Interorganizational Workflows

An approach based on Message Sequence Charts and Petri Nets

W.M.P. van der Aalst[1]
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, telephone: -31 40 2474295, fax: -31 40 2463992, e-mail: wsinwa@win.tue.nl

To date, workflow management focuses on improving the effectivity and efficiency of business processes within one organization. However, today's corporations are challenged to cross organizational boundaries. Phenomena such as electronic commerce, extended enterprises and the Internet stimulate the cooperation between organizations. Therefore, it is interesting to consider workflows distributed over a number of organizations. Interorganizational workflow offers companies the opportunity to re-shape business processes beyond the boundaries of individual organizations. In this paper, we use message sequence charts to specify the interaction between organizations. Petri nets are used to model the workflows inside each organization. Two challenging problems related to interorganizational workflow are tackled in this paper: (1) What are the minimal requirements any interorganizational workflow should satisfy?, and (2) How to decide whether an interorganizational workflow (modeled in terms of Petri nets) is consistent with the interaction structure specified in terms of a message sequence chart?

Keywords: interorganizational workflow; Petri nets; message sequence charts; workflow management; electronic commerce, analysis of workflows.

1 Introduction

Workflow management promises a new solution to an age-old problem: controlling, monitoring, optimizing and supporting business processes. What is new about workflow management is the explicit representation of the business process logic which allows for computerized support. At the moment, there are more than 200 workflow products commercially available and many organizations are introducing workflow technology to support their business processes. Clearly, workflow management is becoming a mature technology which can be applied *within* organizations. However, the number of business processes where multiple organizations are involved is increasing rapidly. Technologies such as Electronic Data Interchange (EDI), the Internet, and the World Wide Web (WWW) enable multiple organizations to participate in shared business processes. The rise of electronic commerce, virtual organizations and extended enterprises highlights the fact that more and more business processes are crossing organizational boundaries ([22]). This means that workflow management should be able to deal with workflow processes which

[1] Part of this work was done at AIFB (University of Karlsruhe, Germany) during a sabbatical leave.

span across multiple organizations.

This paper focuses on interorganizational workflows, i.e., several business partners are involved in shared workflow processes. In this paper, we restrict ourselves to structured processes with a predefined set of tasks and routing constructs. In many cases, where the coordination structure and the interaction between the business partners is not specified explicitly, this is not a realistic assumption ([26]). Nevertheless, there are numerous situations where the organizations participating in a shared workflow processes feel the need to specify the coordination structure explicitly.

This paper focuses on loosely coupled workflow processes. Each business partner has a private workflow process which is connected to the workflow processes of some of the other partners. Two communication mechanisms are used to interact: (i) *asynchronous communication* and (ii) *synchronous communication*. *Message Sequence Charts* (MSC) (cf. [9, 21, 25, 29, 17]) are used to specify the communication between business partners. Loosely coupled workflow processes operate essentially independently, but have to synchronize at certain points to ensure the correct execution of the overall business process. Synchronization of parallel processes is known to be potential source of errors (e.g. deadlock and livelocks). Therefore, it is difficult to establish the correctness for complex interorganizational workflows. This paper uses the notion of correctness introduced in [4]. This notion is referred to as *IO-soundness* and corresponds to the minimal requirements any interorganizational workflow should satisfy. Based on this notion, we present an analysis technique to verify the correctness of an interorganizational workflow. Moreover, we show how to check whether the interorganizational workflow is consistent with respect to the communication structure (i.e. the protocol) specified in terms of message sequence charts.

Because processes are a dominant factor in workflow management, it is important to use an established framework for modeling and analyzing workflow processes [20, 23, 24]. In this paper, we use a framework based on *Petri nets*. Petri nets are a well-founded process modeling technique. The classical Petri net was invented by Carl Adam Petri in the sixties. Since then Petri nets have been used to model and analyze all kinds of processes with applications ranging from protocols, hardware, and embedded systems to flexible manufacturing systems, user interaction, and business processes. There are several reasons for using Petri nets for workflow modeling: their formal semantics, graphical nature, expressiveness, analysis techniques and tools provide a framework for modeling and analyzing workflow processes ([3, 5, 6, 12, 13, 27, 31]).

The remainder of this paper is organized as follows. First, we motivate the focus of this paper on loosely coupled workflows. Then, we will show how Petri nets can be applied to the domain of interorganizational workflows. For this purpose we introduce some Petri-net terminology and some results for the analysis of a single workflow within an organization. Then, we will show that the correctness of interorganizational workflows can be verified using standard Petri net techniques. In the second part of this paper we explore the prob-

lem of checking whether the communication structure between loosely coupled workflow processes is consistent with respect to the communication structure specified in terms of message sequence charts. For the situation where there is just one interaction pattern we define the notion of *1-consistency*. We will show that 1-consistency coincides with the presence of so-called *implicit places*.

2 Interoperability between workflows

As indicated in the introduction, this paper focuses on interorganizational workflows, i.e., several business partners are involved in shared workflow processes. From a technical point of view these issues are also of concern to the Workflow Management Coalition ([24]). The WfMC is working on standards to enable workflow interoperability. These interoperability standards address the technical issues but not the content of the coordination structure. It is our belief that the semantics of the constructs needed to model interorganizational workflows should be defined before solving the technical issues (mainly syntactical). However, before we introduce Petri nets for modeling interorganizational workflows, we focus on the various forms of interoperability.

- *Capacity sharing*
  The first form of interoperability is capacity sharing. This form of interoperability assumes centralized control, i.e., the routing of the workflow is under the control of one workflow manager. The execution of tasks is distributed, i.e., resources of several business partners execute tasks under control of one central workflow engine.

- *Chained execution*
  Chained execution is the second form of interoperability. If chained execution is used, then the workflow process is split into a number of disjunct subprocesses which are executed by different business partners in a sequential order. This form of interoperability requires that a partner transfers or initiates the flow for a case after completing all the work. In contrast to capacity sharing, the control of the workflow is distributed over the business partners.

- *Subcontracting*
  The third form of routing is subcontracting. There is one business partner which subcontracts subprocesses to other business partners. Consider for example company X which sells electronic equipment (make to order). In de business process of the company two of the subprocesses (e.g. production and transport) are subcontracted. For the top-level business partner (i.e. company X) the two subcontracted subprocesses appear to be atomic. For the two business partners executing subcontracted work, the subprocesses can be very complex. Note that the control is hierarchical, i.e., there is one top-level actor and the control is distributed in a tree-like fashion.

- *Case transfer*
  The fourth form of interoperability is case transfer. Each business partner has a copy

of the workflow process description, i.e., the process specification is distributed. However, each case resides at any time at exactly one location. Cases (i.e. process instances) can be transferred from one location to another. A case can be transferred to balance the workload or because tasks are not implemented at all locations.

- *Extended case transfer*
  In the previous form of interoperability it was assumed that each of the business partners uses the same process definition. However, it is possible to allow local variations, e.g., at a specific location the process may be extended with additional tasks. It is important that the extensions allow for the proper transfer of cases. This means that the extensions are executed before transferring the case or that there is some notion of inheritance which allows for the mapping of the state of a case during the transfer.

- *Loosely coupled*
  We use the term loosely coupled to denote the last form of interoperability. For this form of interoperability the process is cut in pieces which may be active in parallel. Moreover, the definition of each of the subprocesses is local, i.e., the environment does not know the internal process, only the protocol which is used to communicate is made public. Each of the business partners may change its workflow process without informing the others as long as the protocol is not affected.

Note that capacity sharing uses centralized control. The other forms of interoperability use a decentralized control. However, note that in case of subcontracting and (extended) case transfer part of the control is (can be) centralized. Chained execution, subcontracting, and loosely coupled use a horizontal partitioning of the workflow, i.e., the process is cut into pieces. (Extended) case transfer uses a vertical partitioning of the flow, i.e., the cases are distributed over the business partners.

This paper focuses on loosely coupled workflow processes. Each business partner has a private workflow process which is connected to the workflow processes of some of the other partners. The communication mechanisms that are used for interaction are synchronous and asynchronous communication. Loosely coupled workflow processes operate essentially independently, but have to synchronize at certain points to ensure the correct execution of the overall business process.

3 Petri nets

This section introduces the basic Petri net terminology and notations. Readers familiar with Petri nets can skip this section. For a review of the history of Petri nets and an extensive bibliography the reader is referred to Murata [28].
The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles.

Definition 1 (Petri net) *A Petri net is a triple* (P, T, F)*:*

- P *is a finite set of places,*
- T *is a finite set of transitions* ($P \cap T = \emptyset$)*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation)*

A place p is called an *input place* of a transition t iff there exists a directed arc from p to t. Place p is called an *output place* of transition t iff there exists a directed arc from t to p. We use $\bullet t$ to denote the set of input places for a transition t. The notations $t\bullet$, $\bullet p$ and $p\bullet$ have similar meanings, e.g. $p\bullet$ is the set of transitions sharing p as an input place. Note that we restrict ourselves to arcs with weight 1. In the context of workflow procedures it makes no sense to have other weights, because places correspond to conditions.

