the sink place.

Definition 18. *The class of marked, labeled P/T-nets $\mathcal{N}$ is equipped with the following termination predicate: $\downarrow = \{(N, [o]) \mid N \text{ is a WF-net} \wedge o = \underline{sink}(N)\}$.*

To define branching bisimilarity, two auxiliary definitions are needed: (1) a relation expressing that a marked, labeled P/T-net can evolve into another marked, labeled P/T-net by executing a sequence of zero or more τ actions; (2) a predicate expressing that a marked, labeled P/T-net can terminate by performing zero or more τ actions.

Definition 19. *The relation* $\_ \Longrightarrow \_ \subseteq \mathcal{N} \times \mathcal{N}$ *is defined as the smallest relation satisfying, for any* $p, p', p'' \in \mathcal{N}$, $p \Longrightarrow p$ *and* $(p \Longrightarrow p' \wedge p' \,[\tau\rangle\, p'') \;\Rightarrow\; p \Longrightarrow p''$.

Definition 20. *The predicate* $\Downarrow \_ \subseteq \mathcal{N}$ *is defined as the smallest set of marked, labeled P/T-nets satisfying, for any* $p, p' \in \mathcal{N}$, $\downarrow p \Rightarrow \Downarrow p$ *and* $(\Downarrow p \wedge p' \,[\tau\rangle\, p) \;\Rightarrow\; \Downarrow p'$.

Let, for any two marked, labeled P/T-nets $p, p' \in \mathcal{N}$ and action $\alpha \in L$, $p \,[(\alpha)\rangle\, p'$ be an abbreviation of the predicate $(\alpha = \tau \wedge p = p') \vee p\,[\alpha\rangle\, p'$. Thus, $p\,[(\tau)\rangle\, p'$ means that zero τ actions are performed, when the first disjunct of the predicate is satisfied, or that one τ action is performed, when the second disjunct is satisfied. For any observable action $a \in L \backslash \{\tau\}$, the first disjunct of the predicate can never be satisfied. Hence, $p \,[(a)\rangle\, p'$ is simply equal to $p \,[a\rangle\, p'$, meaning that a single a action is performed.

Definition 21 (Branching bisimilarity). *A binary relation* $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{N}$ *is called a* branching bisimulation *if and only if, for any* $p, p', q, q' \in \mathcal{N}$ *and* $\alpha \in L$,

1. $p \mathcal{R} q \wedge p \,[\alpha\rangle\, p' \Rightarrow$
   $(\exists\, q', q'' :\; q', q'' \in \mathcal{N} :\; q \Longrightarrow q'' \wedge q'' \,[(\alpha)\rangle\, q' \wedge p \mathcal{R} q'' \wedge p' \mathcal{R} q')$,
2. $p \mathcal{R} q \wedge q \,[\alpha\rangle\, q' \Rightarrow$
   $(\exists\, p', p'' :\; p', p'' \in \mathcal{N} :\; p \Longrightarrow p'' \wedge p'' \,[(\alpha)\rangle\, p' \wedge p'' \mathcal{R} q \wedge p' \mathcal{R} q')$, *and*
3. $p \mathcal{R} q \Rightarrow (\downarrow p \Rightarrow \Downarrow q \wedge \downarrow q \Rightarrow \Downarrow p)$.

Two marked, labeled P/T-nets are called branching bisimilar, *denoted* $p \sim_b q$, *if and only if there exists a branching bisimulation* $\mathcal{R}$ *such that* $p \mathcal{R} q$.

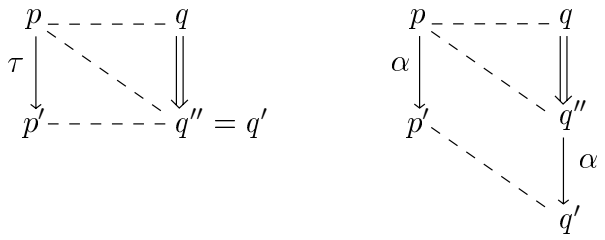

Fig. 11. The essence of a branching bisimulation.

Figure 11 shows the essence of a branching bisimulation. The firing rule is depicted by arrows. The dashed lines represent a branching bisimulation. A marked, labeled P/T-net must be able to simulate any action of an equivalent marked, labeled P/T-net after performing any number of silent actions, except for a silent action which it may or may not simulate. The third property in Definition 21 guarantees that related marked, labeled P/T-nets always have the same termination options.

Branching bisimilarity is an equivalence relation on $\mathcal{N}$, i.e., $\sim_b$ is reflexive, symmetric, and transitive. See [15] for more details and pointers to other notions of branching bisimilarity.

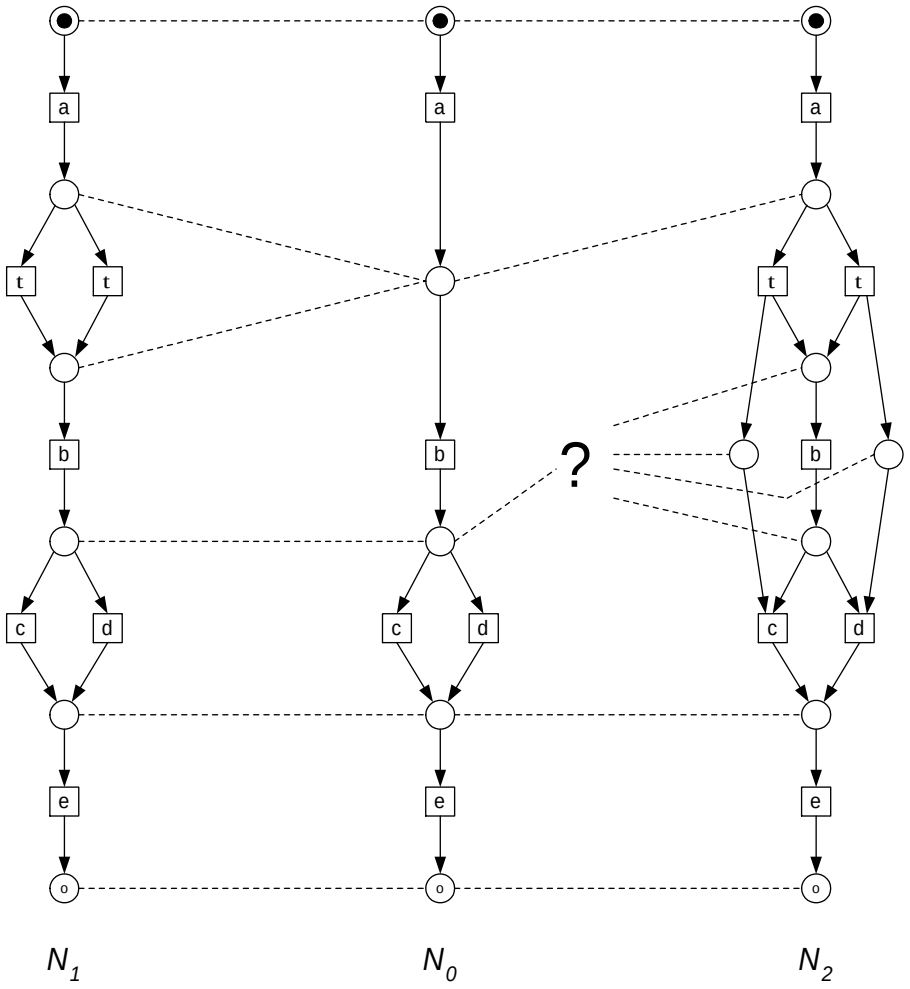

Fig. 12. Three marked WF-nets: the first two are branching bisimilar and the third one is not branching bisimilar to the other two.

To illustrate the relevance of branching bisimilarity as an equivalence notion we use the three marked WF-nets shown in Figure 12. Each of the nets has the following visible behavior: either the trace $abce$ is realized or trace $abde$ is realized. Therefore, it is interesting to investigate whether the three marked WF-nets are branching bisimilar. $(N_0, [i])$ and $(N_1, [i])$ are branching bisimilar. However, $(N_0, [i])$ and $(N_2, [i])$ are not, i.e., although they are trace equivalent $(N_0, [i]) \not\sim_b (N_2, [i])$! The reason is that in N_0 the moment of choice between c and d is made *after* the execution of b while in N_2 the choice is made *before* the execution of b. This distinction is vital when dealing with interorganizational workflows. Assume that b corresponds to sending an request to a supplier and that c is executed in case of a positive response and that d is executed in case of a negative response. In N_0 the WF-net can handle both a positive response (c) and a negative response (d) after sending the request (b). However, in N_2 the WF-net can handle either the positive or the negative response, i.e., the choice between c and d is made before the execution of b. Clearly, the latter WF-net is not acceptable, since it assumes that before sending the request the answer of the supplier is already known. This simple example shows that straightforward notions of equivalence such as trace equivalence (after abstraction of internal steps) are not selective enough for the problems addressed in this paper. Therefore, we use the more refined notion of branching bisimilarity.

Definition 22 (Behavioral equivalence of WF-nets). *For any two WF-nets N_0 and N_1 in $\mathcal{W}$, $N_0 \cong N_1$ if and only if $(N_0, [i]) \sim_b (N_1, [i])$.*

Consider the three nets shown in Figure 12: $N_0 \cong N_1$, $N_0 \ncong N_2$, and $N_1 \ncong N_2$.

2.4 Inheritance

In [7, 8, 15] four notions of inheritance have been identified. Unlike most other notions of inheritance, these notions focus on the dynamics rather than data and/or signatures of methods. These inheritance notions address the usual aspects: *substitutability* (Can the superclass be replaced by the subclass without breaking the system?), *subclassing* (implementation inheritance: Can the subclass use the implementation of the superclass?), and *subtyping* (interface inheritance: Can the subclass use or conform to the interface of the superclass?). The four inheritance notions are inspired by a mixture of these aspects.

In this paper, we restrict ourselves to one of the four inheritance notions: *projection inheritance*. In the future we hope to extend our framework with other notions of inheritance (cf. Section 8). The basic idea of projection inheritance can be characterized as follows.

> *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.*

For projection inheritance, all new methods (i.e., methods added in the subclass) are hidden. Therefore, we introduce the abstraction operator τ_I that can be used to hide methods.

Definition 23 (Abstraction). *Let $N = (P, T, M, F, \ell_0)$ be a labeled P/T-net. For any $I \subseteq L_v$, the abstraction operator τ_I is a function that renames all transition labels in I to the silent action τ. Formally, $\tau_I(N) = (P, T, M, F, \ell_1)$ such that, for any $t \in T$, $\ell_0(t) \in I$ implies $\ell_1(t) = \tau$ and $\ell_0(t) \notin I$ implies $\ell_1(t) = \ell_0(t)$.*

The definition of projection inheritance is straightforward, given the abstraction operator and branching bisimilarity as an equivalence notion.

Definition 24 (Inheritance). *For any two sound WF-nets N_0 and N_1 in $\mathcal{W}$, N_1 is a subclass of N_0 under projection inheritance, denoted $N_1 \leq_{pj} N_0$, if and only if there is an $I \subseteq L_v$ such that $(\tau_I(N_1), [i]) \sim_b (N_0, [i])$.*

It is easy to show that $\leq_{pj}$ is a partial order, i.e., $\leq_{pj}$ is reflexive, anti-symmetric, and transitive [15].

Property 1. Assuming $\cong$, as defined in Definition 22, as the equivalence on sound WF-nets $\leq_{pj}$ is a partial order.

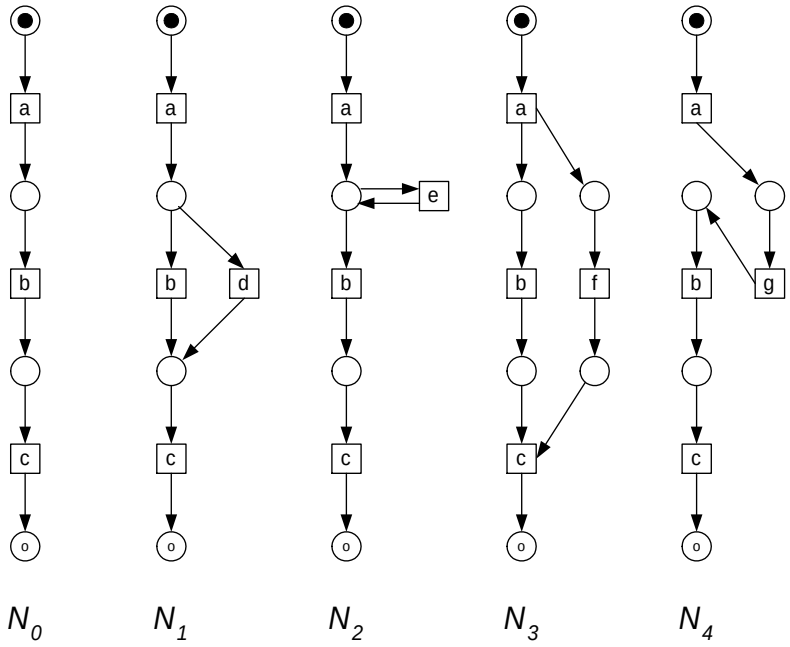

Fig. 13. N_2, N_3, and N_4 are subclasses of N_0 under projection inheritance.

Let us consider the five WF-nets shown in Figure 13 to illustrate the notion of projection inheritance. N_1 is *not* a subclass of N_0 because hiding of the new task d results in a potential trace where a is followed by c without executing b, i.e., the WF-net where d is renamed to τ 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, (2) inserting methods in-between existing methods, and (3) putting new methods in parallel with existing methods. Without proof we summarize some of the results given in [7, 8, 15].

Theorem 2 (Projection-inheritance-preserving transformation rule PPS).
Let $N_0 = (P_0, T_0, M_0, F_0, \ell_0)$ be a sound WF-net in $\mathcal{W}$. If $N = (P, T, M, F, \ell)$ is a labeled P/T-net with place $p \in P$ such that

1. $p \notin \{i, o\}$, $P_0 \cap P = \{p\}$, $T_0 \cap T = \emptyset$,
2. $(\forall\, t : t \in T : \ell(t) \notin \alpha(N_0))$,
3. $(\forall\, t : t \in T \wedge p \in {}^\bullet t : \ell(t) \neq \tau)$,
4. $(N, [p])$ *is live and safe, and*
5. $N_1 = N_0 \cup N$ *is well defined,*

then N_1 is a sound WF-net in $\mathcal{W}$ such that $N_1 \leq_{pp} N_0$.

Note that PPS can be used to construct the subclass N_2 in Figure 13 from the WF-net N_0 shown in the same figure.

Theorem 3 (Projection-inheritance-preserving transformation rule PJS).
Let $N_0 = (P_0, T_0, M_0, F_0, \ell_0)$ be a sound WF-net in $\mathcal{W}$. If $N = (P, T, M, F, \ell)$ is a labeled P/T-net with place $p \in P$ and transition $t_p \in T$ such that

1. $p \notin \{i, o\}$, $P_0 \cap P = \{p\}$, $T_0 \cap T = \{t_p\}$, $(t_p, p) \in F_0$, *and* ${}^{N}{\bullet}t_p = \{p\}$,
2. $(\forall\, t : t \in T \backslash T_0 : \ell(t) \notin \alpha(N_0))$,
3. $(N, [p])$ *is live and safe, and*
4. $N_1 = (P_0, T_0, M_0, F_0 \backslash \{(t_p, p)\}, \ell_0) \cup (P, T, M, F \backslash \{(p, t_p)\}, \ell)$ *is well defined,*

then N_1 is a sound WF-net in $\mathcal{W}$ such that $N_1 \leq_{pj} N_0$.

Transformation rule PJS can be used to construct N_4 from N_0 in Figure 13.

