Advance Access publication on December 4, 2008 doi:10.1093/comjnl/bxn064

Multiparty Contracts: Agreeing and Implementing Interorganizational Processes

Wil M.P. van der Aalst[1], Niels Lohmann[2], Peter Massuthe[1,3], Christian Stahl[1,3,*] and Karsten Wolf[2]

[1]*Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands*
[2]*Universität Rostock, Institut für Informatik, 18051 Rostock, Germany*
[3]*Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, Germany*
**Corresponding author: stahl@informatik.hu-berlin.de*

To implement an interorganizational process between different enterprizes, one needs to agree on the 'rules of engagement'. These can be specified in terms of a contract that describes the overall intended process and the duties of all parties involved. We propose to use such a process-oriented contract which can be seen as the composition of the public views of all participating parties. Based on this contract, each party may locally implement its part of the contract such that the implementation (the private view) agrees on the contract. In this paper, we propose a formal notion for such process-oriented contracts and give a criterion for accordance between a private view and its public view. The public view of a party can be substituted by a private view if and only if the private view accords with the public view. Using the notion of accordance, the overall implemented process is guaranteed to be deadlock-free and it is always possible to terminate properly. In addition, we present a technique for automatically checking our accordance criterion. A case study illustrates how our proposed approach can be used in practice.

Keywords: Petri nets; service contracts; operating guidelines

Received 18 February 2008; revised 31 July 2008
Handling editor: Franco Zambonelli

1. INTRODUCTION

During the last years, interorganizational cooperation between enterprizes distributed all over the world has become increasingly important. To meet the challenges of ever-changing markets, IT systems must provide high flexibility and allow for rapid change. *Service-oriented computing* (SOC) [1, 2] has become the new computing paradigm. A service serves as a building block for designing flexible business processes by composing multiple services. That way, SOC—and in particular the use of web services—reduce the complexity of integrating business processes within and across organizational boundaries. *Service-oriented architectures* serve as an enabler for publishing services via the internet such that these services can be automatically found. By dynamically binding published services with services of other enterprizes, interorganizational cooperation can be achieved. The process of publishing, finding and dynamically binding of services is known as *service brokering*.

Although there are promising approaches, service brokering for interorganizational processes has not become accepted in practice, mainly because there is no accepted standard that can handle syntax, semantics, behavior and quality of services (QoS). An additional limiting factor is that enterprizes usually cooperate only with enterprizes they already know.

Therefore, instead of dynamically binding services using service brokering, interorganizational cooperation between enterprizes is realized by specifying an abstract description of the overall interorganizational process, the *choreography*. The choreography serves as a common *contract* between the parties involved in the overall process, which is the focus of this paper. This contract has the form of an agreed upon process model, similar to [3, 4]. Examples of choreography languages are Web Services Choreography Description Language (WS-CDL) [5], Let's dance [6] and BPEL4Chor [7].

The challenge of the contract approach is to balance the following two conflicting requirements. 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 their services at any point in time. Furthermore, the organizations involved in the cooperation do not want to reveal their trade secrets. Therefore, we propose to use a process-oriented contract that defines 'rules of engagement' without describing the internal processes executed within each party [3, 4].

After having specified the contract, each party will implement its part of the contract (its *public view*) on its own. The implemented version, the *private view*, will usually deviate significantly from its public view. Obviously, these local modifications have to accord to the agreed contract. This is, in fact, a nontrivial task, because it may cause global errors such as deadlocks, as shown in [4], for instance. Because all parties are autonomous, none of them owns the overall contract implementation. Therefore, none of the parties can verify this implementation or verify whether its private view behaves correctly within the contract implementation. As a result, an approach is needed such that each party can locally check whether its implementation guarantees global correctness of the overall process.

Figure 1 illustrates the basic idea. The starting point is a contract partitioned over the four parties involved. The public view of each of the four parties corresponds to a fragment of the contract. Based on the public view, each party will implement its private view. Hence, the actual implementation of the overall process consists of the four private views glued together (as shown in the top-right corner of Fig. 1). We will show that it is possible to ensure that the implementation is free of deadlocks and able to terminate while only locally checking correctness and without restricting the private views unnecessarily.