At any time a place contains zero of more *tokens*, drawn as black dots. The *state* M, often referred to as marking, is the distribution of tokens over places, i.e., $M \in P \rightarrow \mathbb{N}$. We will represent a state as follows: $1p_1 + 2p_2 + 1p_3 + 0p_4$ is the state with one token in place p_1, two tokens in p_2, one token in p_3 and no tokens in p_4. We can also represent this state as follows: $p_1 + 2p_2 + p_3$. To compare states, we define a partial ordering. For any two states M_1 and M_2, $M_1 \leq M_2$ iff for all $p \in P$: $M_1(p) \leq M_2(p)$

The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: they change the state of the net according to the following *firing rule*:

(1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.

(2) An enabled transition may *fire*. If transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t.

Given a Petri net $PN = (P, T, F)$ and a state M_1, we have the following notations:

- $M_1[t\rangle_{PN} M_2$: transition t is enabled in state M_1 and firing t in M_1 results in state M_2
- $M_1[\,\rangle_{PN} M_2$: there is a transition t such that $M_1[t\rangle_{PN} M_2$
- $M_1[\sigma\rangle_{PN} M_n$: the firing sequence $\sigma = t_1 t_2 t_3 \ldots t_{n-1} \in T^*$ leads from state M_1 to state M_n, i.e., $M_1[t_1\rangle_{PN} M_2[t_2\rangle_{PN} \ldots [t_{n-1}\rangle_{PN} M_n$

A state M_n is called *reachable* from M_1 (notation $M_1[*\rangle_{PN} M_n$) iff there is a firing sequence $\sigma = t_1 t_2 \ldots t_{n-1}$ such that $M_1[\sigma\rangle_{PN} M_n$. The subscript $_{PN}$ is omitted if it is clear which Petri net is considered. Note that the empty firing sequence is also allowed, i.e., $M_1[*\rangle M_1$.

We use (PN, M) to denote a Petri net PN with an initial state M. A state M' is a *reachable state* of (PN, M) iff $M[*\rangle M'$. Let us define some properties of Petri nets.

Definition 2 (Live) *A Petri net (PN, M) is live iff, for every reachable state M' and every transition t there is a state M'' reachable from M' which enables t.*

Definition 3 (Bounded, safe) *A Petri net (PN, M) is bounded iff, for every place p the number of tokens in p is bounded in every reachable state. The net is safe iff for each place the maximum number of tokens does not exceed 1.*

Definition 4 (Strongly connected) *A Petri net is strongly connected iff, for every pair of nodes (i.e. places and transitions) x and y, there is a path leading from x to y.*

4 WF-nets

Before we discuss the application of Petri nets to interorganizational workflow, we consider the modeling and analysis of workflows within one organization. In this paper, we focus on the 'process aspect' of workflow management, i.e. we abstract from data, resources and external triggers.

Workflows are *case-based*, i.e., every piece of work is executed for a specific *case*. Examples of cases are a mortgage, an insurance claim, a tax declaration, an order, or a request for information. Cases are often generated by an external customer. However, it is also possible that a case is generated by another department within the same organization (internal customer). The goal of workflow management is to handle cases as efficient and effective as possible. A workflow process is designed to handle similar cases. Cases are handled by executing *tasks* in a specific order. The *workflow process definition* specifies which tasks need to be executed and in what order. Alternative terms for workflow process definition are: 'procedure', 'flow diagram' and 'routing definition'. In the workflow process definition, building blocks such as the AND-split, AND-join, OR-split and OR-join are used to model sequential, conditional, parallel and iterative routing (WFMC [30]). Clearly, a Petri net can be used to specify the routing of cases. *Tasks* are modeled by transitions and causal dependencies are modeled by places. In fact, a place corresponds to a *condition* which can be used as pre- and/or post-conditions for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs. Moreover, in [3, 5] it is shown that the Petri net approach also allows for useful routing constructs absent in many workflow management systems.
A Petri net which models the process aspect of a workflow, is called a *WorkFlow net* (WF-net). It should be noted that a WF-net specifies the dynamic behavior of a single case in isolation.

Definition 5 (WF-net) *A Petri net $PN = (P, T, F)$ is a WF-net (Workflow net) if and only if:*

(i) *PN has two special places: i and o. Place i is a source place: $\bullet i = \emptyset$. Place o is a sink place: $o\bullet = \emptyset$.*

(ii) If we add a transition t^* to PN which connects place o with i (i.e. $\bullet t^* = \{o\}$ and $t^* \bullet = \{i\}$), then the resulting Petri net is strongly connected.

A WF-net has one input place (i) and one output place (o) because any case handled by the procedure represented by the WF-net is created if it enters the workflow management system and is deleted once it is completely handled by the workflow management system, i.e., the WF-net specifies the life-cycle of a case. The second requirement in Definition 5 (the Petri net extended with t^* should be strongly connected) states that for each transition t (place p) there should be a path from place i to o via t (p). This requirement has been added to avoid 'dangling tasks and/or conditions', i.e., tasks and conditions which do not contribute to the processing of cases.

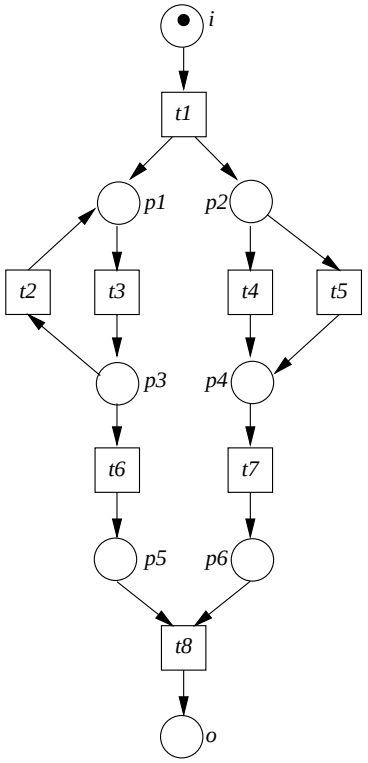

Figure 1: An example WF-net.

Figure 1 shows a WF-net. This WF-net models a workflow process composed of eight tasks: *t1*, ... ,*t8*. The execution of task *t1* enables two parallel flows: (i) *t3* and *t2* can be executed multiple times (iteration) followed by *t6*, and (ii) *t4* or *t5* followed by *t7*. The two parallel flows are synchronized by task *t8*. Note that sequential, conditional, parallel and iterative routing are present in this example.

5 Verification of WF-nets

The two requirements stated in Definition 5 can be verified statically, i.e., they only relate to the structure of the Petri net. However, there is an additional requirement which should be satisfied:

> *For any case, the procedure will terminate eventually and the moment the procedure terminates there is a token in place o and all the other places are empty.*

Moreover, there should be no dead tasks, i.e., it should be possible to execute an arbitrary task by following the appropriate route though the WF-net. These two additional constraints correspond to the so-called *soundness property*.

Definition 6 (Sound) *A procedure modeled by a WF-net $PN = (P, T, F)$ is sound if and only if:*

(i) For every state M reachable from state i, there exists a firing sequence leading from state M to state o. Formally:[2]

$$\forall_M (i[*\rangle M) \Rightarrow (M[*\rangle o)$$

(ii) State o is the only state reachable from state i with at least one token in place o. Formally:

$$\forall_M (i[*\rangle M \ \wedge \ M \geq o) \Rightarrow (M = o)$$

(iii) There are no dead transitions in (PN, i). Formally:

$$\forall_{t \in T} \ \exists_{M, M'} \ i[*\rangle M[t\rangle M'$$

Note that the soundness property relates to the dynamics of a WF-net. The first requirement in Definition 6 states that starting from the initial state (state *i*), it is always possible to reach the state with one token in place *o* (state *o*). If we assume a strong notion of fairness, then the first requirement implies that eventually state *o* will be reached. The fairness assumption is reasonable in the context of workflow management; all choices are made (implicitly en explicitly) by applications, humans or external actors. Clearly, they should not introduce an infinite loop. The second requirement states that the moment a token is put in place *o*, all the other places should be empty. Sometimes the term *proper termination* is used to describe the first two requirements [16]. The last requirement states that there are no dead transitions (tasks) in the initial state *i*.

Given WF-net $PN = (P, T, F)$, we want to decide whether PN is sound. For this purpose we define an extended net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$. $\overline{PN}$ is the Petri net that we obtain by adding an extra transition t^* which connects o and i. The extended Petri net $\overline{PN} = (\overline{P}, \overline{T}, \overline{F})$ is defined as follows: $\overline{P} = P$, $\overline{T} = T \cup \{t^*\}$, and $\overline{F} = F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\}$. Figure 2

[2] Note that there is an overloading of notation: the symbol *i* is used to denote both the *place i* and the *state* with only one token in place *i* (see Section 3).

illustrates the relation between PN and $\overline{PN}$. The extended net $\overline{PN}$ can be used to facilitate the verification of the soundness property. The following theorem shows that the extended net allows for the formulation of the soundness property in terms of well-known Petri net properties.