Theorem 4 (Projection-inheritance-preserving transformation rule $PJ3S$).
Let $N_0 = (P_0, T_0, M_0, F_0, \ell_0)$ be a sound WF-net in $\mathcal{W}$. Let $N = (P, T, M, F, \ell)$ be a labeled P/T-net. Assume that $q \in U$ is a fresh identifier not appearing in $P_0 \cup T_0 \cup P \cup T$. If N contains a place $p \in P$ and transitions $t_i, t_o \in T$ such that

1. ${}^{N}{\bullet}p = \{t_o\}$, $p{\bullet}^{N} = \{t_i\}$,
2. $P_0 \cap P = \emptyset$, $T_0 \cap T = \{t_i, t_o\}$,
3. $(\forall\, t : t \in T \backslash T_0 : \ell(t) \notin \alpha(N_0))$,
4. $(N, [p])$ *is live and safe,*
5. $N_1 = N_0 \cup (P \backslash \{p\}, T, F \backslash \{(p, t_i), (t_o, p)\}, \ell)$ *is well defined,*
6. q *is implicit in* $(N_0^q, [i])$ *with* $N_0^q = (P_0 \cup \{q\}, T_0, F_0 \cup \{(t_i, q), (q, t_o)\}, \ell_0)$, *and*

7. N_0^q *is a sound WF-net,*

then N_1 *is a sound WF-net in* $\mathcal{W}$ *such that* $N_1 \leq_{pj} N_0$.

Transformation rule $PJ3S$ can be used to construct subclass N_3 from superclass N_0 in Figure 13.

Rule PPS can be used to insert a loop or iteration at any point in the process, provided that the added part always returns to the initial state. Rule PJS can be used to insert new methods by replacing a connection between a transition and a place by an arbitrary complex subnet. Rule $PJ3S$ can be used to add parallel behavior, i.e., new methods which are exectuted in parallel with existing methods. The inheritance-preserving transformation rules distinguish the work presented in [7, 8, 15] from earlier work on inheritance. The rules correspond to design constructs that are often used in practice, namely iteration, sequential composition, and parallel composition. If a designer sticks to these rules, inheritance is guaranteed!

2.5 Greatest common divisor/least common multiple of WF-nets

Projection inheritance provides a (partial) ordering of WF-nets. This ordering, like any ordering, can be used to define the notions of *Greatest Common Divisor* (GCD) and *Least Common Multiple* (LCM).

Definition 25 (GCD, LCM). *Let* $N_0, N_1, \ldots, N_{n-1}$, *where* n *is some natural number, and* N *be sound WF-nets in* $\mathcal{W}$.
Net N *is a* Greatest Common Divisor *(GCD) of* $N_0, N_1, \ldots, N_{n-1}$ *if and only if*

1. $(\forall\, k : 0 \leq k < n : N_k \leq_{pj} N)$ *and,*
2. *for any sound WF-net* N' *such that* $(\forall\, k : 0 \leq k < n : N_k \leq_{pj} N')$ *and* $N' \leq_{pj} N$, $N' \cong N$.

Net N *is a* Least Common Multiple *(LCM) of* $N_0, N_1, \ldots, N_{n-1}$ *if and only if*

1. $(\forall\, k : 0 \leq k < n : N \leq_{pj} N_k)$ *and,*
2. *for any sound WF-net* N' *such that* $(\forall\, k : 0 \leq k < n : N' \leq_{pj} N_k)$ *and* $N \leq_{pj} N'$, $N' \cong N$.

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. Consider for example the WF-nets N_0, N_2, N_3, and N_4 shown in Figure 13. The GCD of these four nets is N_0. 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 subclass of N_0, N_2, N_3, and N_4 and at the same time a subclass of N_0. N_0 is reasonable choice for the GCD: Each of the nets executes a, b, and c in sequential order. Figure 14 shows $N_{GCD} = N_0$ as the GCD of N_0, N_2, N_3, and N_4. Figure 14 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}. Therefore, N_{LCM} is indeed the LCM of N_0, N_2, N_3, and N_4. Any sequence

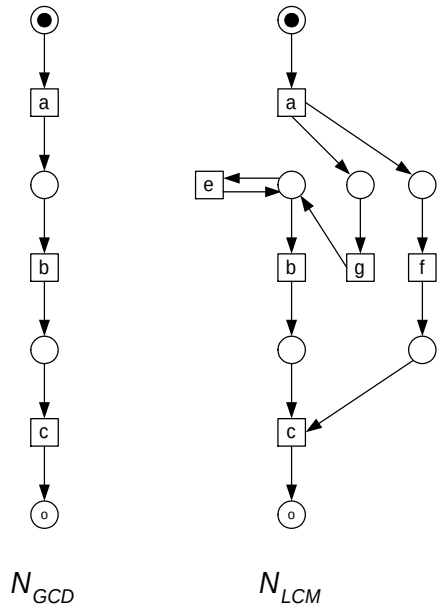

Fig. 14. The greatest common divisor N_{GCD} and least common multiple N_{LCM} of N_0, N_2, N_3, and N_4 shown in Figure 13.

generated by one of the four nets can also be generated by N_{LCM} after the appropriate abstraction.

As Figure 14 indicates, the size of the GCD is typically smaller than the LCM. This may be confusing since the GCD is larger with respect to the ordering $\leq_{pj}$. This apparent paradox can easily be explained by the following property: $N_0 \leq_{pj} N_1$ implies $\alpha(N_0) \supseteq \alpha(N_1)$.

Figure 14 shows the GCD and LCM of a set of WF-nets. Since $\leq_{pj}$ is not a total ordering, the following two questions are interesting: (1) Is there always a GCD/LCM?, and (2) Is the GCD/LCM unique? To answer the first question, we use the following property.

Property 2. Let N_0 and N_1 be two sound WF-nets in $\mathcal{W}$ such that $N_0 \leq_{pj} N_1$. There is no *infinite* chain $N^0 \leq_{pj} N^1 \leq_{pj} \ldots$ of *different* (with respect to $\cong$) sound WF-nets $N^0, N^1, \ldots \in \mathcal{W}$ such that $N_0 \leq_{pj} N^0 \leq_{pj} N^1 \leq_{pj} \ldots \leq_{pj} N_1$.

Proof. Let N and N' be two sound WF-nets in $\mathcal{W}$ such that $N \leq_{pj} N'$. The following three observations are important. First, $\alpha(N') \subseteq \alpha(N)$. Second, if $N \ncong N'$, then $\alpha(N') \subset \alpha(N)$. Third, $\alpha(N) \backslash \alpha(N')$ is finite.

Let $N^0 \leq_{pj} N^1 \leq_{pj} \ldots$ be an infinite chain of different sound WF-nets $N^0, N^1, \ldots \in \mathcal{W}$ such that $N_0 \leq_{pj} N^0 \leq_{pj} N^1 \leq_{pj} \ldots \leq_{pj} N_1$. It follows from the first two of the above observations that $\alpha(N_1) \subseteq \ldots \subset \alpha(N^1) \subset \alpha(N^0) \subseteq \alpha(N_0)$. The third observation above states that $\alpha(N_0) \backslash \alpha(N_1)$ is finite. However, this yields a contradiction, which proves the property. □

Theorem 5 (Existence of a GCD). *Let $N_0, N_1, \ldots, N_{n-1}$, where n is some natural number, be n sound WF-nets in $\mathcal{W}$. There exists a GCD of $N_0, N_1, \ldots, N_{n-1}$.*

Proof. Property 2 states that there are no infinite chains in-between any two sound WF-nets related by $\leq_{pj}$. Consequently, to prove the existence of a GCD, it suffices to show that there exists a sound WF-net that is a superclass of all variants $N_0, N_1, \ldots, N_{n-1}$. Let N_τ be the WF-net containing one task labeled τ, i.e., $N_\tau = (\{i, o\}, \{\tau\}, \emptyset, \{(i, \tau), (\tau, o)\}, \{(\tau, \tau)\})$. Clearly, N_τ is a superclass of any sound WF-net. Thus, N_τ is a superclass of each of the variants, which proves the existence of a GCD. □

Given any set of WF-nets it is possible to find a GCD. However, it is not always possible to find an LCM. Consider for example two WF-nets both containing two transitions labeled a and b. If in one net a is executed before b and in the other net b is executed before a, then it is not possible to construct an LCM. Note that the notion of LCM is related to *multiple inheritance*, i.e., the LCM needs to inherit the properties of multiple WF-nets. In many object-oriented frameworks multiple inheritance is difficult to handle. Therefore, it is not surprising that the LCM does not necessarily exist.

The answer to the first question ("Is there always a GCD/LCM?") is positive with respect to the GCD and negative with respect to the LCM. The answer to the second question ("Is the GCD/LCM unique?") is negative for both GCD's and LCM's. Even if we consider behavioral equivalence (i.e., modulo branching bisimulation) it is easy to find counter examples. Consider for example the two WF-nets both containing two transitions labeled a and b but ordered differently. Both the WF-net containing just the transition a and the WF-net containing just the transition b are GCD's, i.e., there is not one unique GCD. Moreover, the latter two nets (i.e., one WF-net containing just transition a and another WF-net containing just transition b) do not have a unique LCM. The WF-net where transition a is followed transition b is an LCM, but the WF-net where transition b is followed transition a is also an LCM.

For an arbitrary set of WF-nets there may be several really different GCD's and LCM's or there may be no LCM's at all. However, as the following theorem shows, it is quite straightforward to formulate requirements such that there is one unique GCD/LCM.

Theorem 6. *Let $N_0, N_1, \ldots, N_{n-1}$, where n is some natural number, be n sound WF-nets in $\mathcal{W}$.*
If there is a sound WF-net N in $\mathcal{W}$ such that

1. *$(\forall\, k : 0 \leq k < n : N_k \leq_{pj} N)$ and,*
2. *for any sound WF-net N' in $\mathcal{W}$, $(\forall\, k : 0 \leq k < n : N_k \leq_{pj} N')$ implies $N \leq_{pj} N'$,*

then N is a GCD of $N_0, N_1, \ldots, N_{n-1}$ and every sound WF-net N_{GCD} in $\mathcal{W}$ that is also a GCD of $N_0, N_1, \ldots, N_{n-1}$ is branching bisimilar to N (i.e., $N_{GCD} \cong N$).
If there is a sound WF-net N in $\mathcal{W}$ such that

1. *$(\forall\, k : 0 \leq k < n : N \leq_{pj} N_k)$ and,*
2. *for any sound WF-net N' in $\mathcal{W}$, $(\forall\, k : 0 \leq k < n : N' \leq_{pj} N_k)$ implies $N' \leq_{pj} N$,*

then N is an LCM of $N_0, N_1, \ldots, N_{n-1}$ and every sound WF-net N_{LCM} in $\mathcal{W}$ that is also an LCM of $N_0, N_1, \ldots, N_{n-1}$ is branching bisimilar to N (i.e., $N_{LCM} \cong N$).

Proof. The proofs of the two parts of the theorem are very similar. Therefore, we only prove that there exists a unique GCD. Let N be a sound WF-net satisfying the first pair of requirements stated in the theorem.

Consider Definition 25 (GCD). Assume that N' is a sound WF-net such that $(\forall\, k : 0 \leq k < n : N_k \leq_{pj} N')$ and $N' \leq_{pj} N$. It follows that $N \leq_{pj} N'$, which in combination with $N' \leq_{pj} N$ implies that $N \cong N'$ ($\leq_{pj}$ is a partial order; see Property 1). Thus, net N satisfies the requirements in Definition 25, which means that it is a GCD.

Assume that N_{GCD} is an arbitrary GCD of the n variants. It follows from the assumptions on N that $N \leq_{pj} N_{GCD}$. Consequently, Definition 25 yields that $N \cong N_{GCD}$, which completes the proof. □

It is easy to verify that N_{GCD} and N_{LCM} shown in Figure 14 satisfy the requirements stated in Theorem 6. Therefore, N_{GCD} is the only GCD of the WF-nets N_0, N_2, N_3, and N_4 shown in Figure 13 (modulo branching bisimulation) and N_{LCM} is the unique LCM of the same set of WF-nets. Whenever we use the notions of GCD and LCM in this paper, we use Theorem 6 to make sure that they are unique.

3 Interorganizational workflows

The WF-net introduced in Section 2.2 specifies one workflow process in isolation. The goal of the paper is to tackle problems related to interorganizational workflows. Therefore, we define an *interorganizational workflow net* (IOWF-net) as a set of WF-nets connected through *channels*.

Definition 26 (IOWF-net). *An* interorganizational workflow net *(IOWF-net) is a tuple* $(C, n, N_0, N_1, \ldots, N_{n-1}, G)$ *where:*

1. $C \subseteq U$ *is a finite set of* channels,
2. $N_0, N_1, \ldots, N_{n-1}$ *are* n *WF-nets such that:*
   (a) $(\forall\, k : 0 \leq k < n : N_k = (P_k, T_k, M_k, F_k, \ell_k))$,
   (b) $(\forall\, k, l : 0 \leq k < l < n : (P_k \cup T_k \cup M_k) \cap (P_l \cup T_l \cup M_l) = \emptyset)$, *and*
   (c) $(\forall\, k : 0 \leq k < n : (P_k \cup T_k \cup M_k) \cap C = \emptyset)$,
3. $M = (\cup\, k : 0 \leq k < n : M_k)$ *is the union of methods, and*
4. $G \subseteq (C \times M) \cup (M \times C)$ *is a set of directed arcs, called the* channel flow relation.

An IOWF-net consists of a set of interconnected WF-nets. The interconnection structure is specified by a set of channels C, a set of methods M, and a channel flow relation G. Figures 3, 4, and 5 show an example of an IOWF-net. Figure 3 shows the four channels *order*, *specification*, *cost_statement*, and *product*. Figures 4 and 5 show the two WF-nets. Only the methods (i.e., labels) attached to transitions in the WF-nets are visible. N_0^{part} has four methods: *send_order*, *create_specification*, *process_cost_statement*, and *handle_product*. N_1^{part} also has four methods: *receive_order*, *process_specification*, *create_cost_statement*, and *ship_product*. Figure 3 also shows the channel flow relation G, e.g., method *send_order* is connected to channel *order*, channel *order* is connected to method *receive_order*, etc.

The semantics of an IOWF-net are given in terms of a labeled P/T net, i.e., by taking the union of all WF-nets, adding a place for each channel, connecting transitions to

these newly added places as specified by G, and removing superfluous source and sink places, the IOWF-net is transformed into a labeled P/T net. We call this the *flattening* of the interorganizational workflow. The following definition describes a function $\underline{flat}$ which transforms any IOWF-net into a labeled P/T net.

Definition 27 ($\underline{flat}(Q)$). *Let $Q = (C, n, N_0, N_1, \ldots, N_{n-1}, G)$ be an IOWF-net. $N = (P, T, M, F, \ell)$ is the such that:*

1. $P = C \cup (\cup k : 0 \leq k < n : P_k)$,
2. $P_i = \{\underline{source}(N_k) \mid 0 \leq k < n\}$,
3. $P_o = \{\underline{sink}(N_k) \mid 0 \leq k < n\}$,
4. $T = (\cup k : 0 \leq k < n : T_k)$,
5. $M = (\cup k : 0 \leq k < n : M_k)$,
6. $\ell = (\cup k : 0 \leq k < n : \ell_k)$, *and*
7. $F = (\cup k : 0 \leq k < n : F_k) \cup \{(p, t) \in P \times T \mid (p, \ell(t)) \in G\} \cup \{(t, p) \in T \times P \mid (\ell(t), p) \in G\}$.

Let $N' = (P', T, M, F', \ell)$ be the labeled P/T net obtained by removing all places $X = \{p \in P_i \mid {}^{N}\!\bullet(p^{N}\!\bullet) \neq \{p\}\} \cup \{p \in P_o \mid ({}^{N}\!\bullet p)^{N}\!\bullet \neq \{p\}\}$, i.e., $P' = P \backslash X$ and $F' = F \cap ((P' \times T) \cup (T \times P'))$. $\underline{flat}(Q) = N'$ is the flattened IOWF-net.