In this paper, we propose a formalization of service contracts taking the above restrictions into account. The contribution of this paper is 3-fold.

First, we present a notion of a contract specifying the interplay of the participating parties. A party's share of the interaction can be seen as a service and the corresponding part of the contract as the public view of the service. The interaction between services is modeled by asynchronous message passing, thus conforming to the idea of SOC. Each public view is modeled as an *open net* [8]. Open nets, also known as open workflow nets (oWFNs), are a special class of Petri nets [9] that has been proven to be an adequate service model. The composition of the public view open nets is an open net modeling the contract.

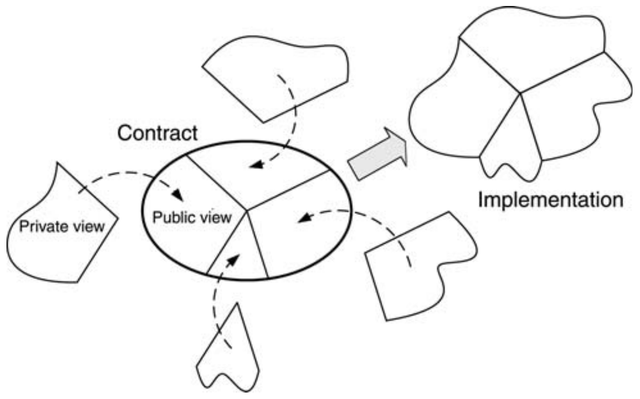


FIGURE 1. If each of the private views accords with its public view, then the resulting overall implementation is 'correct'.

Secondly, we define a notion of *accordance* between the private view of a party, which is represented as an open net as well, with the public view. An open net N' *accords* with an open net N if it has the same (static) interface and any environment that can cooperate with N can also cooperate with N'. This way, we can decide whether a private view of a party agrees on a contract just by comparing it with its public view. Thus, the value of our accordance notion is that it gives a criterion that can be locally verified for asserting global correctness of the overall contract.

Thirdly, we show how accordance can be automatically decided for open nets with acyclic behavior by using the concept of an *operating guideline* [10] of a service as an abstract representation of all environments that the service can cooperate with.