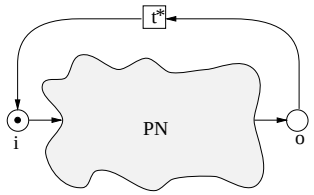

Figure 2: Given a WF-net PN, we construct an extended net $\overline{PN} = (P, T \cup \{t^*\}, F \cup \{\langle o, t^* \rangle, \langle t^*, i \rangle\})$.

Theorem 1 *A WF-net PN is sound if and only if $(\overline{PN}, i)$ is live and bounded.*

Proof.
See [2] or [1]. □

Perhaps surprisingly, the verification of the soundness property boils down to checking whether the extended Petri net is live and bounded! This means that we can use standard Petri-net-based analysis tools to decide soundness.

6 Interorganizational workflows

In the previous two sections, we applied Petri nets to the modeling and analysis of workflows within one organization. Now it is time to consider interorganizational workflows. An *interorganizational workflow* is essentially a set of loosely coupled workflow processes. Typically, there are *n* business partners which are involved in one 'global' workflow process. Each of the partners has its own 'local' workflow process. Each local workflow process is private, i.e. the corresponding business partner has full control over the local part of the workflow. However, these local workflow processes need to communicate because they depend on each other for the correct execution of cases. The global workflow process consists of local workflow processes and an interaction structure. There are two ways to interact: asynchronous communication and synchronous communication. Asynchronous communication corresponds to the exchange of messages between local workflow processes. Synchronous communication forces local workflow processes to execute specific tasks at the same time.

Figure 3 shows an interorganizational workflow which consists of two local workflows *LWF1* and *LWF2*. There are three *asynchronous communication elements*: *ac1*, *ac2*, and *ac3*. There is one synchronous communication element: *sc1*. The asynchronous communication elements are also called *communication places*. These communication places are

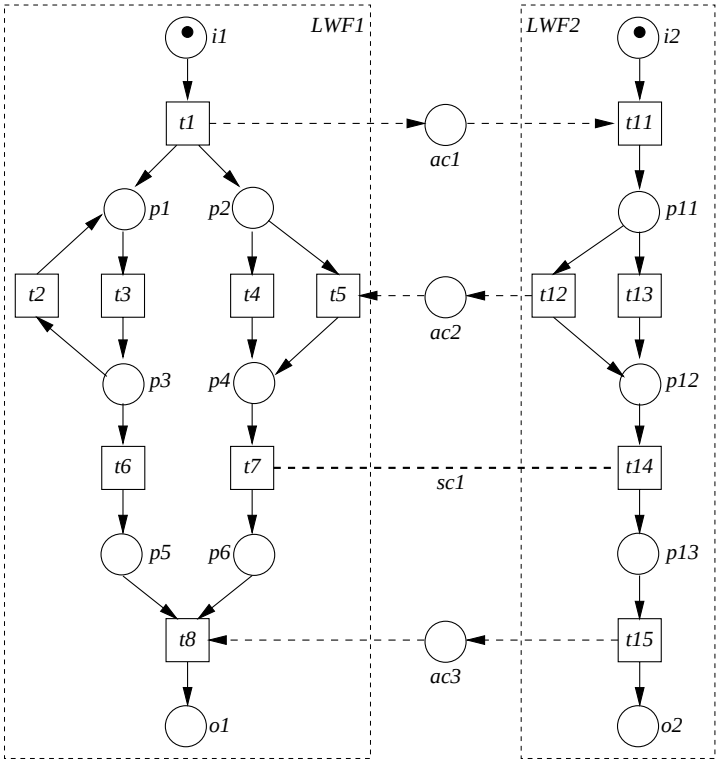

Figure 3: An interorganizational workflow composed of two local workflows.

used to model causal dependencies, e.g., task *t11* in *LWF2* has to wait for the completion of task *t1* in *LWF1*. The *synchronous communication element sc1* forces the transitions *t7* and *t14* to be executed at the same time. Synchronous communication corresponds to the melting of a number of transitions, therefore we also use the term *fusion set* to denote a synchronous communication element. Definition 7 formalizes the concept of an interorganizational workflow.

Definition 7 (IOWF) *An Interorganizational Workflow (IOWF) is a tuple* $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$*, where:*

(i) $n \in \mathbb{N}$ *is the number of local workflow nets,*

(ii) for each $k \in \{1, ..., n\}$*:* PN_k *is a WF-net with source place* i_k *and sink place* o_k*,*

(iii) for all $k, l \in \{1, ..., n\}$*: if* $k \neq l$*, then* $(P_k \cup T_k) \cap (P_l \cup T_l) = \emptyset$*,*

(iv) $T^{\square} = \bigcup_{k\in\{1,...,n\}} T_k$*,* $P^{\square} = \bigcup_{k\in\{1,...,n\}} P_k$*,* $F^{\square} = \bigcup_{k\in\{1,...,n\}} F_k$*,*

(v) P_{AC} *is the set of asynchronous communication elements (communication places),*

(vi) T_{SC} *is the set of synchronous communication elements (fusion sets),*

(vii) $P_{AC} \cap T_{SC} = \emptyset$*,* $(P_{AC} \cup T_{SC}) \cap (P^{\square} \cup T^{\square}) = \emptyset$*,*

(viii) $AC \subseteq P_{AC} \times \mathbb{P}(T^{\square}) \times \mathbb{P}(T^{\square})$ *is the asynchronous communication relation,*[3]

(ix) $SC \subseteq T_{SC} \times \mathbb{P}(T^{\square})$ *is the synchronous communication relation,*

(x) for all $p \in P_{AC}$, $\{(p', x, y) \in AC \mid p' = p\}$ *is a singleton,*

(xi) for all $t \in T_{SC}$, $\{(t', x) \in SC \mid t' = t\}$ *is a singleton,*

(xii) for all $(t_1, x_1), (t_2, x_2) \in SC$*: if* $t_1 \neq t_2$*, then* $x_1 \cap x_2 = \emptyset$.

Each asynchronous communication element corresponds to a place name in P_{AC}. The relation AC specifies a set of input transitions and a set of output transitions for each asynchronous communication element.
Requirement (x) specifies that for each communication place there is one element in AC. Each synchronous communication element is represented by a transition name in T_{SC} which corresponds to a set of fused transitions. The relation SC specifies for each element in T_{SC} the corresponding set of fused transitions. Requirement (xi) specifies that for each fusion set there is one element in SC. Requirement (xii) states that these fusion sets should be disjunct. Note that Definition 7 allows for communication elements which connect transitions within the same local workflow net. Although it does not make sense to do this, there is no compelling reason to forbid this kind of communication. Also note that each local workflow net has an input place i_k and an output place o_k. Sometimes there is no need for these places, e.g., if one organization is a subcontractor of another organization, then the workflow of the subcontractor may be initiated by a message (i.e. an asynchronous communication element). However, for semantical reasons we add the input place i_k and an output place o_k. Consider for example Figure 3; *i2* and *o2* can be removed without changing the actual behavior.

7 Verification of interorganizational workflows

In Section 5 we introduced a technique to verify the correctness of one workflow process definition in isolation. We can use this technique to prove that both local workflows in Figure 3 are correct, i.e., *LWF1* and *LWF2* are sound. However, an interorganizational workflow which is composed of a number of sound local workflows may be subject to synchronization errors. Consider for example the interorganizational workflow shown in Figure 3. It is possible that *LWF1* executes *t4* and *LWF2* executes *t12*. In this case, the message in *ac2* is not handled properly. It is also possible that deadlocks are introduced by the communication elements. If *t7* and *t12* are fused by a synchronous communication element, the workflow will not be able to terminate if *t13* is executed. Because of these problems, we are interested in a notion of soundness for interorganizational workflows. To define a notion of soundness suitable for interorganizational workflows (IO-soundness), we define the *unfolding* of an interorganizational workflow into a WF-net.

[3] $\mathbb{P}(T^{\square})$ is the set of all non-empty subsets of $T^{\square}$.