The definition $\underline{flat}$ is fairly straightforward except for the removal of source and sink places. Source place $\underline{source}(N_k)$ is removed if and only if there is a transition which consumes tokens from $\underline{source}(N_k)$ and at least one other place, i.e., only source places which serve as the only input place for all connected transitions are kept. Similarly, sink place $\underline{sink}(N_k)$ is removed if and only if there is a transition which produces tokens for $\underline{sink}(N_k)$ and at least one other place. Figure 15 shows the flattened interorganizational workflow defined by figures 3, 6 and 8. Note that the source place and sink place of the WF-net shown in 8 have been removed in the flattened net: the source place of N_1^{priv} was not the only input place of *receive_order* and the sink place of N_1^{priv} was not the only output place of *ship_product*. In both cases the start/stop transition is connected to a place representing one of the channels. Note that the WF-net shown in Figure 2 is the flattened interorganizational workflow defined by figures 3, 4, and 5, i.e., the semantics of the interorganizational workflow are not changed after partitioning N^{publ} over a contractor subflow and a subcontractor subflow.

As is illustrated by Figure 15, the channels connect the WF-nets constituting the interorganizational workflow. It is easy to see that these connections may cause deadlocks. For example, if in the WF-net shown in Figure 15 the first procedure is chosen, then the workflow process deadlocks, i.e., the state obtained after executing *send_order*, *receive_order*, *decide*, and *procedure_1* is dead. Another undesirable phenomenon is called *multiple activation*. To explain this anomaly we introduce the term *activation*. A subflow in an IOWF-net is activated if at least one of the places in the subflow is marked (except the source and sink place). Note that a subflow becomes activated after one of the start transitions fires. A subflow becomes deactivated if each of the subflow places is empty after one of the stop transitions fires. Ideally, every activation is followed by a deactivation. However, in an interorganizational workflow an activated subflow could become activated again without being deactivated first. Such a multiple activation may

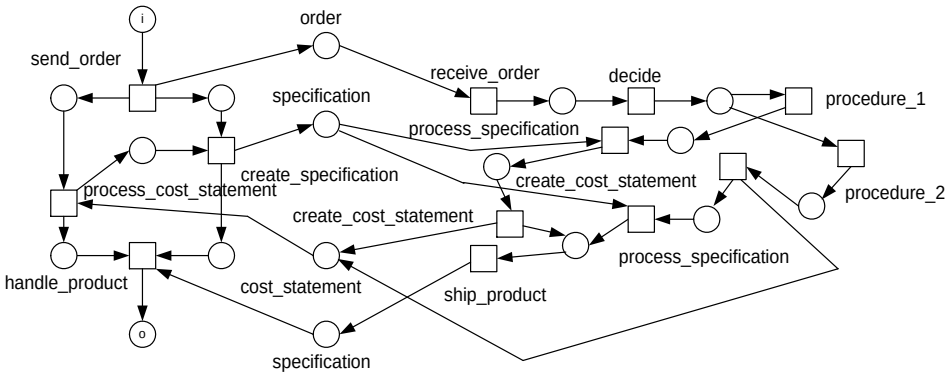

Fig. 15. The flattened interorganizational workflow composed of the private WF-nets shown in figures 6 and 8.

lead to all kinds of anomalies because the subflow embedded in the interorganizational workflow exhibits behavior which is not possible in the WF-net in isolation. For example multiple activation of a subflow k may result in states not considered when establishing the soundness of WF-net N_k. To formulate the requirement that there is no multiple activation, we define the notion of *activation safeness*.

Definition 28 (Activation safeness). *Let (N, s) be a marked, labeled P/T-net in $\mathcal{N}$, where $N = (P, T, M, F, \ell)$. A subset of places $P' \subseteq P$ is* activation safe *in (N, s) if and only if for any reachable state $s' \in [N, s\rangle$, any transition $t \in {\bullet}P' \backslash P'{\bullet}$, and any place $p \in P'$: $(N, s')[t\rangle$ implies $s'(p) = 0$.*

A set of places P' is activation safe if all transitions producing tokens for P' but not consuming tokens from P' are not enabled as long as there are tokens in P'. It is easy to see that there is no multiple activation of a subflow in an interorganizational workflow if and only if the places of each subflow are activation safe. Using Definition 28, we can formulate the notion of *soundness* for IOWF-nets.

Definition 29 (Soundness). *Let $Q = (C, n, N_0, N_1, \ldots, N_{n-1}, G)$ be an IOWF-net and let $N = (P, T, M, F, \ell)$ be the corresponding flattened net, i.e., $N = \underline{flat}(Q)$. Q is sound if and only if:*

1. *$(\forall\, k : 0 \leq k < n : N_k \in \mathcal{W})$, i.e., all subflows are sound,*
2. *$N \in \mathcal{W}$, i.e., the flattened IOWF-net is a sound WF-net, and*
3. *$(\forall\, k : 0 \leq k < n : P_k \backslash \{\underline{source}(N_k), \underline{sink}(N_k)\}$ is activation safe in $(N, [i]))$, i.e., there is no multiple activation.*

The first two requirements are fairly straightforward: A sound IOWF-net is composed of sound WF-nets and the flattened IOWF-net is also a sound WF-net. The last requirement has been added to avoid multiple activation. The IOWF-net defined by figures 3, 4, and 5 is an example of a sound IOWF-net: it is composed of two sound WF-nets, the flattened IOWF-net is a sound WF-net, and there is no multiple activation. The IOWF-net defined by figures 3, 6, and 8 is not sound because the flattened net is not sound.

The first two requirements in Definition 29 can be checked using the result stated in Theorem 1. The last requirement does not correspond to well-established notions such as liveness and safeness and may be hard to check since there are no efficient analysis techniques/tools to verify this requirement. Therefore, we introduce a stronger requirement which can be validated syntactically (i.e., based on the structure of the flattened net). This requirement states that there is not a path from a transition inside one of the subflows to one of its start transitions not containing one of its stop transitions, i.e., the topology of the net guarantees that a subflow cannot trigger itself indirectly before it is deactivated. In other words: there is no *self triggering*. The following property defines the absence of self triggering and shows that the absence of self triggering assures that there is no multiple activation.

Property 3 (Self triggering). Let $Q = (C, n, N_0, N_1, \ldots, N_{n-1}, G)$ be an IOWF-net satisfying the first two requirements stated in Definition 29 (i.e., all subflows are sound and the flattened IOWF-net N is a sound WF-net). If $(\forall\, k, t, t' : 0 \leq k < n \wedge t \in T_k \wedge t' \in \underline{start}(N_k)$: all non-trivial directed paths in N from t to t' contain at least one occurrence of a transition in $\underline{stop}(N_k)$) and $(\forall\, k : 0 \leq k < n : (\cap\, t : t \in \underline{start}(N_k) : {}^{N}\!\bullet t) \neq \emptyset)$ (i.e., start transitions share input places), then Q is sound.

Proof. To prove this property, we use proof by contradiction, i.e., we assume that for some subflow $k \in D$, there is a firing sequence σ such that $(N, [i])\ [\sigma\rangle\ (N, s)$, $t \in \underline{start}(N_k)$, $(N, s)[t\rangle$, and $p \in P_k$ is marked in s. Without loss of generality, we further assume that s was the first state in the sequence having these properties (i.e., a start transition is enabled while a place in P_k is marked). Partition the sequence σ in two subsequences σ_1 and σ_2 such that σ_2 contains all firings since the last firing of a transition in $\underline{stop}(N_k)$, i.e., σ_1 is either empty or ends with the last firing of a transition in $\underline{stop}(N_k)$. The first sequence ends in state s' (i.e., $(N, [i])\ [\sigma_1\rangle\ (N, s')$). Note that in s' all places in P_k are empty. (Otherwise there would have been a prefix of σ containing the anomaly.) Now we concentrate on the second subsequence: $(N, s')\ [\sigma_2\rangle\ (N, s)$. In this sequence no transition in $\underline{stop}(N_k)$ fires. Therefore, we remove all transitions $\underline{stop}(N_k)$ from N and name the new net N'. Note that $(N', s')\ [\sigma_2\rangle\ (N', s)$. The requirement that all non-trivial directed paths in N from a transition inside N_k to one of the start transitions in N_k contain at least one of the stop transitions in N_k implies that we can partition the transitions of N' in two subsets T_X and T_Y such that $\{t \in T \backslash T_k \mid t {}^{N'}\!\bullet\ \cap\ {}^{N'}\!\bullet \underline{start}(N_k) \neq \emptyset\} \subseteq T_X$, $T_k \subseteq T_Y$, and ${}^{N'}\!\bullet T_X \cap T_Y\, {}^{N'}\!\bullet\ = \emptyset$ because all stop transitions have been removed. Now we apply the well-known exchange lemma (see for example page 23 in [21]) which allows us to project σ_2 onto the transitions in T_X and T_Y: σ_{2X} and σ_{2Y}. Since ${}^{N'}\!\bullet T_X \cap T_Y\, {}^{N'}\!\bullet\ = \emptyset$, the exchange lemma shows that we can first execute σ_{2X} followed by σ_{2Y}. Let state s'' be the state after executing σ_{2X}, i.e., $(N', s')\ [\sigma_{2X}\rangle\ (N', s'')$. It is easy to see that in s'' at least one of the input places of the start transitions of N_k contains multiple tokens, because start transitions share input places. (Note that σ_{2Y} marks a place in P_k, i.e., fires at least one start transition of N_k, and also enables a start transition of N_k without adding any new tokens to the input places.) Therefore, the safeness property is violated. The sequence composed of σ_1 followed by σ_{2X} is also possible in $(N, [i])$. Therefore, $(N, [i])$ cannot be a sound WF-net and we find a contradiction. □

Property 3 shows that the only way that a subflow becomes activated multiple times (i.e., the place is not activation safe), is through self triggering.

Based on the notion of soundness of IOWF-nets (Definition 29) and the semantics of IOWF-nets (Definition 27), we define requirements for a partitioning to be valid.

Definition 30 (Valid partitioning). *Let N be a sound WF-net and Q be an IOWF-net. Q is a* valid partitioning *of N if and only if Q is sound and $N = \underline{flat}(Q)$.*

The IOWF-net defined by figures 3, 4, and 5 is a valid partitioning of the WF-net shown in Figure 2.

4 P2P approach

The example used in the introduction is a nice illustration of the anomalies that can occur if business partners do not have a common understanding of the interorganizational workflow at hand. Many problems that are not likely to occur in a workflow which is under the supervision of one organizational unit will occur in a workflow partitioned over multiple organizations. Recent developments such as the rise of the WWW, business-to-business E-commerce, and networked virtual companies, have fueled the need for more (complex) interorganizational workflows. To avoid anomalies such as interorganizational deadlocks and livelocks on the one hand and to allow local autonomy for the organizational units involved on the other hand, we propose the P2P (Public-to-Private) approach. This approach is based on projection inheritance and consists of three steps. These steps were already introduced informally in Section 1.

Definition 31 (P2P approach). *Let $D = \{0, 1, \ldots, n-1\}$ be a set of n domains. The* Public-To-Private *(P2P) approach consists of the following three steps:*

Step 1 *Create a* public workflow $N^{publ} = (P^{publ}, T^{publ}, M^{publ}, F^{publ}, \ell^{publ})$ *such that N^{publ} is a sound WF-net.*

Step 2 *Map each task onto one of the domains, i.e., construct a function $\underline{map} : T^{publ} \rightarrow D$ and a valid partitioning $Q^{part} = (C, n, N_0^{part}, N_1^{part}, \ldots, N_{n-1}^{part}, G)$ of N^{publ} such that for all $k \in D$: $N_k^{part} = (P_k^{part}, T_k^{part}, M_k^{part}, F_k^{part}, \ell_k^{part})$ and $T_k^{part} = \{t \in T^{publ} \mid \underline{map}(t) = k\}$. Q^{part} is called the* partitioned public workflow *and for each domain $k \in D$: N_k^{part} is called the* public part *of k.*

Step 3 *For each domain $k \in D$ define a sound WF-net $N_k^{priv} = (P_k^{priv}, T_k^{priv}, M_k^{priv}, F_k^{priv}, \ell_k^{priv})$ such that N_k^{priv} is a subclass of N_k^{part} under projection inheritance (i.e., $N_k^{priv} \leq_{pj} N_k^{part}$), the labels of start and stop transitions are not changed (i.e., $\{\ell_k^{priv}(t) \mid t \in \underline{start}(N_k^{priv}) \cup \underline{stop}(N_k^{priv})\} \subseteq \alpha(N_k^{part})$), and $Q^{overall} = (C, n, N_0^{priv}, N_1^{priv}, \ldots, N_{n-1}^{priv}, G)$ is an IOWF-net. N_k^{priv} is called the* private workflow *of domain k, $Q^{overall}$ is called the* overall workflow, *and $N^{overall} = \underline{flat}(Q^{overall})$ is called the* overall WF-net.

In the first step, the public workflow is created. This workflow is specified in terms of a sound WF-net N^{publ} and serves as a *contract* between all business partners involved. Figure 2 shows an example of such a public workflow. In the second step, the

public workflow is partitioned over the set of domains D. Note that each domain corresponds to an organizational entity. For the definition of the P2P approach, we prefer to use the more neutral term "domain" instead of terms like "business partner" and "organizational unit". Figures 3, 4, and 5 show an example of such a partitioning over two domains. Note that the partitioned workflow is a valid interorganizational workflow Q^{part} as defined in definitions 26 and 30. At first glance it may seem that this requirement is rather restrictive. This is not the case, as we will motivate later. As a result of the partitioning, each fragment of the partitioned workflow corresponds to one of the domains and is represented by a sound WF-net. The WF-net N_k^{part} of a domain k is called the public part of k. In the final step, the public parts are replaced by private workflows. Each private workflow corresponds to the actual workflow as it is executed in one of the domains. The key of the P2P approach is that each private workflow N_k^{priv} is a subclass of the corresponding private workflow N_k^{part} under projection inheritance, i.e., $N_k^{priv} \leq_{pj} N_k^{part}$. Moreover, a private workflow is not allowed to change the labels of start and stop transitions. This requirement follows from the fact that at the interorganizational level it has to be clear whether a domain is active or not. Figures 7 and 9 show two private workflows satisfying the requirements formulated in Step 3. The interganizational workflow obtained by connecting the private workflows is called the overall workflow $Q^{overall}$. Note that figures 3, 7, and 9 describe such an overall workflow. Since $Q^{overall}$ is an IOWF-net, we can use function $\underline{flat}$ to obtain the overall WF-net $N^{overall} = \underline{flat}(Q^{overall})$. In the next section we will show that the fattened IOWF-net is indeed a WF-net. In fact we will show both the WF-net $N^{overall}$ and the IOWF-net $Q^{overall}$ are sound. Figure 16 shows the fattened IOWF-net of the overall workflow described in figures 3, 7, and 9. It is easy to verify that the result shown in Figure 16 is indeed a sound WF-net.