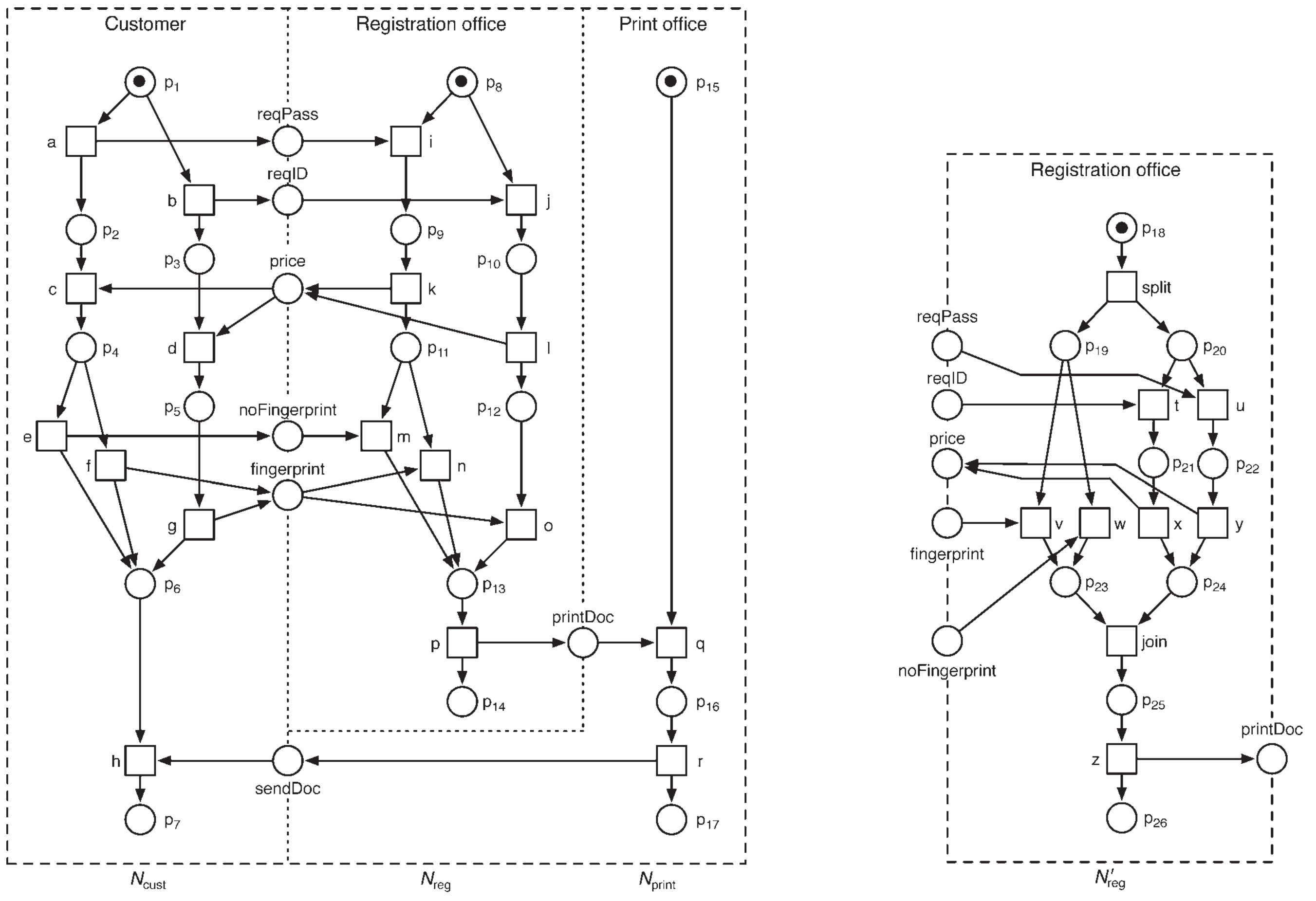
To justify the practical applicability of our approach, we translate a real-life BPEL4Chor choreography into our formal model, yielding a contract. The public view of a party of the derived contract is then manually refined into a private view and we decide if this private view is a correct implementation of the party's public view by the help of the operating guidelines of both models.

The remainder of this paper is structured as follows. We start explaining our approach by using an example contract in Section 2. Then, Section 3 defines open nets and contracts. The notion of accordance is defined in Section 4 where it is applied to relate the private and public views of a contract. Section 5 presents an algorithm to decide accordance using operating guidelines. In Section 6, we show how our proposed approach can be used in practice. Related work is presented in Section 7. Finally, Section 8 concludes the paper.

2. EXAMPLE

In this section, we introduce our running example of a contract organizing the registration process for a passport or an ID card. Its open net model is depicted in Fig. 2a. Open nets are a special class of Petri nets. Before describing the contract, we shortly introduce the main concepts of Petri nets and open nets.

Petri nets have two kinds of nodes, *places* and *transitions*, which are connected by a flow relation. Graphically, a place is represented by a circle, a transition by a box and the flow relation by directed arcs between them. Although transitions



(a) A contract between a customer, a registration office, and a print office.

(b) A private view of the registration office.

FIGURE 2. An example contract for an ID card or passport registration (**a**) and a possible implementation of the registration office (**b**).

represent dynamic elements, for example, an activity in a process, places represent static elements, such as causality between activities or a message channel. A *state* of the Petri net is represented by a *marking* which is a distribution of tokens (graphically represented by black dots) over a (sub-)set of places. The dynamics of a Petri net is defined by its markings and the *firing rule*. The firing rule defines enabledness of Petri net transitions and their effects. A transition, say t, is enabled if there is a token on every place in its pre-set (i.e. those places that have an arc to t). The firing of an enabled transition t yields in a new marking, which is derived from its predecessor marking by consuming (i.e. removing) a token from each place of t's pre-set and producing (i.e. adding) a token on each place of t's post-set (i.e. those places p with an arc from t to p).