Definition 8 *Let* $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ *be an interorganizational workflow.* $\mathcal{U}(IOWF) = (P^U, T^U, F^U)$ *is the unfolding of* $IOWF$ *which is defined as follows:*

(i) $P^U = P^\square \cup P_{AC} \cup \{i, o\}$,

(ii) $T^U = r(T^\square) \cup T_{SC} \cup \{t_i, t_o\}$,

(iii) $\{i, o, t_i, t_o\} \cap (P^\square \cup T^\square \cup P_{AC} \cup T_{SC}) = \emptyset$,

(iv) r *is a renaming function:* $r(x) = t_{SC}$ *if there is a* $t_{SC} \in T_{SC}$ *and a* $y \subseteq T^\square$ *such that* $(t_{SC}, y) \in SC$ *and* $x \in y$, *otherwise* $r(x) = x$,

(v) $F' = F^\square \cup \{(t, p) \in T^\square \times P_{AC} \mid (p, x, y) \in AC \wedge t \in x\} \cup$
$\{(p, t) \in P_{AC} \times T^\square \mid (p, x, y) \in AC \wedge t \in y\} \cup$
$\{(i, t_i), (t_o, o)\} \cup \{(t_i, i_k) \mid k \in \{1, ..., n\}\} \cup \{(o_k, t_o) \mid k \in \{1, ..., n\}\}$

(vi) $F^U = \{(r(x), r(y)) \mid (x, y) \in F'\}$.

In the unfolded net all the local WF-nets are connected to each other by a start transition t_i and a termination transition t_o. Moreover, a global source place i and a global sink place o have been added. Asynchronous communication elements are mapped onto ordinary places (P_{AC}). Transitions which are fused together by synchronous communication elements are replaced by new transitions (T_{SC}). Note that we use a renaming function r to map old transitions onto new transitions. The result of the unfolding is a new WF-net.

Lemma 1 *Let* $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ *be an IOWF.* $\mathcal{U}(IOWF) = (P^U, T^U, F^U)$ *is a WF-net.*

Proof.
It is easy to see that the two requirements stated in Definition 5 are satisfied: (i) there is one source place i and one sink place o, and (ii) every node is on a path from i to o . □

It is easy to see that the behavior of the unfolded net corresponds to the overall behavior of the interorganizational workflow. This allows us to define the soundness property for interorganizational workflows.

Definition 9 (IO-Soundness) *An interorganizational workflow* $IOWF$ *is IO-sound iff it is locally sound and globally sound.* $IOWF$ *is locally sound iff each of its local workflow nets* PN_k *is sound.* $IOWF$ *is globally sound iff* $\mathcal{U}(IOWF)$ *is sound.*

The interorganizational workflow shown in Figure 3 is an example of workflow which is locally sound but not globally sound. The unfolded net is not sound, because if *t4* and *t12* fire, a token gets stuck in place *ac2*. The error can be corrected by replacing the asynchronous communication element *ac2* by a synchronous communication element. Figure 4 shows an example of an interorganizational workflow which is globally sound but not locally sound. The WF-net *LWF2* is not sound because an arbitrary number of tokens may get trapped in place *p14*. However, asynchronous communication element *ac2* prevents

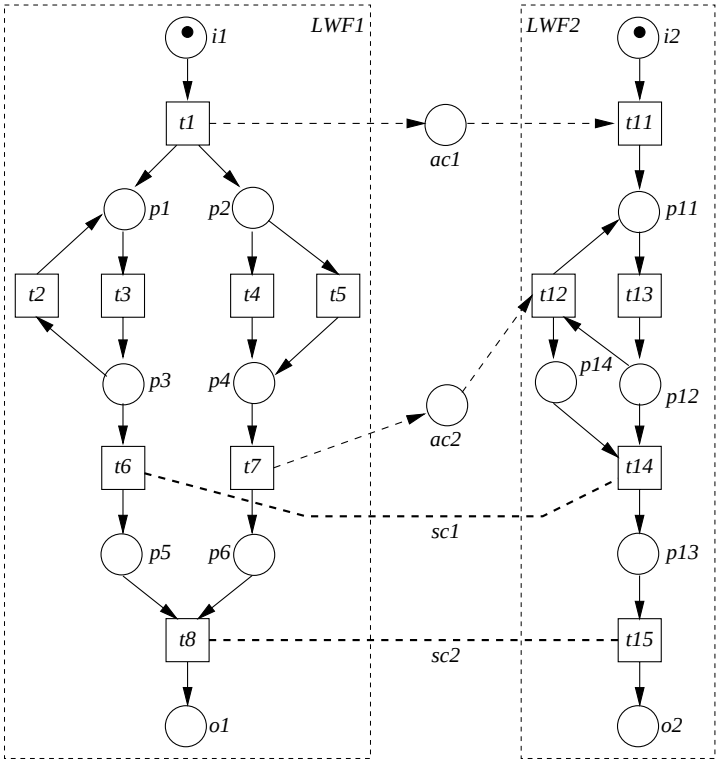

Figure 4: An IOWF which is globally sound but not locally sound.

this from happening in the unfolded net. These examples show that it is possible to have a locally sound interorganizational workflow which is not globally sound and vice-versa. In order to be truly sound (IO-sound), the interorganizational workflow should be both locally sound and globally sound. Figure 5 is an example of a sound interorganizational workflow.

The IO-soundness of an interorganizational workflow $IOWF = (PN_1, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ corresponds to the soundness of $n + 1$ WF-nets: $PN_1, ..., PN_n$ and $\mathcal{U}(IOWF)$. Therefore, we can use Theorem 1 to verify the correctness of interorganizational workflows. This means that we can use standard techniques and software tools. For example, we can use *Woflan* ([19]). Woflan is an analysis tool dedicated to the analysis of workflows which are specified in terms of Petri nets.

For arbitrary interorganizational workflows, IO-soundness is decidable but also EXPSPACE-hard ([1]). However, there are some interesting subclasses which allow for more efficient analysis techniques. For example, many workflow management systems only allow for workflow process definitions which correspond to free-choice Petri nets ([11]). For this subclass and several others (cf. [1]), the soundness property can be verified in polynomial time.

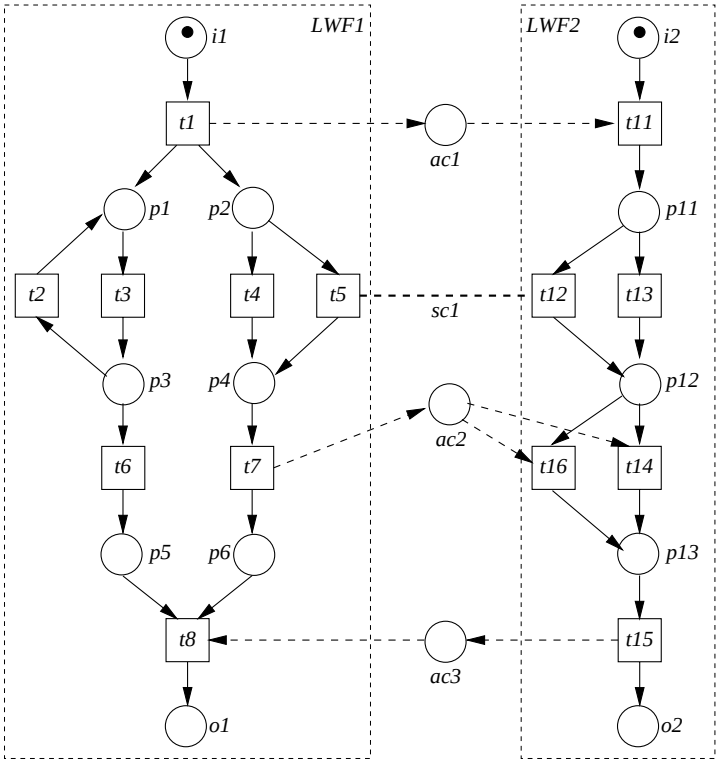

Figure 5: An IO-sound interorganizational workflow.

8 Message sequence charts

Interorganizational workflows are described in terms of individual tasks and causal relations. In most cases, the design of an interorganizational workflow starts with the specification of the communication structure, i.e., the protocol. Clearly, a description in terms of a Petri net is too detailed to start with. Therefore, another technique is needed to specify the communication structure between multiple loosely coupled workflows. In this paper, we use *Message Sequence Charts* (MSC) extended with synchronous communication for this purpose.

Message sequence charts are a widespread graphical language for the visualization of communications between systems/processes [9, 25, 29, 17]. The representation of message sequence charts is intuitive and focuses on the messages between communication entities. Figure 6 shows a message sequence chart with three business partners also called *instances*. Instances communicate via *messages*. Messages are either *asynchronous* (*m1*, *m2*, *m4*, *m5*, *m7*, *m8*, and *m9*) or *synchronous* (*m3* and *m6*). Note that a standard message sequence chart does not allow for synchronous messages. We need synchronous messages to model synchronous communication, e.g., a phone call to exchange information. Each message has a sender and a receiver. For example, instance 1 is the sender of *m1* and instance 2 is the receiver of *m1*. For asynchronous messages the message is received by the receiver only after it has been sent by the sender. Each synchronous message results in the

synchronization of two instances. Within each instance events are ordered, e.g., instance 3 sends *m4* only after the receipt of *m2*. The ordering of events is specified by the time axis of an instance which is represented by a vertical line. In a message sequence chart, it is also possible to specify coregions. A coregion is represented by a dashed part of the time axis of an instance. Events in a coregion are assumed to be unordered in time. In Figure 6 there is one coregion: the receipt of *m4*, the receipt of *m5*, and the sending of *m7* are unordered in time.