The P2P approach does not impose any restrictions on the public workflow, i.e., any sound WF-net can be promoted to public workflow. However, the requirement that the partitioning in Step 2 has to be valid may seem quite restrictive. Recall that the partitioning is only valid if all public parts (i.e., local fragments of the workflow) are sound, there is no multiple activation, and the flattened IOWF-net equals the public workflow. To illustrate the implications of these prerequisites consider the three WF-nets shown in Figure 17. Each of the four WF-nets consists of four sequential tasks: *prepare*, *produce*, *assemble*, and *ship*. The dashed lines indicate the partitioning of the corresponding WF-nets. The first WF-net, i.e., Figure 17(a), is partitioned vertically: *prepare* and *produce* are mapped onto one domain and *assemble* and *ship* are mapped onto another domain. It is easy to see that this partitioning is valid. The WF-net shown in Figure 17(b) is partitioned horizontally: *prepare* and *ship* are mapped onto one domain and *produce* and *assemble* are mapped onto another domain. This partitioning is not valid because the first fragment, i.e., the P/T net on the left-hand-side of the line in Figure 17(b), is not a WF-net. The causal relation between *prepare* and *ship* via *produce* and *assemble* is removed by partitioning the WF-net. The P2P requires each fragment to be a sound WF-net, i.e., each public part of the workflow in isolation should be a correct workflow. Clearly, the requirement stated in Step 2 is very restrictive. However, a closer observation of Figure 17(b) shows that there is no apparent reason why the horizontal partitioning should not be allowed. As Figure 17(c) shows the problem can easily be

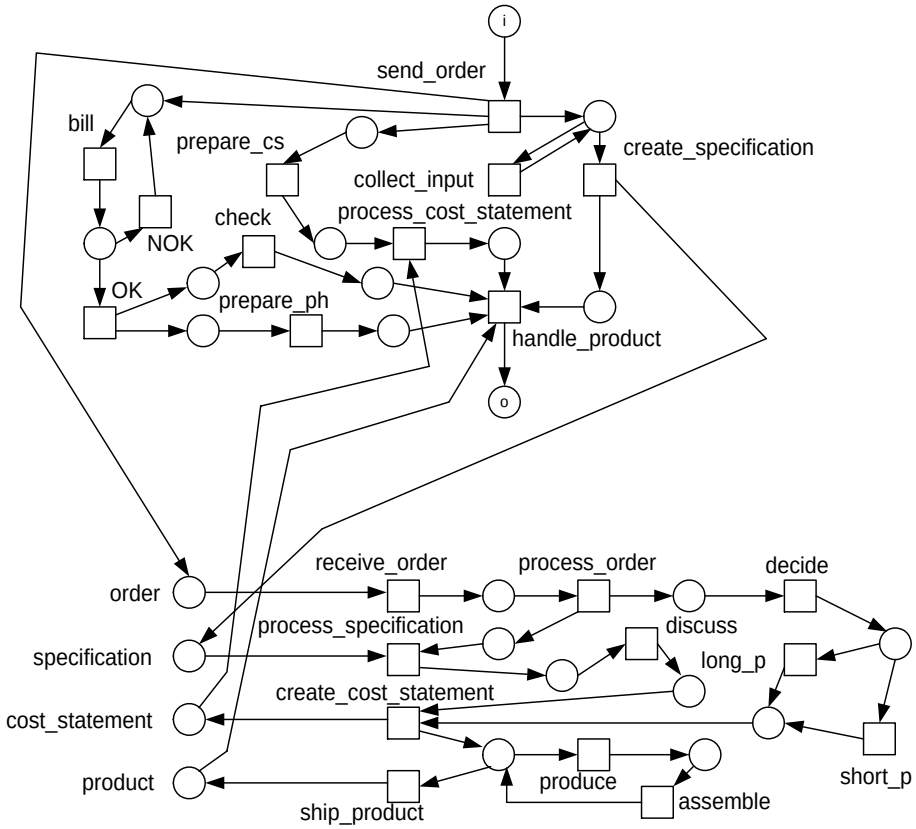

Fig. 16. The flattened IOWF $N^{overall}$ composed of the private WF-nets shown in figures 7 and 9.

solved by adding an *implicit place c4*. The horizontal partitioning of the WF-net with *c4* is valid. This example shows that if a partitioning is not valid, it is often possible to "massage" the public workflow a bit to make a valid partitioning possible.

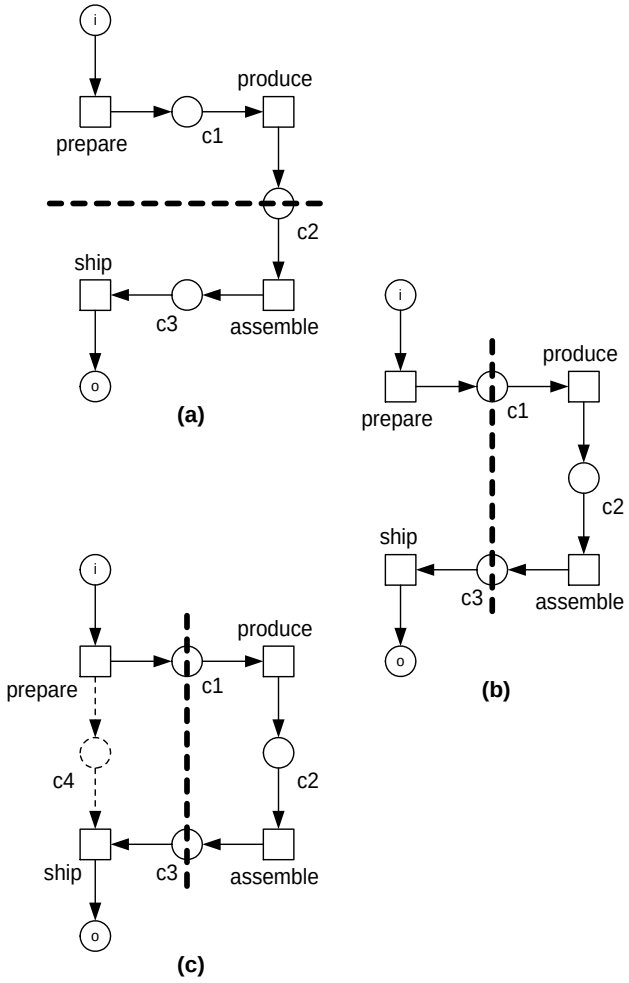

Fig. 17. A valid partitioning (a), a non-valid partitioning (b), and a valid partitioning obtained by adding implicit place *c4* (c).

Place *c4* is called implicit since it does not influence the behavior of the WF-net, i.e., a place of a marked P/T net is said to be *implicit* or *redundant* if and only if it does not depend on the number of tokens in the place whether any of its output transitions is enabled by some reachable marking.

Definition 32 (Implicit place). *Let (N, s) with $N = (P, T, M, F, \ell)$ be a marked, labeled, ordinary P/T net. A place $p \in P$ is called* implicit *in (N, s) if and only if, for all reachable markings $s' \in [N, s\rangle$ and transition $t \in p\bullet$, $s' \geq \bullet t \backslash \{p\} \Rightarrow s' \geq \bullet t$.*

Implicit places and their properties have been studied in [18, 20]. Adding implicit places does not change the behavior. In fact, extending a WF-net with implicit places yields a P/T net which is branching bisimilar to the original net [15]. From a computational point of view, it may be quite expensive to check whether a place is implicit. However, several authors have investigated techniques to find structural implicit places [17, 18, 20]. A structural implicit place is a place which is guaranteed to be implicit by the structure of the Petri net. Every structural implicit place is an implicit place, but there may be implicit places which are not structural implicit. Since the set of all structural implicit places can be found without constructing the reachability graph, it allows for very efficient analysis techniques. For this particular application, it suffices to only consider structural implicit places. Moreover, it is quite easy to build a facility which (semi-)automatically adds implicit places where needed. For more information on adding (structural) implicit places, we refer to [4, 6].

By adding implicit places it is possible to make any partitioning valid as long as each public part of the public workflow has a clear starting point and a clear ending point, i.e., subflow k is activated by firing one of its start transitions and is deactivated by firing one of its stop transitions, every activation/deactivation is communicated via a channel or the source/sink place of the public workflow, and there is no multiple activation. These conditions are quite reasonable: it should always be clear if a domain is activated or not. If these conditions are met, then the partitioning can be made valid by adding (structural) implicit places.

5 The overall workflow realizes the public workflow

The P2P approach starts with the creation of a public workflow which serves as some kind of contract. Then, the P2P approach partitions the public workflow and creates a set of private workflows which together constitute the overall workflow. One can think of the overall workflow as the interorganizational workflow actually being executed. In this section, we show that when using the P2P approach the overall workflow in fact realizes the public workflow. To be more precise, we will show that:

- the flattened overall workflow is a sound WF-net (i.e., $N^{overall} \in \mathcal{W}$),
- the overall workflow $Q^{overall}$ is sound, and
- the overall WF-net is a subclass of the public workflow under projection inheritance (i.e., $N^{overall} \leq_{pj} N^{publ}$).

To prove these properties, we start this section with a rather complex but fairly general theorem. The theorem states that under certain conditions, a subflow can be replaced a subclass subflow without endangering soundness and yielding a subclass.

Figure 18 illustrates the essence of Theorem 7: Consider a sound WF-net N_0 composed of N_A and N_B. N_A and N_B communicate through a set of common places $P_A \cap P_B$. N_B is chosen in such a way that if we remove the places $P_A \cap P_B$ and

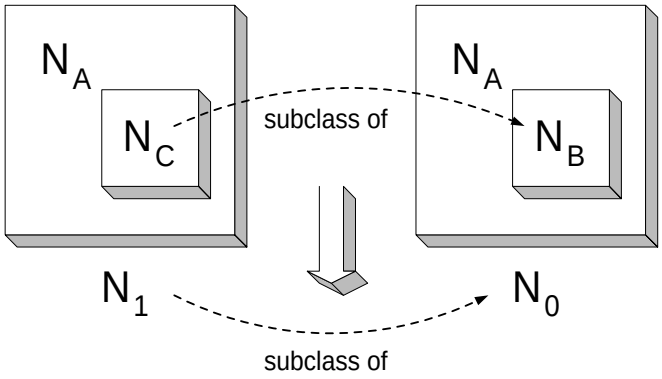

Fig. 18. The essence of Theorem 7: if N_C^W is a subclass of, then N_1 is a subclass of N_0.

add a source and sink place we obtain a sound WF-net N_B^W. In addition, it is assumed that there is no multiple activation. Moreover, there are three additional P/T nets N_C, N_C^W, and N_1. N_1 is composed of N_A and N_C. The connections between N_A and N_C in N_1 are essentially the same as the connections between N_A and N_B in N_0, e.g., $P_A \cap P_C = P_A \cap P_B$ (see Theorem 7 for details). Moreover, N_C is chosen in such a way that if we remove the places $P_A \cap P_C$ and add a source and sink place we obtain a sound WF-net N_C^W *which is a subclass of* N_B^W *under projection inheritance*. Under these conditions N_1 is guaranteed to be sound and a subclass of N_0. In other words: Theorem 7 shows that inheritance is some kind of congruence under the composition of WF-nets.

Theorem 7 (Compositionality of projection inheritance). *Let* $N_0 = (P_0, T_0, M_0, F_0, \ell_0)$, $N_1 = (P_1, T_1, M_1, F_1, \ell_1)$, $N_A = (P_A, T_A, M_A, F_A, \ell_A)$, $N_B = (P_B, T_B, M_B, F_B, \ell_B)$, $N_C = (P_C, T_C, M_C, F_C, \ell_C)$, $N_B^W = (P_B^W, T_B^W, M_B^W, F_B^W, \ell_B^W)$, *and* $N_C^W = (P_C^W, T_C^W, M_C^W, F_C^W, \ell_C^W)$ *be labeled P/T-nets. If*

1. N_0 *is a sound WF-net in* $\mathcal{W}$ *with source place* $i = \underline{source}(N_0)$ *and sink place* $o = \underline{sink}(N_0)$,
2. $N_0 = N_A \cup N_B$ *is well defined,*
3. $N_1 = N_A \cup N_C$ *is well defined,*
4. $T_A \cap T_B = \emptyset$,
5. $T_A \cap T_C = \emptyset$,
6. $P_A \cap P_B = P_A \cap P_C$,
7. N_B^W *is a sound WF-net in* $\mathcal{W}$ *such that* $\underline{strip}(N_B^W) = (P_B \backslash P_A, T_B, M_B, F_B \cap ((P_B^W \times T_B^W) \cup (T_B^W \times P_B^W)), \ell_B)$, $i_B = \underline{source}(N_B^W)$, $o_B = \underline{sink}(N_B^W)$, *and* $\{i_B, o_B\} \cap P_0 = \emptyset$,
8. N_C^W *is a sound WF-net in* $\mathcal{W}$ *such that* $\underline{strip}(N_C^W) = (P_C \backslash P_A, T_C, M_C, F_C \cap ((P_C^W \times T_C^W) \cup (T_C^W \times P_C^W)), \ell_C)$, $i_C = \underline{source}(N_C^W)$, $o_C = \underline{sink}(N_C^W)$, *and* $\{i_C, o_C\} \cap P_1 = \emptyset$,
9. $\{\ell_B(t) \mid t \in \underline{start}(N_B^W)\} = \{\ell_C(t) \mid t \in \underline{start}(N_C^W)\}$, *i.e., the sets of start labels coincide,*

10. $\{\ell_B(t) \mid t \in \underline{stop}(N_B^W)\} = \{\ell_C(t) \mid t \in \underline{stop}(N_C^W)\}$, *i.e., the sets of stop labels coincide,*
11. $(\forall\, t : t \in T_B \wedge \ell_B(t) = \tau : ({}^{N_0}\bullet t \cap P_A = \emptyset) \wedge (t^{N_0}\bullet \cap P_A = \emptyset))$,
12. $(\forall\, t : t \in T_C \wedge \ell_1(t) \notin \alpha(N_B^W) : ({}^{N_1}\bullet t \cap P_A = \emptyset) \wedge (t^{N_1}\bullet \cap P_A = \emptyset))$,
13. $(\forall\, t, t' : t \in T_B \wedge t' \in T_C \wedge \ell_B(t) = \ell_C(t') : ({}^{N_0}\bullet t \cap P_A = {}^{N_1}\bullet t' \cap P_A) \wedge (t^{N_0}\bullet \cap P_A = t'^{N_1}\bullet \cap P_A))$,
14. P_B^W *is activation safe in* $(N_0, [i])$, *and*
15. $N_C^W \leq_{pj} N_B^W$,

then N_1 *is a sound WF-net in* $\mathcal{W}$ *such that* $N_1 \leq_{pj} N_0$.

Proof. The proof consists of three parts. First, we provide some useful observations. Then, we show that there is a branching bisimulation between $(N_0, [i])$ and $\tau_I(N_1, [i])$ ($I = \alpha(N_1) \backslash \alpha(N_0)$). Finally, we show that N_1 is a sound WF-net and conclude that $N_1 \leq_{pj} N_0$ using the branching bisimulation.

Part A
The following observations are useful for the remainder of the proof:

1. Since $N_C^W \leq_{pj} N_B^W$, $\alpha(N_B^W) \subseteq \alpha(N_C^W)$ and there is a branching bisimulation $\mathcal{R}_{BC}$ such that $(N_B^W, [i_B])\ \mathcal{R}_{BC} \tau_I(N_C^W, [i_C])$ with $I = \alpha(N_C^W) \backslash \alpha(N_B^W) = \alpha(N_1) \backslash \alpha(N_0)$. Without loss of generality we assume that $\mathcal{R}_{BC} \subseteq \{((N_B^W, s_B), (N_C^W, s_C)) \mid s_B \in [N_B^W, [i_B]\rangle \wedge s_C \in [N_C^W, [i_C]\rangle\}$.
   $\diamond$ This follows directly from the definition of projection inheritance.
2. $(\forall\, t, t' : t \in T_B \wedge t' \in T_B \wedge \ell_B(t) = \ell_B(t') : ({}^{N_0}\bullet t \cap P_A = {}^{N_0}\bullet t' \cap P_A) \wedge (t^{N_0}\bullet \cap P_A = t'^{N_0}\bullet \cap P_A))$, i.e., transitions in T_B with identical labels have identical effects on the interface $P_A \cap P_B$.
   $\diamond$ If both transitions have a τ label, then there are no connections to the interface $P_A \cap P_B$. If the transitions have a visible label, then there is a corresponding transition in N_C. Since the connections of this transition in N_C to places in $P_A \cap P_B$ are identical to those of t and t', the external connections of t and t' have to match.
3. $(\forall\, t, t' : t \in T_C \wedge t' \in T_C \wedge \ell_C(t) = \ell_C(t') : ({}^{N_1}\bullet t \cap P_A = {}^{N_1}\bullet t' \cap P_A) \wedge (t^{N_1}\bullet \cap P_A = t'^{N_1}\bullet \cap P_A))$, i.e., transitions in T_C with identical labels have identical effects on the interface $P_A \cap P_C$.
   $\diamond$ If both transitions have a τ label or a label not used in N_B, then there are no connections to the interface $P_A \cap P_B$. If the transitions have a visible label used in N_B, then there is a corresponding transition in N_B. Since the connections of this transition in N_B to places in $P_A \cap P_C$ are identical to those of t and t', the external connections of t and t' have to match.
4. For any $t \in \underline{start}(N_B^W)$, there exists a $t' \in \underline{start}(N_C^W)$ such that $\ell_B(t) = \ell_C(t')$ and ${}^{N_0}\bullet t = {}^{N_1}\bullet t'$, and for any $t \in \underline{stop}(N_C^W)$, there exists a $t' \in \underline{stop}(N_B^W)$ such that $\ell_C(t) = \ell_B(t')$ and $t^{N_1}\bullet = t'^{N_0}\bullet$.
   $\diamond$ This follows directly from the requirement that all start and stop labels are visible, the sets of start labels coincide, the sets of stop labels coincide, and transitions

in different nets with identical commonly visible labels have identical sets of input/output places.