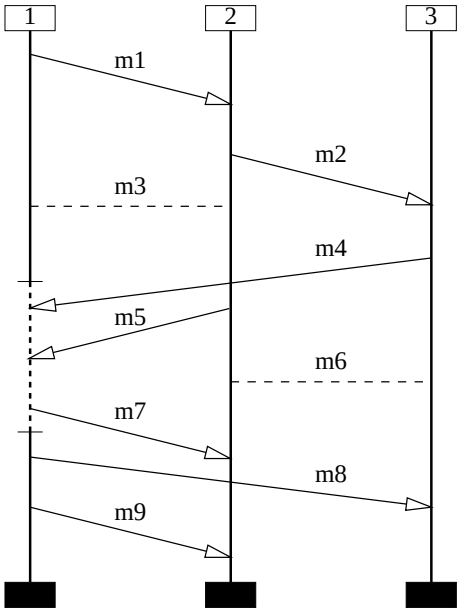

Figure 6: A message sequence chart.

In this paper, we consider a variant of message sequence charts as defined in [9] or [21]. The basic message sequence chart has been extended with synchronous messages and coregions. However, for reasons of simplicity, we do not allow for process creation, process termination, timers, and refinement. To formalize the semantics of the message sequence charts used in this paper, we use a notation adopted from [14].

Definition 10 (Message sequence chart) *A message sequence chart is a tuple* $MSC = (I, M_A, M_S, from, to, \{\leq_i\}_{i \in I})$*:*

- I *is a finite set of instances (business partners),*
- M_A *is a finite set of asynchronous messages,*
- M_S *is a finite set of synchronous messages,*
- $M_A \cap M_S = \emptyset$ *and* $M = M_A \cup M_S$ *is the set of messages,*
- to *and* $from$ *are functions from* M *to* I*,*

- *for each* $i \in I$: $\leq_i$ *is a partial order on* $\{?m \mid m \in M_A \wedge to(m) = i\} \cup \{!m \mid m \in M_A \wedge from(m) = i\} \cup \{!?m \mid m \in M_S \wedge i \in \{to(m), from(m)\}\}$.

If m is an asynchronous message, then $!m$ corresponds to the event of sending the message and $?m$ corresponds to the event of receiving the message. For a synchronous message both instances synchronize on the event $!?m$. For each instance i, $\leq_i$ specifies the ordering of events along the time axis of i. $\leq_i$ is a partial order instead of a total ordering because of the existence of coregions. In Figure 6, $!m1 \leq_i !?m3$, $!?m3 \leq_i ?m4$, $!?m3 \leq_i ?m5$,$!?m3 \leq_i !m7$, $?m4 \leq_i !m8$, $?m5 \leq_i !m8$, $!m7 \leq_i !m8$, and $!m8 \leq_i !m9$. Note that the events in the coregion are unordered, e.g., $?m4 \not\leq_i !m7$.

Definition 11 ($\leq^{MSC}$) *Let* $MSC = (I, M_A, M_S, from, to, \{\leq_i\}_{i \in I})$ *be a message sequence chart.*

- $\leq^{inst} = \bigcup_{i \in I} \leq_i$,
- $\leq^{oi} = \{(!m, ?m) \mid m \in M_A\}$,
- $\leq^{MSC} = (\leq^{inst} \cup \leq^{o})^+$.

$\leq^{oi}$ is a partial order which reflects the production before consumption principle. $\leq^{MSC}$ is the transitive closure of (1) the partial orders within the instances ($\leq^{ins}$) and (2) the partial order between the production and consumption of asynchronous messages ($\leq^{oi}$). Consider for example the message sequence chart shown in Figure 6: $!m2 \leq^{MSC} ?m9$, $!?m3 \leq^{MSC} ?m8$, $!m4 \not\leq^{MSC} ?m5$, $?m8 \not\leq^{MSC} ?m9$.
A message sequence chart MSC is inconsistent iff $\leq^{MSC}$ does not define a partial order. In this case, the message sequence chart contains a deadlock due to cyclic dependencies. In the remainder we assume that the message sequence charts are consistent.
Note that $\leq^{MSC}$ is a partial order on $A = \{?m \mid m \in M_A\} \cup \{!m \mid m \in M_A\} \cup \{!?m \mid m \in M_S\}$. A is a the set of *event labels*.

9 Consistency of interorganizational workflows

Message sequence charts can be used to specify the interaction between loosely coupled workflow processes. These message sequence charts serve as a starting point for the design of complex interorganizational workflows. The interorganizational workflow should be designed in such a way that it is consistent with the message sequence charts, i.e., the message sequence charts can be seen as a partial specification of an interorganizational workflow. Therefore, it is interesting to be able to decide whether the implementation (interorganizational workflow) meets the specification (message sequence charts).

Figure 7 shows the relation between the behavior specified by the message sequence charts (MSC) and the behavior possible in the interorganizational workflow (IOWF). Ideally, IOWF and MSC coincide. If this is not the case, there are three possibilities:

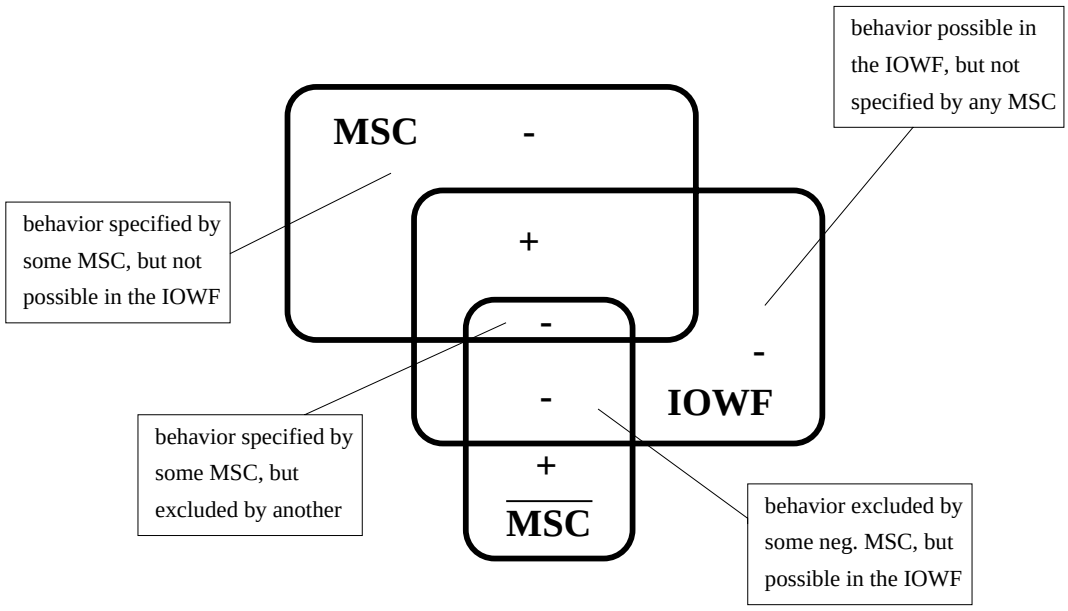

Figure 7: The relation between the behavior specified by the message sequence charts and the interorganizational workflow.

1. The IOWF allows for communication patterns which are not specified in any of the MSC's, i.e., the MSC's are more restrictive.

2. There is an MSC which describes a communication pattern which cannot be realized by the IOWF, i.e., the IOWF is more restrictive.

3. A combination of the previous two. On the one hand the IOWF allows for a pattern not in any of the MSC's. On the other hand, the IOWF excludes a pattern which corresponds to one of the MSC's.

If there are many alternative communication patterns, it will often be the case that there is not an exact match. On the one hand, it is difficult to give an exhaustive description of all possible communication patterns in terms of message sequence charts. In most cases only the typical patterns are given in terms of a limited set of message sequence charts. On the other hand, the behavior of the interorganizational workflow can be more restrictive to facilitate the implementation, to improve the performance, or to exploit knowledge. For example, based on the characteristics of the case (e.g. the invoice amount of an order exceeds 100.000 dollar), certain patterns are excluded (e.g. the business partner that has to deliver the goods will wait until the invoice is paid). Although it is not realistic to assume an exact match between the behavior specified by the message sequence charts and the behavior possible in the interorganizational workflow, it is interesting to see in what way they deviate from each other. The business partners have to observe these deviations to see whether they are acceptable or not. Deviations which are not acceptable lead to modifications of the interorganizational workflow or result in the creation, deletion, or modification

of message sequence charts.

It is also possible to formulate *negative* message sequence charts. A negative message sequence chart corresponds to a communication pattern which should not occur. Since the number of admissible patterns is enormous, it can be more efficient to specify faulty behavior instead of a complete specification of desired behavior. Figure 7 also shows the relation between the behavior of the interorganizational workflow (IOWF) and the message sequence charts which correspond to anomalous communication patterns ($\overline{\text{MSC}}$). IOWF and $\overline{\text{MSC}}$ should be disjuct. A communication pattern in the intersection of IOWF and $\overline{\text{MSC}}$ is not acceptable and should lead to a modification of the interorganizational workflow. Note that it is possible that MSC and $\overline{\text{MSC}}$ overlap (see Figure 7). In this case, there is a conflict because there is a pattern which corresponds to both desired and anomalous behavior. Such a conflict needs to be resolved before investigating whether the message sequence charts and the interorganizational workflow are consistent.

Verifying whether a set of message sequence charts and an interorganizational workflow are consistent is very difficult. Comparing the dynamic behavior of two models and deciding whether they are equivalent is known to be a hard problem from a computational point of view. One can even think of situations where equivalence of behavior is undecidable (e.g. the equality problem for Petri nets is undecidable [18]). In fact, for the general case it is even difficult to formalize a suitable notion of consistency. To be able to distinguish between external, or observable, behavior and internal, or silent, behavior, we need a notion of branching bisimilarity ([15]) to compare message sequence charts and interorganizational workflows. Moreover issues such as iteration and fixing the moment of choice complicate the definition of consistency.

In this paper, we tackle the problem of deciding consistency for the situation where there is just one message sequence chart. This means that constructs such as choice and iteration are only allowed inside each of the local workflow processes. The business partners involved in the global workflow process communicate according to one predefined communication pattern. We will use the term *1-consistency* for this restricted notion of consistency. A message sequence chart and an interorganizational workflow are 1-consistent if their corresponding behaviors coincide. Although there are many situations where the notion of 1-consistency is not applicable, there are several reasons for investigating this property. First of all, there are situations where business partners follow one predefined communication pattern. Secondly, even if the whole interaction structure comprises many alternative communication patterns, parts without iteration and choice should be 1-consistent. Thirdly, exceptions cause iteration and choice. By abstracting from these exceptions we often obtain a situation where 1-consistency applies. Finally, the restriction to 1-consistency allows for the application of the structural theory of Petri nets. In Section 11, we will show that structural implicit places ([8]) can be used to prove 1-consistency, thus avoiding the state explosion problem.

10 Definition of 1-consistency

In this section, we define the notion of 1-consistency and in the next section we present a technique to verify whether an interorganizational workflow is 1-consistent with respect to a message sequence chart.

Definition 12 ($\mathcal{L}$) *Let $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ be an interorganizational workflow and $\mathcal{U}(IOWF) = (P^U, T^U, F^U)$. $\mathcal{L}$ is a function from T^U to the powerset of $\{?m \mid m \in P_{AC}\} \cup \{!m \mid m \in P_{AC}\} \cup \{!?m \mid m \in T_{SC}\}$. For $t \in T^U$, $\mathcal{L}(t) = \{?m \mid m \in P_{AC} \wedge m \in \bullet t\} \cup \{!m \mid m \in P_{AC} \wedge m \in t\bullet\} \cup \{!?m \mid m \in T_{SC} \wedge m = t\}$.*

Function $\mathcal{L}$ maps transitions onto the events associated with the transition. Consider for example the interorganizational workflow shown in Figure 8: $\mathcal{L}(t11) = \{!m1\}$, $\mathcal{L}(t21) = \emptyset$, $\mathcal{L}(t22) = \{?m1, !m2\}$, and $\mathcal{L}(t24) = \{!m5, !?m6\}$.
In a message sequence chart each message is sent exactly once. Therefore, we introduce the notion of 1-liveness.

Definition 13 (1-live) *A transition t in a Petri net (PN, M) is 1-live iff, (1) for every state M' reachable without firing t, there is a state M'' reachable from M' which enables t, and (2) for every state M' reachable via a firing sequence which fires t, there is no state reachable from M' which enables t.*

In a message sequence chart each message is exchanged between two instances. Therefore, we demand that each communication place in P_{AC} has one sender and one receiver. Moreover, each transition involved in the communication between instances has to be 1-live. An interorganizational workflow which meets these two requirements is called a restricted interorganizational workflow. Figure 8 shows an example of a restricted interorganizational workflow.

Definition 14 (Restricted IOWF) *A restricted interorganizational workflow (RIOWF) is an interorganizational workflow $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$, which satisfies the following requirements:*

(i) for each $(p, x, y) \in AC$: x and y are singletons,

(ii) for each transition t in $\mathcal{U}(IOWF)$: if $\mathcal{L}(t) \neq \emptyset$, then t is 1-live in $(\mathcal{U}(IOWF), i)$.

For restricted interorganizational workflows we define the inverse function of $\mathcal{L}$. $\mathcal{L}^{-1}$ maps each element of $\{?m \mid m \in P_{AC}\} \cup \{!m \mid m \in P_{AC}\} \cup \{!?m \mid m \in T_{SC}\}$ onto a single transition in T^U.
For restricted interorganizational workflows we define the notion of 1-consistency.

Definition 15 (1-consistent) *Let $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ be a restricted interorganizational workflow and let $MSC = (I, M_A, M_S, from, to, \{\leq_i\}_{i \in I})$ be a message sequence chart. $IOWF$ is 1-consistent with respect to MSC if and only if*

(i) $P_{AC} = M_A$ and $T_{SC} = M_S$,

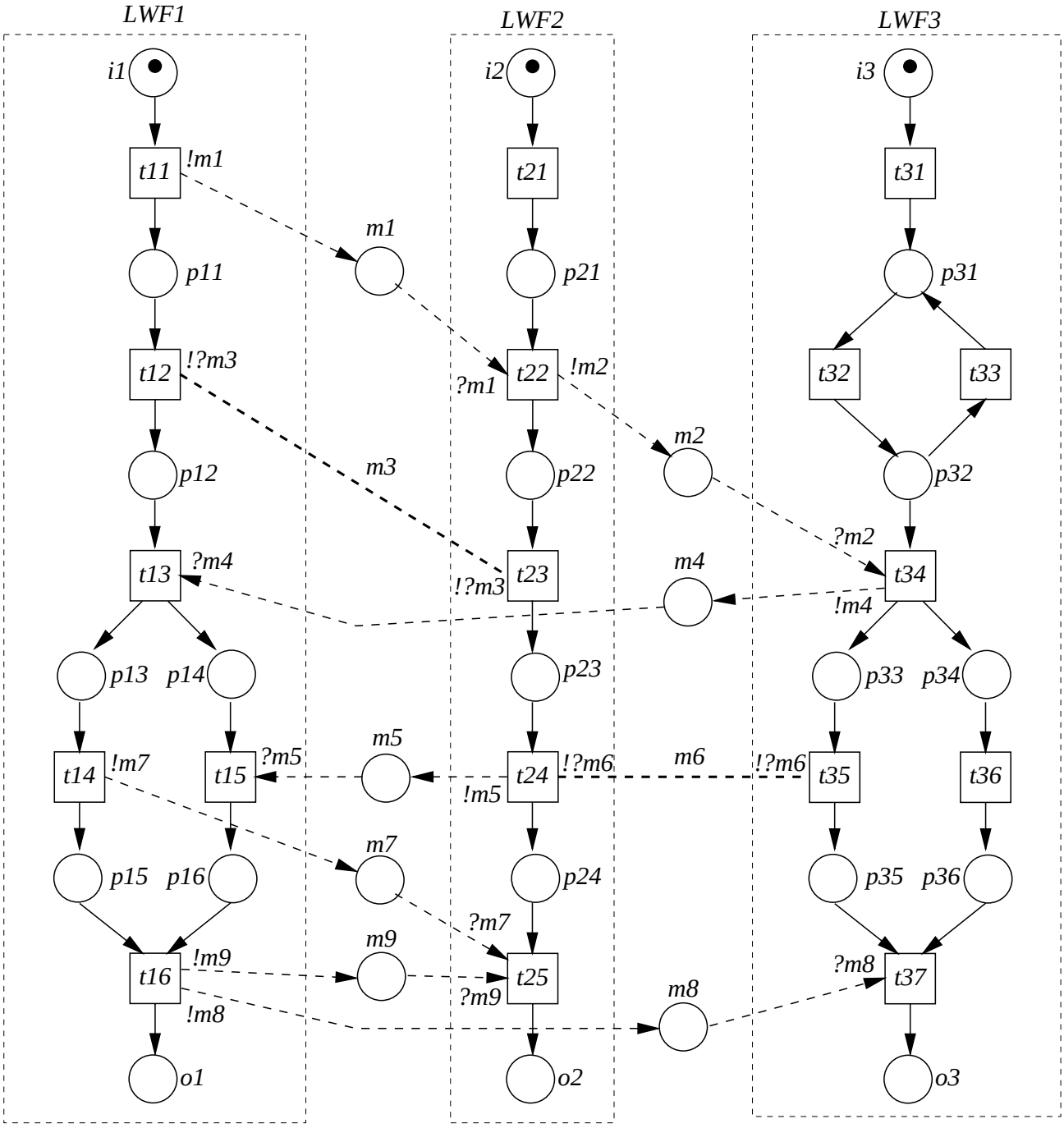

Figure 8: A restricted interorganizational workflow.