5. N_0, N_1, N_B^W, and N_C^W completely determine N_A, N_B, and N_C.
   $\diamond$ $N_A = N_0 \cap N_1$, $N_B = ({}^{N_0}\bullet T_B^W \cup T_B^W {}^{N_0}\bullet, T_B^W, M_B^W, F_0 \cap ((P_B \times T_B) \cup (T_B \times P_B)), \ell_B^W)$, and $N_C = ({}^{N_1}\bullet T_C^W \cup T_C^W {}^{N_1}\bullet, T_C^W, M_C^W, F_1 \cap ((P_C \times T_C) \cup (T_C \times P_C)), \ell_C^W)$.
6. Any marking $s_0 \in [N_0, [i]\rangle$ can be partitioned into s_A and s_B such that $s_0 = s_A + s_B$, $s_A \in \mathcal{B}(P_A)$, $s_B \in \mathcal{B}(P_0 \backslash P_A)$, and $s_B = \mathbf{0}$ or $s_B \in [N_B^W, [i_B]\rangle$.
   $\diamond$ Initially, s_B is empty. (Note that $i \in P_A$.) The only way to mark places in $P_0 \backslash P_A$ is to fire a transition in $\underline{start}(N_B^W)$. However, P_B^W is activation safe. Therefore, the start transitions are blocked until $P_B \backslash P_A = P_0 \backslash P_A$ is empty again and it is not possible to reach states outside $[N_B^W, [i_B]\rangle$.

Part B

Based on $\mathcal{R}_{BC}$ and N_0, N_1, N_B^W, and N_C^W as defined above. We define $\mathcal{R}_{01}$ as follows: $\mathcal{R}_{01} = \{((N_0, s_A + s_B), \tau_I(N_1, s_A + s_C)) \mid s_A \in \mathcal{B}(P_A) \wedge s_B \in \mathcal{B}(P_0 \backslash P_A) \wedge s_C \in \mathcal{B}(P_1 \backslash P_A) \wedge s_A + s_B \in [N_0, [i]\rangle \wedge s_A + s_C \in [N_1, [i]\rangle \wedge ((s_B = \mathbf{0} \wedge s_C = \mathbf{0}) \vee ((N_B^W, s_B) \mathcal{R}_{BC} \tau_I (N_C^W, s_C)))\}$. We show that $\mathcal{R}_{01}$ is a branching bisimulation and that $(N_0, [i]) \mathcal{R}_{01} \tau_I (N_1, [i])$.

Consider two markings $s_0 \in [N_0, [i]\rangle$ and $s_1 \in [N_1, [i]\rangle$ such that $(N_0, s_0) \mathcal{R}_{01}$ $\tau_I(N_1, s_1)$. The bags s_0 and s_1 can be partitioned as in the definition of $\mathcal{R}_{01}$, i.e., $s_0 = s_A + s_B$, $s_1 = s_A + s_C$, $s_A \in \mathcal{B}(P_A)$, $s_B \in \mathcal{B}(P_0 \backslash P_A)$, $s_C \in \mathcal{B}(P_1 \backslash P_A)$. For these two markings we will verify the three requirements stated in the definition of branching bisimilarity.

1. Assume that $t \in T_0$ is such that (N_0, s_0) $[\ell_0(t)\rangle$ (N_0, s_0'). Bag s_0' can be partitioned into s_A' and s_B' as before. We need to prove that there exist s_1', s_1'' such that $(N_1, s_1) \Longrightarrow (N_1, s_1'')$ $[(\ell_0(t))\rangle$ $(N_1, s_1') \wedge (N_0, s_0) \mathcal{R}_{01} (N_1, s_1'') \wedge (N_0, s_0') \mathcal{R}_{01}$ (N_1, s_1').
   - If $t \in T_A$, then t is also enabled in (N_1, s_1) and firing t only affects places in P_A because ${}^{N_0}\bullet t \cup t {}^{N_0}\bullet = {}^{N_1}\bullet t \cup t {}^{N_1}\bullet \subseteq P_A$. Moreover, $\ell_0(t) = \ell_1(t)$. Therefore, $s_1'' = s_1$ and $s_1' = s_A' + s_C$ are such that $(N_1, s_1) \Longrightarrow (N_1, s_1'')$ $[(\ell_0(t))\rangle$ $(N_1, s') \wedge (N_0, s_0) \mathcal{R}_{01} (N_1, s_1'') \wedge (N_0, s_0') \mathcal{R}_{01} (N_1, s_1')$.
   - If $t \notin T_A$, then $t \in T_B$.
     - Assume $t \in \underline{start}(N_B^W)$. Since P_B^W is activation safe in $(N_0, [i])$, $s_B = \mathbf{0}$. Moreover, $s_C = \mathbf{0}$, because $s_B = \mathbf{0}$, $(N_0, s_0) \mathcal{R}_{01} \tau_I (N_1, s_1)$, and there is no s_C such that $(N_B^W, \mathbf{0}) \mathcal{R}_{BC} (\tau_I (N_C^W, s_C))$. Since $t \in \underline{start}(N_B^W)$, each place in ${}^{N_0}\bullet t$ is marked in both s_0 and s_1. Moreover, $\ell_0(t) \neq \tau$. Clearly, there is a $t' \in T_C$ such that $\ell_0(t) = \ell_1(t')$ and ${}^{N_1}\bullet t' = {}^{N_0}\bullet t \subseteq \mathcal{B}(P_A)$. Since s_0 and s_1 are identical with respect to the places in P_A, t' is also enabled in (N_1, s_1). Moreover, the result of firing t' is identical to t with respect to the places in P_A. Let s_C' be such that $(N_C^W, [i_C])$ $[\ell_C(t')\rangle$ (N_C^W, s_C') and $(N_B^W, s_B') \mathcal{R}_{BC} \tau_I (N_C^W, s_C')$. Such a s_C' exists because $(N_B^W,$ $[i_B])$ $\mathcal{R}_{BC}$ $\tau_I (N_C^W, [i_C])$. It is easy to see that $s_1'' = s_1$ and $s_1' = s_A' + s_C'$

are such that $(N_1, s_1) \Longrightarrow (N_1, s_1'')[(\ell_0(t))\rangle(N_1, s_1') \wedge (N_0, s_0)\mathcal{R}_{01}(N_1, s_1'') \wedge (N_0, s_0')\mathcal{R}_{01}(N_1, s_1')$.

- Assume $t \in \underline{stop}(N_B^W)$. Clearly, $s_B \neq \mathbf{0}$. Hence, $(N_B^W, s_B)\ \mathcal{R}_{BC}\ (\tau_I(N_C^W, s_C))$. Transition t is also enabled in (N_B^W, s_B). Since N_B^W is sound, $s_B \in [N_B^W, [i_B]\rangle$, and $t \in \underline{stop}(N_B^W)$, $(N_B^W, s_B)\ [\ell(t)\rangle\ (N_B^W, [o_B])$. Hence, $s_B' = \mathbf{0}$. Since $(N_B^W, s_B)\mathcal{R}_{BC}(\tau_I(N_C^W, s_C))$, it is straightforward to show that in (N_C^W, s_C) a sequence consisting of zero or more silent steps can be executed followed by the firing of a transition t' such that $\ell_0(t) = \ell_1(t')$. Let s_C' be the resulting marking. Since N_C^W is sound, $s_C \in [N_C^W, [i_C]\rangle$, and $t' \in \underline{stop}(N_C^W)$, $s_C' = [o_C]$. Clearly, the same sequence can be executed in (N_1, s_1) leading to s_1'. Note that in s_1' only places in P_A are marked. Since the effects of transitions t in N_0 and t' in N_1 on the places in P_A are identical, $s_1' = s_0'$. Therefore, there are s_1'' and s_1' such that $(N_1, s_1) \Longrightarrow (N_1, s_1'')\ [(\ell_0(t))\rangle\ (N_1, s') \wedge (N_0, s_0)\mathcal{R}_{01}(N_1, s_1'') \wedge (N_0, s_0')\mathcal{R}_{01}(N_1, s_1')$.
- Assume $t \in T_B \backslash (\underline{start}(N_B^W) \cup \underline{stop}(N_B^W))$. Since $s_B \neq \mathbf{0}$, $(N_B^W, s_B)\ \mathcal{R}_{BC}\ (\tau_I(N_C^W, s_C))$.
  - If $\ell_0(t) = \tau$, then choose $s_1' = s_1'' = s_1$. It is easy to see that $(N_1, s_1) \Longrightarrow (N_1, s_1'')\ [(\ell_0(t))\rangle\ (N_1, s') \wedge (N_0, s_0)\mathcal{R}_{01}(N_1, s_1'') \wedge (N_0, s_0')\mathcal{R}_{01}(N_1, s_1')$.
  - If $\ell_0(t) \neq \tau$, then it is straightforward to show that in (N_C^W, s_C) a sequence consisting of zero or more silent steps can be executed followed by the firing of a transition t' such that $\ell_0(t) = \ell_1(t')$. Let s_C' be the resulting marking. Clearly, $(N_B^W, s_B')\mathcal{R}_{BC}(\tau_I(N_C^W, s_C'))$ and $s_C' \in \mathcal{B}(P_1 \backslash P_A)$, i.e., s_C' does not mark o_C. The same sequence can be executed in (N_1, s_1) leading to s_1'. The effect of the execution of t' on the places in P_A is identical to the effect of t on the places in P_A , i.e., $(\forall t, t' : t \in T_B \wedge t' \in T_C \wedge \ell_B(t) = \ell_C(t') : (\ {}^{N_0}\!\bullet t \cap P_A = {}^{N_1}\!\bullet t' \cap P_A) \wedge (t^{N_0}\!\bullet \cap P_A = t'^{N_1}\!\bullet \cap P_A))$. Therefore, there are s_1'' and s_1' such that $(N_1, s_1) \Longrightarrow (N_1, s_1'')\ [(\ell_0(t))\rangle\ (N_1, s') \wedge (N_0, s_0)\mathcal{R}_{01}(N_1, s_1'') \wedge (N_0, s_0')\mathcal{R}_{01}(N_1, s_1')$.

2. Assume that $t \in T_1$ is such that $(N_1, s_1)\ [\ell_1(t)\rangle\ (N_1, s_1')$. We need to prove that there exist s_0', s_0'' such that $(N_0, s_0) \Longrightarrow (N_0, s_0'')\ [(\ell_1(t))\rangle\ (N_0, s_0') \wedge (N_0, s_0'')\mathcal{R}_{01}$ $(N_1, s_1) \wedge (N_0, s_0')\mathcal{R}_{01}(N_1, s_1')$. The proof is almost identical to the proof in the other direction. The only issue which should be noted is that if $t \in \underline{start}(N_C^W)$ is enabled in (N_1, s_1), then $s_C = \mathbf{0}$: Because t is also enabled in (N_0, s_0) and P_B^W is activation safe, $s_B = \mathbf{0}$. Moreover, there is no s_C such that $(N_B^W, \mathbf{0})\ \mathcal{R}_{BC}$ $(\tau_I(N_C^W, s_C))$. Hence, $s_C = \mathbf{0}$.
3. Assume $\downarrow s_0$. We need to prove that $\Downarrow s_1$. $\downarrow s_0$ implies that $s_0 = [o]$, $s_A = [o]$, and $s_B = \mathbf{0}$. If $s_C = \mathbf{0}$, then $s_1 = [o]$ and $\Downarrow s_1$ (in fact $\downarrow s_1$). It is not possible that $s_C \neq \mathbf{0}$, because this would imply that $(N_B^W, \mathbf{0})\mathcal{R}_{BC}\tau_I(N_C^W, s_C)$ which is not possible. Similarly, it can be shown that $\downarrow s_1$ implies $\Downarrow s_0$.

From the definition of $\mathcal{R}_{01}$ it follows that $(N_0, [i])\mathcal{R}_{01}\tau_I(N_1, [i])$.

Part C

Remains to prove that N_1 is a sound WF-net. It is easy to see that N_1 is a WF-net: There is one source place i, one source place o, and every node is on a path from i to o. To prove that N_1 is sound, consider an arbitrary marking $s_1 \in [N_1, [i]\rangle$. For this marking there is a counterpart s_0 in the original net (N_0) such that $s_0 \in [N_0, [i]\rangle$ and $(N_0, s_0)\mathcal{R}_{01}\tau_I(N_1, s_1)$. Using s_0 we verify the four requirements for soundness:

- $(N_1, [i])$ is safe because, for any place $p \in P_A$, $s_1(p) = s_0(p) \leq 1$, and there is a marking $s_C \in [N_C^W, [i_C]\rangle$ such that for any place $p \in P_1 \backslash P_A$: $s_1(p) = s_C(p) \leq 1$.
- Suppose that $o \in s_1$. Since N_0 is sound, $s_0 = [o]$. Since $(N_0, s_0)\mathcal{R}_{01}\ \tau_I(N_1, s_1)$, the other places in P_A are empty. The places in $P_1 \backslash P_A$ are also empty, because otherwise there would be a nonempty bag s_C such that $s_C \neq [o_B]$ and $(N_B^W, \mathbf{0})\mathcal{R}_{BC}\ \tau_I(N_C^W, s_C)$. Clearly this is not possible because from s_C it would be possible to fire a non-τ-labeled transition.
- From s_0 it is possible to reach the marking $[o]$ in N_0 because N_0 is sound. Since $(N_0, s_0) \sim_b \tau_I(N_1, s_1)$ it is possible to do the same in N_1 starting from s_1.
- To prove that there are no dead transitions in $(N_1, [i])$, we first consider transitions in T_A. Suppose a transition $t \in T_A$ is enabled in (N_0, s_0), then t is also enabled in (N_1, s_1). Since there are no dead transitions in $(N_0, [i])$, it is possible to enable any transition $t \in T_A$ starting from $(N_1, [i])$. Transitions in $T_1 \backslash T_A$ are not dead, because there are no dead transitions in $(N_C^W, [i_C])$.

Since N_1 is a sound WF-net and $\mathcal{R}_{01}$ is a branching bisimulation, we conclude that $N_1 \leq_{pj} N_0$. □

By applying Theorem 7 it is possible to prove that $N^{overall}$ obtained using the P2P approach is a sound WF-net and a subclass of N^{publ}.

Theorem 8. *Let D, N^{publ}, $Q^{overall}$, and $N^{overall}$ be as defined in Definition 31.*

1. *$Q^{overall}$ is sound, and*
2. *$N^{overall}$ is a subclass of N^{publ} under projection inheritance (i.e., $N^{overall} \leq_{pj} N^{publ}$).*

Proof. For any $k \in D$: $Q_k = (C, n, N_0^{priv}, \ldots, N_{k-1}^{priv}, N_k^{part}, \ldots, N_{n-1}^{part}, G)$, i.e., Q_k is the IOWF-net where the first k public parts are replaced by private workflows. We use induction in k to prove that Q_k is sound and $\underline{flat}(Q_k)$ is a subclass of N^{publ} for any $k \in D$.

Base case Assume that $k = 0$. $Q_k = Q^{part}$. Q^{part} is a valid partitioning of N^{publ}. Therefore, Q_k is sound and $\underline{flat}(Q_k) = N^{publ}$. Moreover, $\underline{flat}(Q_k)$ is a subclass of N^{publ}.

Inductive step Assume that $k \geq 1$. The induction hypothesis states that Q_{k-1} is sound and $\underline{flat}(Q_{k-1})$ is a subclass of N^{publ}. We need to prove that Q_k is sound and $\underline{flat}(Q_k)$ is a subclass of N^{publ}. $Q_A = (C, n, N_0^{priv}, \ldots, N_{k-2}^{priv}, N_k^{part}, \ldots, N_{n-1}^{part}, G')$ with $G' = G \backslash ((C \times M_{k-1}) \cup (M_{k-1} \times C))$. Clearly, Q_{k-1}, Q_k and Q_A are IOWF-nets. Therefore, we can apply the function $\underline{flat}$ to obtain $N_1 = \underline{flat}(Q_k)$, $N_0 = \underline{flat}(Q_{k-1})$, and $N_A = \underline{flat}(Q_A)$ Let $N_B^W = N_{k-1}^{part}$ and $N_C^W = N_{k-1}^{priv}$.

It is easy to verify that N_0, N_1, N_A, N_B^W, and N_C^W satisfy all the requirements mentioned in Theorem 7. Therefore, Theorem 7 can be used to show that $N_1 = \underline{flat}(Q_k)$ is sound and $N_1 = \underline{flat}(Q_k) \leq_{pj} N_0 = \underline{flat}(Q_{k-1})$. Since projection inheritance is transitive, $\underline{flat}(Q_k) \leq_{pj} \underline{flat}(Q_{k-1})$, and $\underline{flat}(Q_{k-1}) \leq_{pj} N^{publ}$, we conclude that $\underline{flat}(Q_k) \leq_{pj} N^{publ}$. Remains to prove that there is no multiple activation in Q_k. Since $\underline{flat}(Q_k) \leq_{pj} N^{publ}$ and N^{publ} contains all start and stop transitions of the individual subflows, it is not possible to fire a start transition while the places in the subflow are not empty. This can easily be verified by considering the firing sequences in $\underline{flat}(Q_k)$ after abstraction. Q_k is sound because all subflows are sound, $\underline{flat}(Q_k)$ is sound, and there is no multiple activation.

Hence, $Q^{overall} = Q_n$ is sound and $N^{overall} = \underline{flat}(Q_n)$ is a subclass of N^{publ}. □

Theorem 8 clearly shows the value of the P2P approach: Without the need for any coordination among the business parters involved, the resulting interorganizational workflow is guaranteed to be sound. Moreover, it is guaranteed that the resulting interorganizational workflow realizes the public workflow, i.e., the tasks agreed upon in the public workflow are executed in the proper order. Consider for example the public workflow N^{part} shown in Figure 2 and the IOWF-net $Q^{overall}$ described by figures 3, 7, and 9. $Q^{overall}$ can be obtained via the P2P approach since the WF-net shown in Figure 7 is a subclass of the WF-net shown in Figure 4 and the WF-net shown in Figure 9 is a subclass of the WF-net shown in Figure 5. Therefore, it is guaranteed that the overall WF-net shown in Figure 16 is sound and a subclass of the WF-net Figure 2.

6 Local view

In this section we focus on the view on the interorganizational workflow from the perspective of one of the domains. The local view of a domain is a detailed description its own private workflow and a high-level description of the part of the workflow handled by the other domains, i.e., the local view of $k \in D$ is composed of N_0^{part}, N_1^{part}, ..., N_{k-1}^{part}, N_k^{priv}, N_{k+1}^{part}, ..., N_{n-1}^{part}.

Definition 33 (Local view). *Let D, Q^{part}, N_k^{part}, and N_k^{priv}, etc. be as defined in Definition 31. For all $k \in D$: $Q_k^{view} = (C, n, N_0, N_1, \ldots, N_{n-1}, G)$ with $N_k = N_k^{priv}$ and $(\forall\, l : 0 \leq l < n \wedge l \neq k : N_l = N_l^{part})$ is called the* local view. *$N_k^{view} = \underline{flat}(Q_k^{view})$ is the* local view WF-net.

Figure 19 shows the local view WF-net N_0^{view} of the contractor on the interorganizational workflow described by figures 3, 7, and 9. The contractor has a detailed view of its own part of the workflow (left) and a high-level view of the subcontractor's part of the workflow (right).

The following theorem shows that each local view has some desirable properties.

Theorem 9. *Let D, N^{publ}, $N^{overall}$, Q_k^{view}, N_k^{view}, etc. be as defined in definitions 31 and 33. For any $k \in D$:*

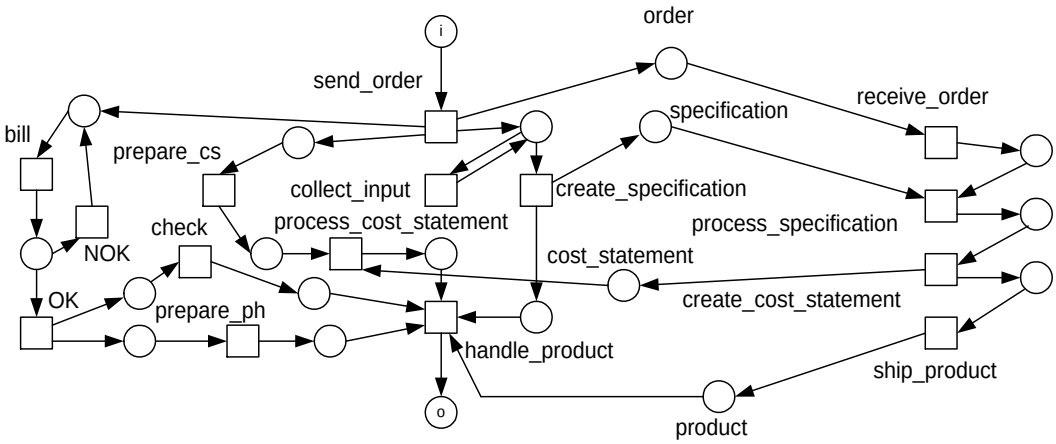

Fig. 19. The local view N_0^{view} of the contractor on the interorganizational workflow composed of the private WF-nets shown in figures 7 and 9.

1. *Q_k^{view} is sound,*
2. *$N^{overall}$ is a subclass of N_k^{view} (i.e., $N^{overall} \leq_{pj} N_k^{view}$), and*
3. *N_k^{view} is a subclass of N^{publ} (i.e., $N_k^{view} \leq_{pj} N^{publ}$).*

Proof. Reorder the domains such that k is the first domain, i.e., $Q^{part} = (C, n, N_k^{part}, N_0^{part}, N_1^{part}, \ldots, N_{k-1}^{part}, N_{k+1}^{part}, \ldots N_{n-1}^{part}, G)$. After reordering use a proof similar to the proof of Theorem 8. Let Q_k be as defined in the proof of Theorem 8 (after reordering). $Q_k^{view} = Q_1$, $N_k^{view} = \underline{flat}(Q_1)$, $N^{publ} = \underline{flat}(Q_0)$, $N^{overall} = \underline{flat}(Q_{n-1})$. Clearly, Q_k^{view} is sound. Moreover, for all $i, j \in D$ with $i \geq j$: $\underline{flat}(Q_i) \leq_{pj} \underline{flat}(Q_j)$. Hence, $N^{overall} \leq_{pj} N_k^{view} \leq_{pj} N^{publ}$. □

Theorem 9 illustrates that the local views generated by the P2P approach are consistent with the public workflow and the overall workflow, i.e., each local view is sound, is a subclass of the public workflow, and a superclass of the overall workflow. Consider for example the interorganizational workflow described by figures 3, 7, and 9. The contractor's local view described by the WF-net shown in Figure 19 is guaranteed to be sound, is a subclass of the the WF-net shown in Figure 2, and is a superclass of the WF-net shown in Figure 16.

If we combine all local views and calculate the GCD, i.e., the part of the workflow all domains agree on, then we obtain the public workflow. If we calculate the LCM, then we obtain the overall workflow.

Theorem 10. *Let D, N^{publ}, $N^{overall}$, N_k^{view}, etc. be as defined in definitions 31 and 33.*

1. *N^{publ} is the greatest common divisor (GCD) of $N_0^{view}, N_1^{view}, \ldots N_{n-1}^{view}$.*
2. *$N^{overall}$ is the least common multiple (LCM) of $N_0^{view}, N_1^{view}, \ldots N_{n-1}^{view}$.*

Proof. To prove that N^{publ} is *the* GCD, we use Theorem 6. We need to prove that (a) $(\forall\, k : 0 \leq k < n : N_k^{view} \leq_{pj} N^{publ})$, and (b) for any sound WF-net N' in $\mathcal{W}$, $(\forall\, k : 0 \leq k < n : N_k^{view} \leq_{pj} N')$ implies $N^{publ} \leq_{pj} N'$. Property (a) follows directly from Theorem 9. To prove (b) take an arbitrary N' in $\mathcal{W}$ such that $(\forall\, k : 0 \leq k < n : N_k^{view} \leq_{pj} N')$. We need to show that $N^{publ} \leq_{pj} N'$. First note that $\alpha(N^{publ}) = (\cap\, k : 0 \leq k < n : \alpha(N_k^{view})) \supseteq \alpha(N')$. Take an arbitrary local view WF-net, e.g., N_0^{view}. Hiding the methods $\alpha(N_0^{view}) \backslash \alpha(N^{publ})$in N_0^{view} yields N^{publ}. Hiding the methods $\alpha(N_0^{view}) \backslash \alpha(N')$ in N_0^{view} yields N'. Since $(\alpha(N_0^{view}) \backslash \alpha(N^{publ})) \subseteq (\alpha(N_0^{view}) \backslash \alpha(N'))$, we obtain N' by abstracting from the methods $\alpha(N^{publ}) \backslash\, \alpha(N')$ in N^{publ}, i.e., $N^{publ} \leq_{pj} N'$ and (b) holds.

The proof that $N^{overall}$ is *the* LCM is similar. A crucial element of this proof is the observation that $\alpha(N^{overall}) = (\cup\, k : 0 \leq k < n : \alpha(N_k^{view})) \subseteq \alpha(N')$. □

Theorems 9 and 10 illustrate the fact that both the workflow all business partners agreed on (N^{publ}) and the actual workflow ($N^{overall}$) are in harmony with the local views. These results demonstrate the sophistication of the P2P approach.

7 Case: An E-bookstore

Throughout this paper we used the rather small example of a contractor and a subcontractor to illustrate the P2P approach. In this section, we will use a larger, more realistic, example. This example is inspired by electronic bookstores such as Amazon [12] and Barnes and Noble [14]. In this section, we design the interorganizational workflow for ordering books. The scope of the workflow process includes the billing and shipping of books. The interorganizational workflow will be partitioned over four domains: (1) the *customer*, (2) the *bookstore* (e.g., Amazon or Barnes and Noble), (3) the *publisher*, and (4) the *shipper*.

7.1 Step 1: Designing the public workflow

Figure 20 shows the public workflow N^{publ}. This workflow is the contract between the customer, the bookstore, the publisher, and the shipper. Clearly there are many customers, publishers, and shippers. Therefore, N^{publ} should be considered as the contract between all customers, publishers, and shippers. However, since we model the processing of an order for a single book, we can assume, without loss of generality, that only one customer, one publisher, and one shipper (if any) are involved. Figure 20 shows that the workflow process is initiated by a customer placing an order (task *place_order*). This customer order is sent to and handled by the bookstore (task *handle_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 (task *place_order*). We will use the term "bookstore order" to refer to the transferred order. The bookstore order is evaluated by the publisher (task *eval_order*) and either accepted (task *b_accept*) or rejected (task *b_reject*). In both cases an appropriate signal is sent to the bookstore. If the bookstore receives a negative answer, it decides (task *decide*) to either search for an alternative publisher (task *alt_publ*) or to reject the customer order (task *c_reject*).

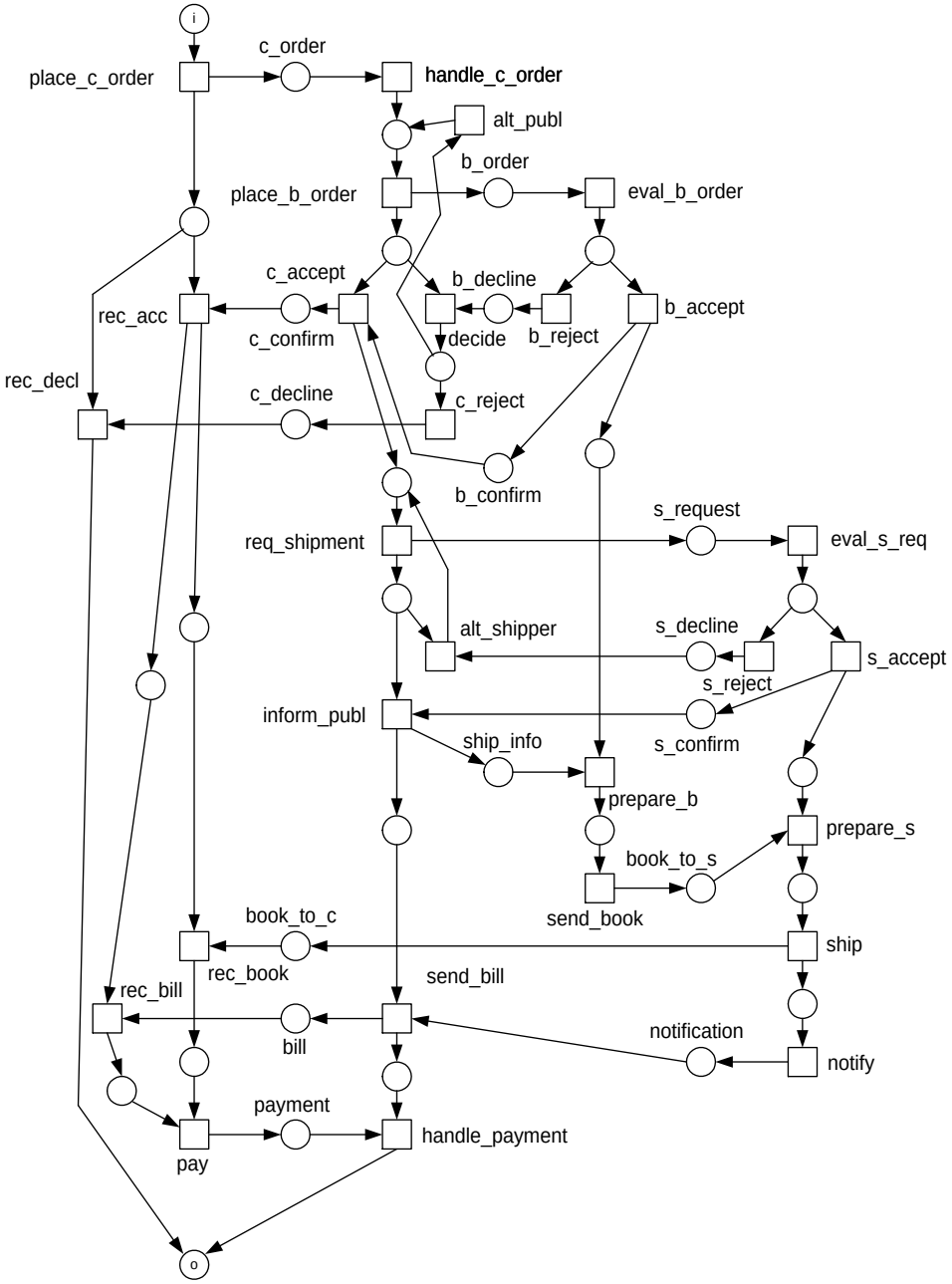

Fig. 20. The public workflow N^{publ}.