(ii) $\mathcal{U}(IOWF) = (P^U, T^U, F^U)$ *is the unfolding of* $IOWF$ *with source place* i*. For each* $t_1, t_2 \in T^U$*: if there is a firing sequence starting in state* i *which fires transition* t_1 *before transition* t_2*, then* $\forall_{a_1 \in \mathcal{L}(t_1)} \forall_{a_2 \in \mathcal{L}(t_2)} \neg(a_2 \leq^{MSC} a_1)$.

A restricted interorganizational workflow is 1-consistent with respect to a message sequence chart if the message names used in the message sequence chart correspond to the names of communication elements in the interorganizational workflow and none of the possible firing sequences violates precedence constraints specified in the message sequence chart. The interorganizational workflow shown in Figure 8 is 1-consistent with respect to the message sequence chart shown in Figure 6. However, if place *m9* connects *t16* and *t22* instead of *t16* and *t25*, then the interorganizational workflow is not 1-consistent with respect to a message sequence chart in Figure 6. The inconsistency is a result of the fact that there is a firing sequence such that *t22* fires before *t16*, $?m9 \in \mathcal{L}(t22)$, $!m9 \in \mathcal{L}(t16)$, and $(!m9 \leq^{MSC} ?m9)$. 1-consistency can be verified by generating all possible firing sequences and checking whether the partial order $\leq^{MSC}$ is not violated by any of these sequences. Clearly, 1-consistency is decidable but also very hard to verify if all firing sequences have to be considered. Therefore, we propose a technique to facilitate the verification of 1-consistency.

11 Verification of 1-consistency

A restricted interorganizational workflow is 1-consistent with respect to a message sequence chart if all possible firing sequences satisfy the partial order $\leq^{MSC}$. Instead of checking all possible firing sequences we propose a technique based on the notion of *implicit places* ([8]). An implicit place, also called a redundant place, is a place which always contains sufficient tokens to allow for the firing of the transitions connected to it. In this paper, we use a generalized notion: the *implicit place set*.

Definition 16 (Implicit place set) *Let* (PN, M) *be a marked Petri net with* $PN = (P, T, F)$ *and* $P_I \subseteq P$*.* P_I *is an implicit place set iff for every reachable state* M' *and any transition* $t \in T$*: if each place in* $(\bullet t \setminus P_I)$ *contains a token in state* M'*, then each place in* $(\bullet t \cap P_I)$ *contains a token in* M'*. Place* $p \in P$ *is an implicit place iff* $\{p\}$ *is an implicit place set.*

To introduce some basic results for implicit place sets, we define the projection operator ($\Uparrow$).

Definition 17 ($\Uparrow$) *Let* (PN, M) *be a marked Petri net with* $PN = (P, T, F)$ *and* $P' \subseteq P$*.*

- $PN \Uparrow P' = (P', T, F \cap ((P' \times T) \cup (T \times P')))$,
- $(M \Uparrow P') \in P' \to \mathbb{N}$, $\forall_{p \in P'} (M \Uparrow P')(p) = M(p)$,
- $(PN, M) \Uparrow P' = (PN \Uparrow P', M \Uparrow P')$.

An implicit place set does not restrict the set of possible firing sequences. Therefore, it can be removed without changing the behavior. Moreover, a set of places is an implicit place set, if and only if, each of the places is implicit.

Lemma 2 *Let (PN_1, M_1) be a marked Petri net with $PN_1 = (P_1, T_1, F_1)$ and $P_I \subseteq P_1$. Let $(PN_2, M_2) = (PN_1, M_1) \Uparrow (P_1 \setminus P_I)$ be the Petri net obtained by removing the places in P_I.*

(i) $\forall_{\sigma \in T_1^*}\ M_1[\sigma\rangle_{PN_1} \ \Rightarrow\ M_2[\sigma\rangle_{PN_2}$

(ii) If P_I is an implicit place set of (PN_1, M_1), then $\forall_{\sigma \in T_2^*}\ M_2[\sigma\rangle_{PN_2} \ \Rightarrow\ M_1[\sigma\rangle_{PN_1}$.

(iii) P_I is an implicit place set of (PN_1, M_1), if and only if, for all $p \in P_I$: p is an implicit place of $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$.

Proof.
(i) Trivial, removing places does not restrict the set of possible firing sequences.
(ii) Suppose that (ii) does not hold, i.e., P_I is an implicit place set and σ is a firing sequence which leads to a state M_2' in PN_2 where transition t is enabled and σ leads to a state M_1' in PN_1 where t is not enabled. Clearly, this is not possible because P_I is an implicit place set.
(iiia) Suppose that $\Rightarrow$ does not hold, i.e., P_I is an implicit place set but there is a p which is not an implicit place of $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$. There is a firing sequence σ in $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$ which leads to a state where a transition t is not enabled (p is empty) but each place in $\bullet t \cap (P_1 \setminus P_I)$ contains a token. σ is also a possible firing sequence in (PN_1, M_1) (apply (i) and (ii)). The state reached by firing σ in (PN_1, M_1) is such that t is not enabled (p is empty) but each place in $\bullet t \setminus P_I$ is marked. Hence, P_I is not an implicit place set.
(iiib) Suppose that $\Leftarrow$ does not hold, i.e., for all $p \in P_I$: p is an implicit place of the marked net $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$ but P_I is not an implicit place set. There is a firing sequence σ in (PN_1, M_1) which leads to a state such that transition t is not enabled but each place in $\bullet t \cap (P_1 \setminus P_I)$ is marked. This means that there is a $p \in (\bullet t \cap P_I)$ which is not marked. σ is also a possible firing sequence in $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$ (apply (i)). The state reached by firing σ in $(PN_1, M_1) \Uparrow ((P_1 \setminus P_I) \cup \{p\})$ is such that t is not enabled (p is empty) but each place in $\bullet t \setminus \{p\}$ is marked. Hence, p is not an implicit place.
□

1-consistency can be checked via an approach based on implicit place sets. Transitions in a restricted interorganizational workflow are associated with events. A message sequence chart specifies a partial order on these events. Therefore, the message sequence chart indirectly specifies a partial order on transitions. This partial order can be expressed in terms of places connecting transitions. The following theorem shows that these additional places are implicit, if and only if, the interorganizational workflow is 1-consistent with respect to the message sequence chart.

Theorem 2 *Let $IOWF = (PN_1, PN_2, ..., PN_n, P_{AC}, AC, T_{SC}, SC)$ be a restricted interorganizational workflow and let $MSC = (I, M_A, M_S, from, to, \{\leq_i\}_{i \in I})$ be a message sequence chart such that $P_{AC} = M_A$ and $T_{SC} = M_S$. Let $\mathcal{U}(IOWF) = (P^U, T^U, F^U)$ be the unfolding of $IOWF$ and $\mathcal{V}(IOWF) = (P^V, T^V, F^V)$ a Petri net defined as follows:*

- $P_I = \{(a_1, a_2) \in \leq^{MSC} \mid \mathcal{L}^{-1}(a_1) \neq \mathcal{L}^{-1}(a_2)\}$ *and* $P_V = P_U \cup P_I$.
- $T_V = T_U$,
- $F_V = F_U \cup \{(t, p) \mid p = (a_1, a_2) \in P_I \wedge t \in T_V \wedge a_1 \in \mathcal{L}(t)\} \cup \{(p, t) \mid p = (a_1, a_2) \in P_I \wedge t \in T_V \wedge a_2 \in \mathcal{L}(t)\}$.

$IOWF$ is 1-consistent with respect to MSC, if and only if, P_I is an implicit place set of $(\mathcal{V}(IOWF), i)$.