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 (task *rec_decl*), then the workflow terminates. If the bookstore receives a positive answer (task *c_accept*), the customer is informed (task *rec_acc*) and the bookstore continues processing the customer order. The bookstore sends a request to a shipper (task *req_shipment*), the shipper evaluates the request (task *eval_s_req*) and either accepts (task *s_accept*) or rejects (task *b_reject*) the 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 (task *inform_publ*), the publisher prepares the book for shipment (task *prepare_b*), and the book is sent from the publisher to the shipper (task *send_book*). The shipper prepares the shipment to the customer (task *prepare_s*) and actually ships the book to the customer (task *ship*). The customer receives the book (task *rec_book*) and the shipper notifies the bookstore (task *notify*). The bookstore sends the bill to the customer (task *send_bill*). After receiving both the book and the bill (task *rec_bill*), the customer makes a payment (task *pay*). Then the bookstore processes the payment (task *handle_payment*) and the interorganizational workflow terminates. It is easy to check that the public workflow shown in Figure 20 is indeed a sound WF-net.

7.2 Step 2: Partitioning the public workflow

The public workflow is partitioned over four domains: (1) the customer domain, (2) the bookstore domain, (3) the publisher domain, and (4) the shipper domain. Based on the description it is clear how the workflow needs to be partitioned. Figures 21, 22, 23, 24, and 25 show the partitioned workflow. Note that the partitioning is valid, i.e., all public parts (i.e., local fragments of the workflow) are sound, there is no multiple activation, and the flattened IOWF-net equals the public workflow.

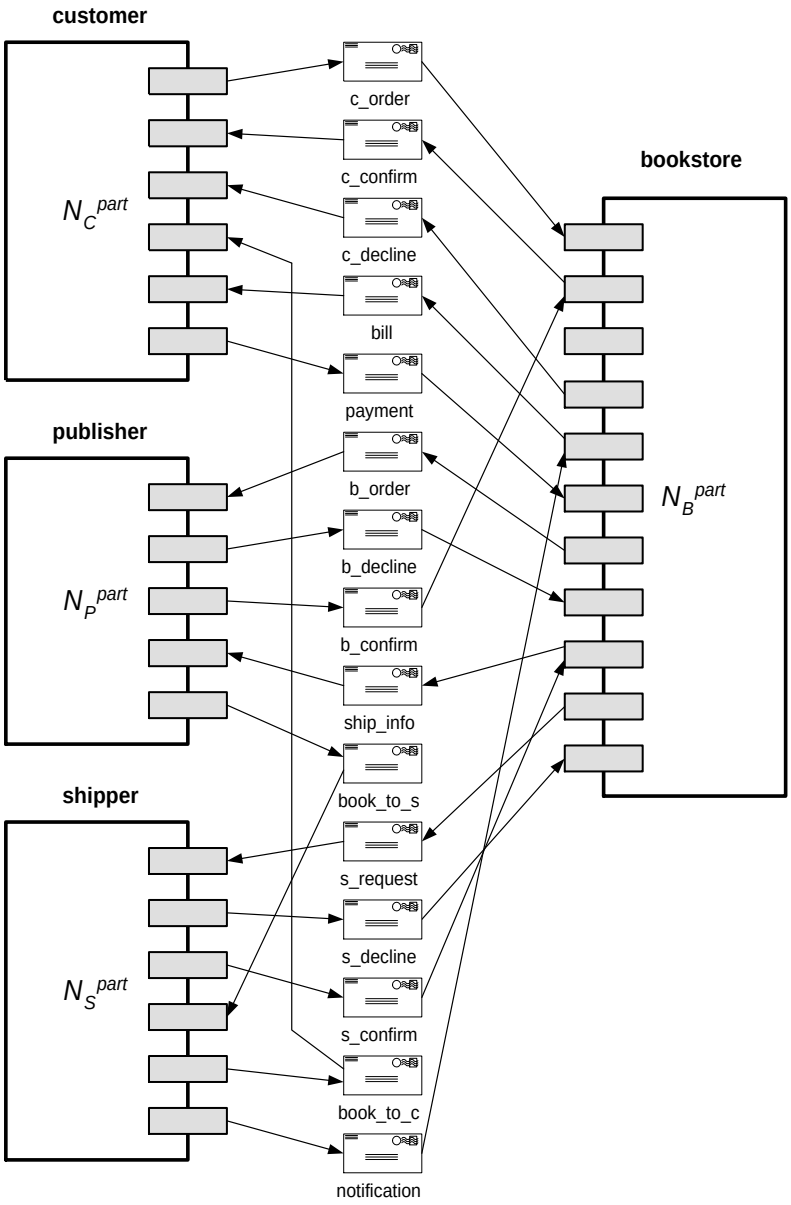

Fig. 21. The interorganizational workflow Q^{part}.

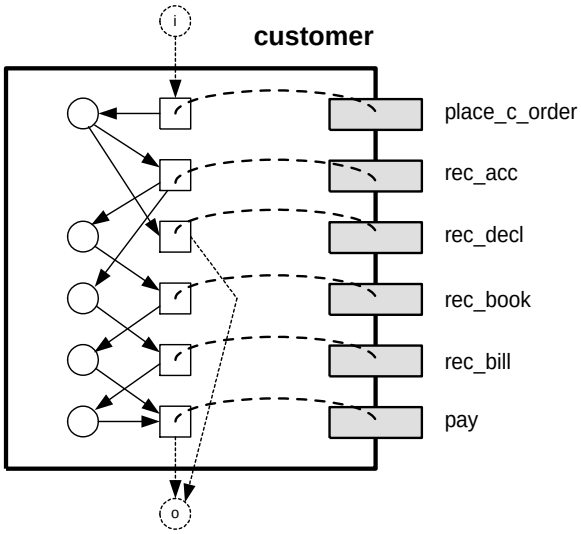

Fig. 22. The WF-net N_C^{part} (public part of the customer domain).

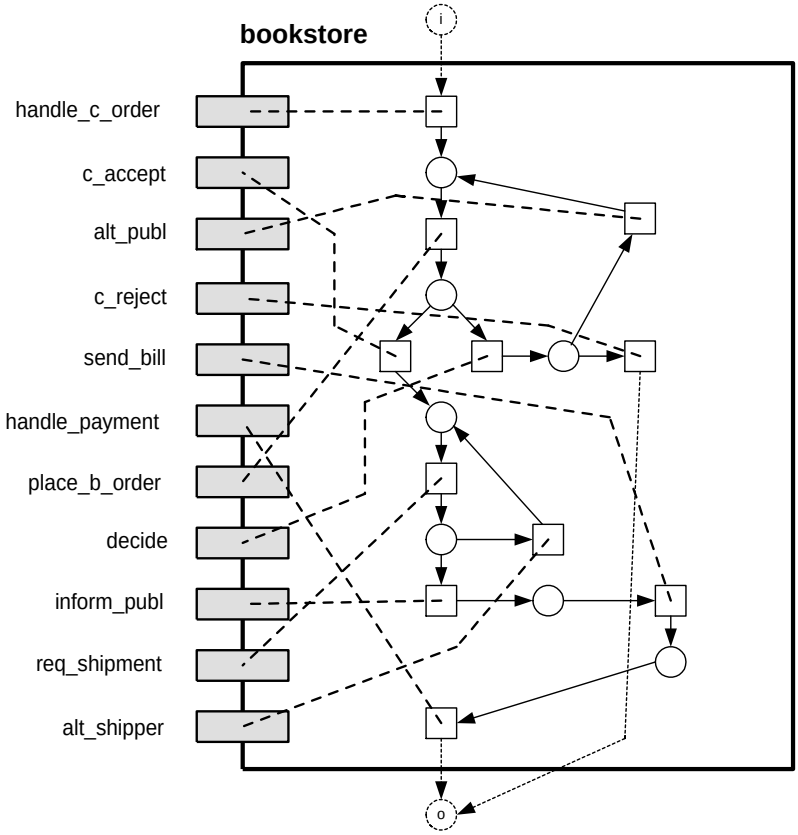

Fig. 23. The WF-net N_B^{part} (public part of the bookstore domain).

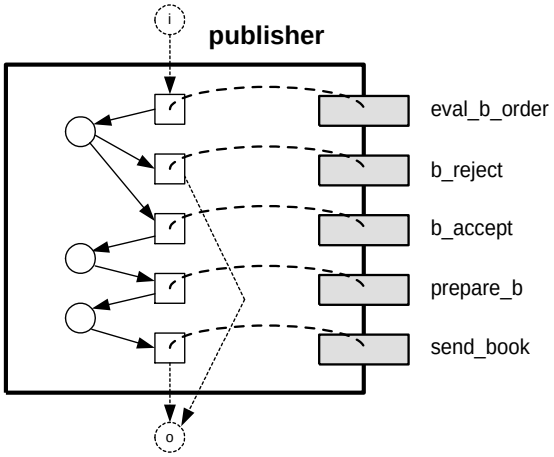

Fig. 24. The WF-net N_P^{part} (public part of the publisher domain).

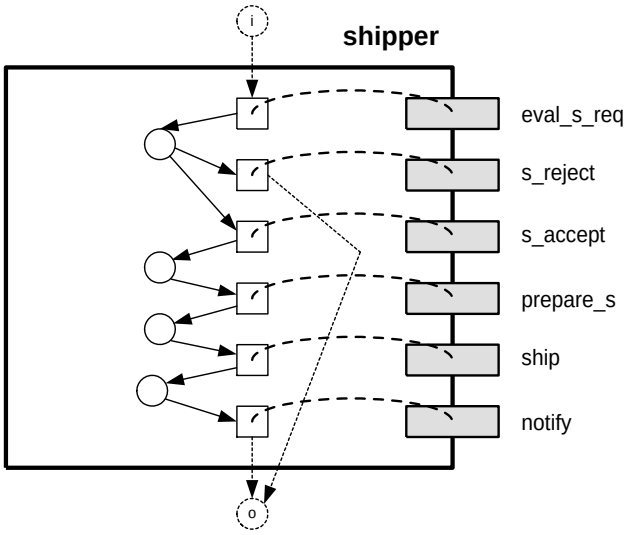

Fig. 25. The WF-net N_S^{part} (public part of the shipper domain).

7.3 Step 3: Designing the private workflows

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 the public part. The private workflow of the customer, bookstore, publisher, and shipper may be much more complex as indicated by figures 22, 23, 24, and 25. Moreover, a private workflow in one domain can be changed dynamically as long as the resulting workflow is a subclass of the public part.

Let us assume that the private workflow of the customer coincides with the public part shown in Figure 22, i.e., $N_C^{priv} = N_C^{part}$. We could have created a much more detailed model of the customer. However, we selected the simplest realization possible because we are more interested in the other three domains.

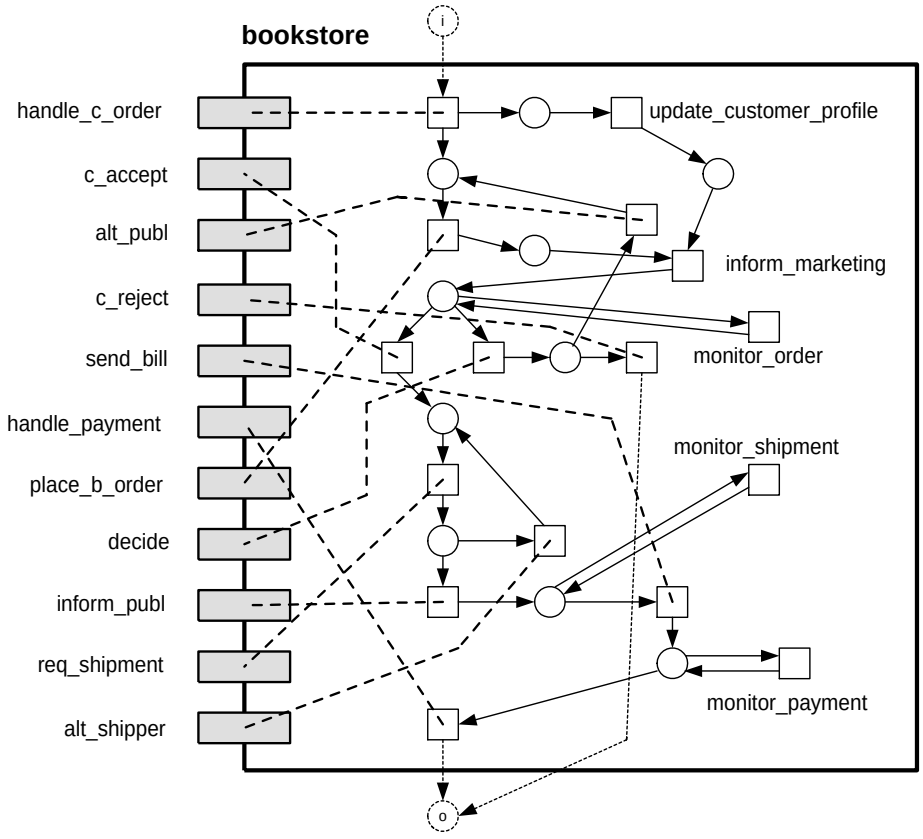

Fig. 26. The WF-net N_B^{priv} (private workflow of the bookstore domain).

Figure 26 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

(task *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 (task *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. The private workflow N_B^{priv} (Figure 26) is a subclass of N_B^{part} (Figure 23) *by construction*: tasks *monitor_order*, *monitor_shipment*, and *monitor_payment* can be added by applying transformation rule PPS three times, task *inform_marketing* can be added using transformation rule PJS, and task *update_customer_profile* can be added using transformation rule $PJ3S$.

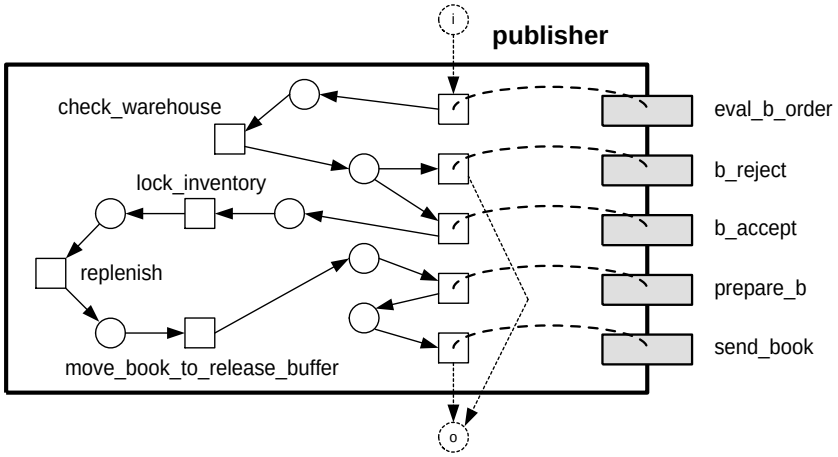

Fig. 27. The WF-net N_P^{priv} (private workflow of the publisher domain).

The private workflow of the publisher, shown in Figure 27, has been extended using transformation rule PJS. Task *check_warehouse* has been added in-between the receipt of the order and the decision to accept and reject. (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 (task *lock_inventory*), the stock is replenished (if possible) (task *replenish*), and the book is moved to the part of the warehouse reserved for books which are waiting for shipment (task *move_book_to_release_buffer*). It is easy to verify that the private workflow N_P^{priv} (Figure 27) is a subclass of N_P^{part} (Figure 24). Note that we just used the transformation rule PJS.

Figure 28 shows the private workflow of the shipper. Six new tasks, i.e., tasks not present in the public workflow, have been added. Task *check_availability_trucks* is ex-

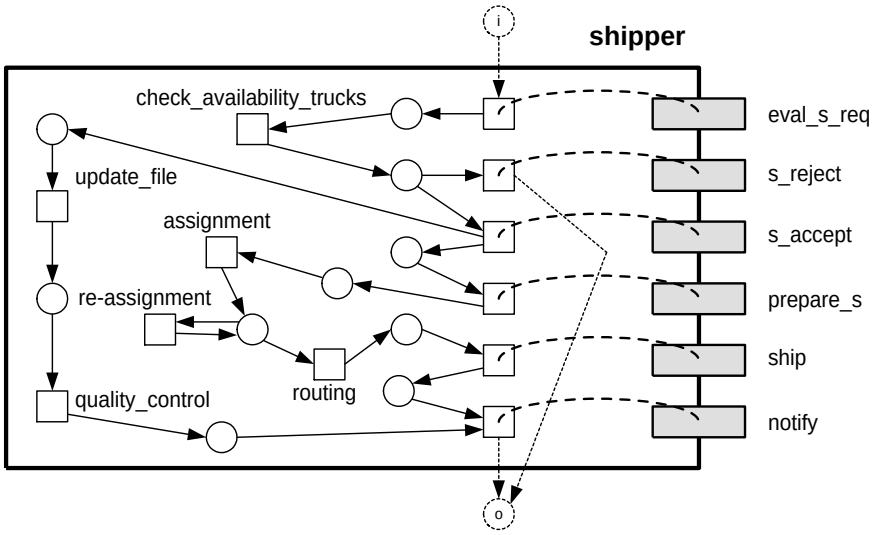

Fig. 28. The WF-net N_S^{priv} (private workflow of the shipper domain).

ecuted 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 parallel with the preparation and shipment tasks. After preparation, shipments are assigned to trucks (task *assignment*). Based on the assignment, the routing of the truck is determined (task *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 Figure 28 is indeed a subclass of Figure 25. 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.

Each of the private workflows is a subclass of the corresponding public part of the public workflow. Therefore, it is guaranteed that the overall workflow is sound and a subclass of the public workflow. Also the local views (i.e., the part of the overall workflow visible from the perspective of one domain) are consistent with both the public workflow and the overall workflow. It is important to note that the P2P approach is *constructive*: By applying the three transformation rules introduced in Section 2.4, 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.

The design of the interorganizational workflow involving a customer, bookstore, publisher, and shipper presented in this section 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. However, even for the simple process described in this section one has to be careful not to make any design errors such as the ones described in the introduction (e.g., a deadlock as a result of reversing the order

of two tasks). 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.

8 Related work and future extensions

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 [22, 23]. For the reader interested in the application of Petri nets to workflow management, we refer to the two recent workshops on workflow management held in conjunction with the annual International Conference on Application and Theory of Petri Nets [10, 36] and an elaborate paper on workflow modeling using Petri nets [3].

Only a few papers in the literature focus on the verification of workflow process definitions. In [25] 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 [11] 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 (see Section 2.2). The technique presented in [11] 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 [42] 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. Based on this reduction technique the analysis tool *FlowMake* [41] has been developed. Flowmake can interface with the IBM MQSeries Workflow product. Some researchers worked on the compositional verification of workflows [2, 15, 50] using well-known Petri-net results such as the refinement rules in [48].

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 [4, 29]. In [4] 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 [29]. 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 [4, 29] 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 presented in Section 2.4.

In the last decade several researchers [13, 27, 28, 33] explored notions of behavioral inheritance (also named subtyping or substitutability). 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 [15]. 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 [7, 15] and *transfer rules* to *migrate* instances from a superclass to a subclass and vice versa [8]. These features are very relevant for a both constructive and robust approach towards interorganizational workflows.

In this paper, we did not address implementation issues. Most of today's commercial workflow systems use a centralized enactment service. Therefore, many of the research prototypes such as MENTOR (University of Saarland at Saarbrucken), METEOR (University of Georgia), MOBILE (University of Erlangen), Panta Rhei (University of Klagenfurt), and WASA (University of Muenster) focus on distribution aspects. These systems typically provide for message passing. Therefore, they can be used to support the P2P approach. A more detailed discussion on the architecture of an enactment service to take care of interorganizational workflows is outside the scope of this paper. The focus of this paper is on the *design* and *analysis* of interorganizational workflows.

We have developed a tool named *Woflan* (WOrkFLow ANalyzer, [3, 9, 49]). 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 WFMS-independent analysis tool. In principle it can interface with many workflow management systems. At the moment, Woflan can interface with the WFMS COSA (Software Ley [45]), the WFMS METEOR (LSDIS [43]), the WFMS Staffware (Staffware [46]), and the BPR-tool Protos (Pallas Athena [38]). Woflan has *not* been designed to analyze interorganizational workflows. However, Woflan can be used to verify the soundness property used throughout this paper. Moreover, Woflan can also check whether one workflow (i.e., WF-net) is a subclass of another workflow. One of the key features of Woflan is ability to guide the user to the source of a design error, i.e., Woflan supplies many context-sensitive diagnostics which support the user in correcting design flaws. Although Woflan has not specifically been designed to verify the correctness of interorganizational workflows, it can support some of the crucial steps in the P2P approach, e.g., Woflan can verify the correctness of the public workflow, and, for each domain, Woflan can be used to check whether the private workflow is a subclass of the corresponding public part under projection inheritance. Figure 29 shows a screenshot of Woflan while analyzing the public workflow of the book ordering process shown in Figure 20 specified using COSA.

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 [8] can be used to migrate cases (i.e., 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. As long as the superclass/subclass relationships are established, it is possible to migrate cases without jeopardizing the correctness of both the local and overall workflow. Second, we want to tackle a topic we did not address in this paper: reconfiguration of interorganizational workflows. In this paper, we assumed the public workflow and the partitioning of the public workflow over the domains to be fixed. In real applications,

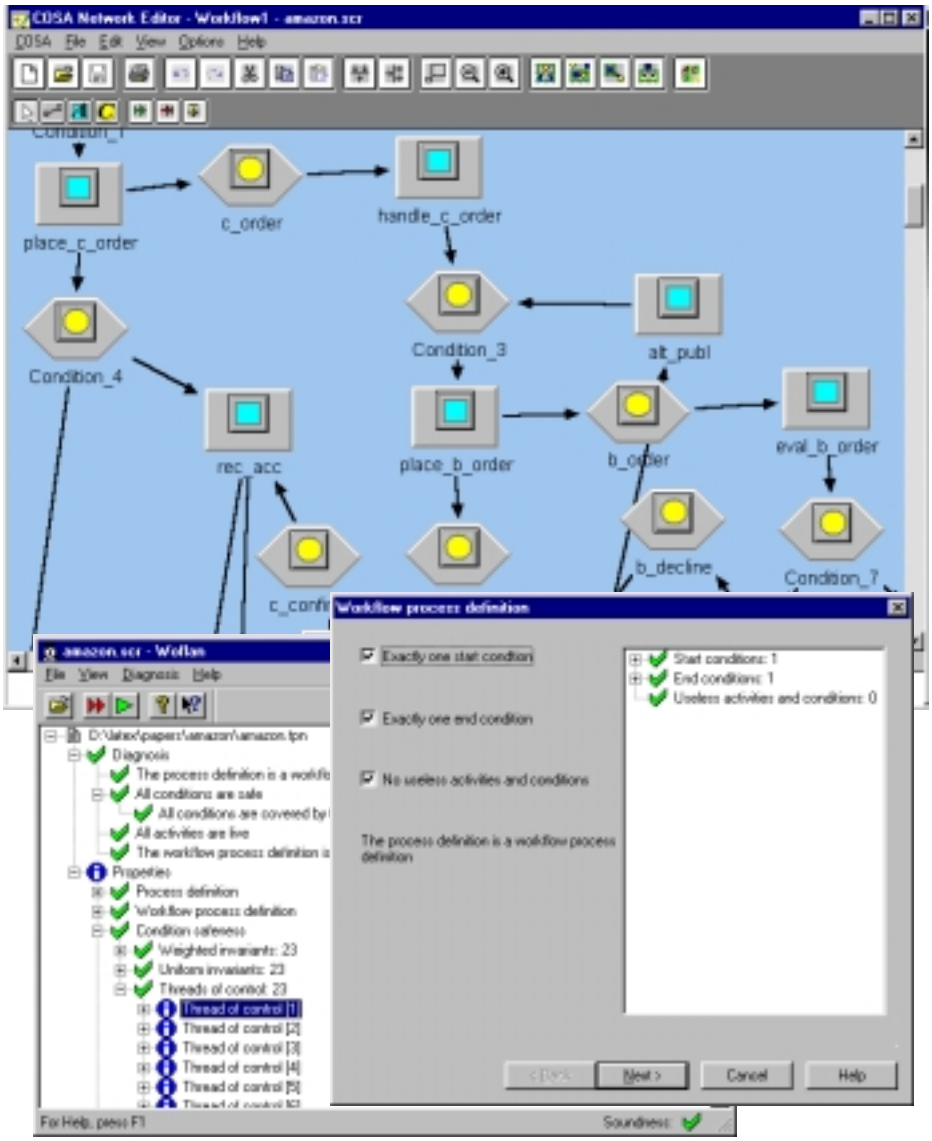

Fig. 29. A screenshot showing COSA (background) and Woflan (foreground): The public workflow shown in Figure 20 is implemented using COSA and verified using Woflan.

tasks are moved from one organization to another and the "contract" (i.e., the public workflow) is changed on a regular basis. A preliminary exploration of these problems shows that the P2P approach can be extended to address these reconfiguration issues. Third, we want to evaluate the P2P approach using a real implementation. For example, we could use the workflow management system METEOR [43] to enact some of the interorganizational workflows designed using the P2P approach. The METEOR system is entirely based on CORBA to provide a platform independent and reliable environment. It also supports interoperability mechanisms like SWAP and JFLOW. Moreover, the METEOR3 model introduces the notion of *foreign* task vs. *native* tasks. A foreign task refers to a task whose realization (implementation) is unknown to workflow designer, whereas the implementation details are known to the workflow designer for a native task. Another important feature for interorganizational workflows are *channels* (also called sink nodes) that are used to specify communication or synchronization between two independent workflows. Some preliminary work using METEOR and a predecessor of the P2P approach was already presented in [6]. Fourth, we would like to experiment with other notions of inheritance. This paper deploys only the notion of projection inheritance. In [7, 8, 15] we defined three other notions of inheritance. These notions seem to be less suitable for interorganizational workflows. Nevertheless, we would like to try to generalize some of the results using a weaker notion of inheritance. Finally, we plan to extend Woflan to offer more support for interorganizational workflows.

Acknowledgements The author would like to thank Twan Basten for his excellent work on inheritance of dynamic behavior, Eric Verbeek for the development of Woflan, a verification tool which can be used to analyze many of the properties defined in this paper, and Kemafor Anyanwu for applying the P2P approach to a telecommunications case.

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. Finding Errors in the Design of a Workflow Process: A Petri-net-based Approach. In W.M.P. van der Aalst, G. De Michelis, and C.A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, volume 98/7 of *Computing Science Reports*, pages 60–81, Lisbon, Portugal, 1998. Eindhoven University of Technology, Eindhoven.
3. 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.
4. 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.
5. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.

6. 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.
7. W.M.P. van der Aalst and T. Basten. Life-cycle Inheritance: A Petri-net-based Approach. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 62–81. Springer-Verlag, Berlin, 1997.
8. W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An approach to tackling problems related to change. Computing Science Reports 99/06, Eindhoven University of Technology, Eindhoven, 1999.
9. W.M.P. van der Aalst, D. Hauschildt, and H.M.W. Verbeek. A Petri-net-based Tool to Analyze Workflows. In B. Farwer, D. Moldt, and M.O. Stehr, editors, *Proceedings of Petri Nets in System Engineering (PNSE'97)*, pages 78–90, Hamburg, Germany, September 1997. University of Hamburg (FBI-HH-B-205/97).
10. W.M.P. van der Aalst, G. De Michelis, and C.A. Ellis, editors. *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, Lisbon, Portugal, June 1998. UNINOVA, Lisbon.
11. 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.
12. Amazon.com, Inc. Amazon.com. http://www.amazon.com, 1999.
13. P. America. Designing an Object-Oriented Programming Language with Behavioral Subtyping. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Foundation of Object-Oriented Languages*, volume 489 of *Lecture Notes in Computer Science*, pages 60–90. Springer-Verlag, Berlin, 1991.
14. Barnes and Noble. bn.com. http://www.bn.com, 1999.
15. 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.
16. R. Benjamin and R. Wigand. Electronic markets and virtual value chains on the information superhighway. *Sloan Management Review*, pages 62–72, 1995.
17. G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, Berlin, 1986.
18. G. Berthelot. Transformations and Decompositions of Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 360–376. Springer-Verlag, Berlin, 1987.
19. 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.
20. J.M. Colom and M. Silva. Improving the Linearly Based Characterization of P/T Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 113–146. Springer-Verlag, Berlin, 1990.
21. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
22. 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.
23. C.A. Ellis and G.J. Nutt. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

24. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.
25. 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.
26. R. Kalakota and A.B. Whinston. *Frontiers of Electronic Commerce*. Addison-Wesley, Reading, Massachusetts, 1996.
27. H. Kilov and W. Harvey, editors. *Object-Oriented Behavioral Specifications*, volume 371 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, USA, 1996.
28. H. Kilov, B. Rumpe, and I. Simmonds, editors. *Behavioral Specifications of Businesses and Systems*, volume 523 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, USA, 1999.
29. 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.
30. A. Kumar and J.L. Zhao. Workflow Support for Electronic Commerce Applications. http://spot.colorado.edu/˜akhil/, 1999.
31. R.M. Lee. Distributed Electronic Trade Scenarios: Representation, Design, Prototyping. *International Journal of Electronic Commerce*, 3(2):105–120, 1999.
32. R.M. Lee and R.W.H. Bons. Soft-Coded Trade Procedures for Open-edi. *International Journal of Electronic Commerce*, 1(1):27–49, 1996.
33. B. Liskov and J. Wing. A Behavioral Notion of Subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6):1811–1841, November 1994.
34. 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.
35. 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.
36. G. De Michelis, C. Ellis, and G. Memmi, editors. *Proceedings of the Second Workshop on Computer-Supported Cooperative Work, Petri nets and Related Formalisms*, Zaragoza, Spain, June 1994.
37. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
38. Pallas Athena. *Protos User Manual*. Pallas Athena BV, Plasmolen, The Netherlands, 1999.
39. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
40. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets II: Applications*, volume 1492 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
41. W. Sadiq and M.E. Orlowska. FlowMake Product Information, Distributed Systems Technology Centre, Queensland, Australia. http://www.dstc.edu.au/Research/Projects/FlowMake/productinfo/index.html.
42. 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.
43. 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.

44. A.P. Sheth, W.M.P. van der Aalst, and I.B. Arpinar. Processes Driving the Networked Economy: ProcessPortals, ProcessVortex, and Dynamically Trading Processes. *IEEE Concurrency*, 7(3):18–31, 1999.
45. Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, Germany, 1998.
46. Staffware. *Staffware 2000 / GWD User Manual*. Staffware plc, Berkshire, United Kingdom, 1999.
47. The White House. A Framework for Global Electronic Commerce. http://www.ecommerce.gov/framewrk.htm, 1997.
48. R. Valette. Analysis of Petri Nets by Stepwise Refinements. *Journal of Computer and System Sciences*, 18:35–46, 1979.
49. 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.
50. M. Voorhoeve. Compositional Modeling and Verification of Workflow Processes. In *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 184–200. Springer-Verlag, Berlin, 2000.
51. V. Zwass. Electronic commerce: structures and issues . *International Journal of Electronic Commerce*, 1(1):3–23, 1996.