Proof.
Both the 'if' and the 'only if' part are proven using contraposition.
(i) Suppose $IOWF$ is 1-consistent with respect to MSC and P_I is not an implicit place set of $(\mathcal{V}(IOWF), i)$. There is place $p = (a_2, a_1) \in P_I$, a transition t_1 such that $p \in {\bullet}t_1$, and a possible firing sequence σ such that in the state reached by firing σ all the places in ${\bullet}t_1 \cap P_U$ are marked and p is empty (i.e. transition t_1 is not enabled). Since p is a place in P_I, there is just one transition t_2 which puts tokens in p. Transition t_2 is the only transition for which $a_2 \in \mathcal{L}(t_2)$. Note that $t_1 \neq t_2$, because $t_1 = \mathcal{L}^{-1}(a_1)$, $t_2 = \mathcal{L}^{-1}(a_2)$, $\mathcal{L}^{-1}(a_1) \neq \mathcal{L}^{-1}(a_2)$. σ is also a possible firing sequence in $(\mathcal{U}(IOWF), i)$, see Lemma 2(i). Moreover σt_1 is a possible firing sequence. Transition t_1 is 1-live, i.e., in σt_1 there is precisely one firing of t_1. Hence t_1 does not occur in σ. In $(\mathcal{V}(IOWF), i)$, t_1 is the only transition which consumes tokens from p. If p is empty after firing σ, then t_2 does not occur in σ. Because t_2 is 1-live in $(\mathcal{U}(IOWF), i)$ and $t_1 \neq t_2$, it is possible to extend firing sequence σt_1 into a firing sequence where t_1 fires before t_2. Moreover, by the definition of P_I, $a_2 \leq^{MSC} a_1$, $a_1 \in \mathcal{L}(t_1)$ and $a_2 \in \mathcal{L}(t_2)$. Hence, $IOWF$ is not 1-consistent.
(ii) Suppose P_I is an implicit place set of $(\mathcal{V}(IOWF), i)$ and $IOWF$ is not 1-consistent with respect to MSC. There exists a firing sequence σ in $(\mathcal{U}(IOWF), i)$ such that t_1 fires before t_2 and there is a $a_1 \in \mathcal{L}(t_1)$ and $a_2 \in \mathcal{L}(t_2)$ such that $a_2 \leq^{MSC} a_1$. Moreover, t_1 and t_2 are 1-live and $t_1 = \mathcal{L}^{-1}(a_1) \neq \mathcal{L}^{-1}(a_2) = t_2$. Therefore, $(a_2, a_1) \in P_I$. σ is also a possible firing sequence of $(\mathcal{V}(IOWF), i)$, see Lemma 2(ii). (a_2, a_1) is a place in $\mathcal{V}(IOWF)$ which connects t_2 and t_1. Transition t_2 is the only transition which produces tokens for (a_2, a_1), the input place of t_1. Therefore, t_2 has to fire before t_1 and σ is not a possible firing sequence of $(\mathcal{V}(IOWF), i)$. □

Theorem 2 shows that 1-consistency can be verified by checking whether a set of places is implicit. Note that the set P_I as a whole has to be an implicit place set. It is not sufficient to demand that the individual places in P_I are implicit. This is the reason we extended the notion of implicit place to implicit place *set*. Nevertheless, we can express the notion of implicit place set into the more well-known notion of implicit place (see Lemma 2(iii)).

The observation that 1-consistency corresponds to implicit places allows for the application of well-known Petri-net-based techniques. In fact there are several tools, which allow for the detection of implicit places. Theorem 2 shows that it is not necessary to check whether every possible firing sequence matches the partial order specified by the message

sequence chart.

From a computational point of view, verification becomes expensive if there are a lot of places in the set P_I, i.e., the are a lot of ordering relations to be checked. However, it is possible to reduce the set P_I to $\leq^{inst}$ because the ordering imposed by $\leq^{oi}$ is already present and the transitive closure follows from the net structure. Therefore, only a limited number of places needs to be checked. Moreover, several authors have investigated techniques to find structural implicit places ([10, 7, 8]). 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. The following definition is a generalization of the definition in [8] for implicit place sets.

Definition 18 (Structural implicit place set) *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$ and $P_I \subseteq P$. P_I is a structural implicit place set iff for each $p \in P_I$ there is a set $Q \subseteq (P \setminus P_I)$ such that.*

(i) $M(p) \geq \sum_{q \in Q} M(q)$

(ii) $\forall_{t \in T}\ X_{tp} \geq \sum_{q \in Q} X_{tq}$, *where X is the incidence matrix of PN*

(iii) $\forall_{t \in T}\ p \in {\bullet t} \Rightarrow Q \cap {\bullet t} \neq \emptyset$

There are several differences with the definition in [8]. First of all, the notion has been extended to sets. Secondly, we only consider nets with arc weights 1. Finally, in (ii) we replaced the equality (=, i.e. place invariant) by $\geq$. It is easy to see that a structural implicit place set is implicit indeed.

Lemma 3 *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$ and $P_I \subseteq P$. If P_I is a structural implicit place set, then P_I is an implicit place set.*

As a consequence of Lemma 3, we can prove that an interorganizational workflow is 1-consistent with respect to a message sequence chart by checking whether the set P_I is a structural implicit place set. Consider for example the interorganizational workflow shown in Figure 8. It suffices to consider the structural implicit places to prove that the workflow is 1-consistent with respect to the message sequence chart shown in Figure 6.

12 Conclusion

In this paper, we have shown that a particular kind of interorganizational workflows can be modeled in terms of a set of Petri nets connected via communication elements. We also investigated a basic property that any interorganizational workflow should satisfy: IO-soundness. We have seen that this property coincides with well-known Petri net properties.

To establish the correctness of an interorganizational workflow composed of n local workflows, we need to prove liveness and boundedness for $n + 1$ WF-nets using standard techniques. In most cases, the interorganizational workflow should obey a given communication structure in addition to the IO-soundness property. In this paper, we showed that given a message sequence chart which specifies the communication between business partners, it is possible to use an efficient technique to verify whether the interorganizational workflow is 1-consistent with the message sequence chart. This technique uses an extension of the concept of implicit places.
At the moment our notion of consistency is restricted to the situation with one message sequence chart. Future work will aim at extending some of the results for the situation with multiple message sequence charts (i.e. n-consistency).

References

[1] W.M.P. van der Aalst. Structural Characterizations of Sound Workflow Nets. Computing Science Reports 96/23, Eindhoven University of Technology, Eindhoven, 1996.

[2] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azema 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.

[3] W.M.P. van der Aalst. Chapter 10: Three Good reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama et al., editor, *Information and Process Integration in Enterprises: Rethinking documents*, The Kluwer International Series in Engineering and Computer Science, pages 161–182. Kluwer Academic Publishers, Norwell, 1998.

[4] W.M.P. van der Aalst. Modeling and Analyzing Interorganizational Workflows. In L. Lavagno and W. Reisig, editors, *Proceedings of the 1998 International Conference on Application of Concurrency to System Design (CSD'98)*, pages 262–272, Fukushima, Japan, March 1998. IEEE Computer Society Press.

[5] 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.

[6] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Modellen, Methoden en Systemen (in Dutch)*. Academic Service, Schoonhoven, 1997.

[7] 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.

[8] 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.

[9] CCITT. CCITT Recomendation Z.120: Message Sequence Chart (MSC92). Technical report, CCITT, Geneva, 1992.

[10] 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.

[11] J. Desel and J. Esparza. *Free choice Petri nets*, volume 40 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, Cambridge, 1995.

[12] C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock and C. Ellis, editors, *Conf. on Organizational Computing Systems*, pages 10 – 21. ACM SIGOIS, ACM, Aug 1995. Milpitas, CA.

[13] 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.

[14] A. Engels, S. Mauw, and M.A. Reniers. A hierarchy of communication models for message sequence charts. In *FORTE/PSTV'97 (to appear)*, 1997.

[15] R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89: Proceedings of the IFIP 11th. World Computer Congress*, pages 613–618, San Fransisco, CA, USA, August/September 1989. Elsevier Science Publishers B.V., North-Holland, 1989.

[16] K. Gostellow, V. Cerf, G. Estrin, and S. Volansky. Proper Termination of Flow-of-control in Programs Involving Concurrent Processes. *ACM Sigplan*, 7(11):15–27, 1972.

[17] J. Grabowski, P. Graubmann, and E. Rudolph. Towards a Petri net based sematics definition for Message Sequence Charts. In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects, Proceedings of the sixth SDL Forum*, pages 179–190. North-Holland, 1993.

[18] M.H.T. Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2:77–95, 1976.

[19] D. Hauschildt, H.M.W. Verbeek, and W.M.P. van der Aalst. WOFLAN: a Petri-net-based Workflow Analyzer. Computing Science Reports 97/12, Eindhoven University of Technology, Eindhoven, 1997.

[20] K. Hayes and K. Lavery. Workflow management software: the business opportunity. Technical report, Ovum Ltd, London, 1991.

[21] ITU-TS. ITU-TS Recomendation Z.120: Message Sequence Chart 1996 (MSC96). Technical report, ITU-TS, Geneva, 1996.

[22] R. Kalakota and A.B. Whinston. *Frontiers of Electronic Commerce*. Addison-Wesley, Reading, Massachusetts, 1996.

[23] T.M. Koulopoulos. *The Workflow Imperative*. Van Nostrand Reinhold, New York, 1995.

[24] P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.

[25] S. Mauw and M.A. Reniers. An algebraic semantics of Basic Message Sequence Charts. *The Computer Journal*, 37(4):269–277, 1994.

[26] 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.

[27] 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.

[28] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[29] E. Rudolph, J. Grabowski, and P. Graubmann. Tutorial on Message Sequence Charts. *Computer Networks and ISDN Systems*, 28(12):1629–1641, 1996.

[30] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

[31] M. Wolf and U. Reimer, editors. *Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM'96), Workshop on Adaptive Workflow*, Basel, Switzerland, Oct 1996.