Open nets are a special class of Petri nets that extend classical Petri nets by an interface for communication with other open nets. The net of Fig. 2a modeling the contract is an example for an open net. For the moment, ignore the dotted lines separating Fig. 2a into three parts. Graphically, an open net is framed by a dashed line with its interface places depicted on this line. Since our contract has an empty interface, *no* places are situated on the outer frame in Fig. 2a.

Three *parties* are involved in our contract: a customer who wants to register at the registration office for either a passport or an ID card; a registration office that receives the form from the customer, stores his fingerprint (if needed) and forwards all information to the print office and a print office that prints the document and sends it to the customer. The three parties are modeled by the three open nets N_{cust}, N_{reg} and N_{print}, respectively, which are highlighted in Fig. 2a by the dotted lines inside the contract. The respective interface places are situated along the dotted lines.

The contract in Fig. 2a organizes the registration for a passport or an ID card as follows. Initially, the customer either sends the registration form for a passport (transition a) or an ID card (transition b) to the registration office. This is modeled by an *explicit choice* of transitions a and b; that is, both transitions are enabled by the token on place p_1 but only one of them can fire. If transition a fires, the token on p_1 is consumed and a token is produced on the two places p_2 and reqPass. Firing transition b results in consuming the token on p_1 and producing a token on the places p_3 and reqID.

In both markings, the customer waits for a price message (i.e. a token on the place price) which is sent by the registration office. Next, the registration office can receive the respective form. In case the customer requested for a passport (respectively, for an ID card) there is a token on the place reqPass (respectively, reqID) and thus enabling the transition i (respectively, j). Both transitions model a *deferred choice*. Firing of transition i (respectively, j) models the receiving of the respective form and results in consuming the token from the interface place p_8 and yielding a token on p_9 (respectively, p_{10}). In either way, the registration office returns a message containing the price of the requested document to the customer (transitions k and l, respectively).

Afterwards, the customer receives the price message (transitions c and d, respectively). In case the customer requested for a passport, he can decide whether his biometrical data (i.e. a fingerprint) is stored (transition f) or not (transition e). For an ID card, however, a fingerprint is mandatory (transition g). The registration office receives the data (by firing transitions m, n and o, respectively). After having received the form and the biometrical data, the registration office forwards all information to the print office (transition p). The print office receives this information (transition q), prints the requested document and sends it directly to the customer (transition r). Finally, the customer receives his document (transition h).

The three open nets N_{cust}, N_{reg} and N_{print} of Fig. 2a specify the *public view* of each of the parties involved. In contrast to the overall open net of the contract, each such open net *has* interface places. For example, N_{print} has one input place (printDoc) and one output place (sendDoc). N_{cust} has (among others) the input place sendDoc and the open net N_{reg} has (among others) the output place printDoc. By merging these interface places, the public views can be plugged together into the global contract. For example, the output place printDoc of N_{reg} is merged with the input place printDoc of N_{print} and the input place sendDoc of N_{cust} is merged with the output place sendDoc of N_{print}.

After the parties agreed on such a contract, each of them now may implement its public view. The resulting private view may again be expressed as an open net. Figure 2b shows an example for a private view of the registration office; that is, the open net N'_{reg} is an implementation of the open net N_{reg}. Within an reorganization, the registration office is divided into two departments. One department is responsible for storing the biometrical data (transitions v and w) and the other department has to receive and to check the forms (transitions t and u) and to send the pricing information (transitions x and y). The service that each department offers can be used independently from the other department. That way, the office hopes to serve the customer faster and thus reduce costs. As a result, in the private view in Fig. 2b both departments run concurrently. This is modeled by transitions split and join. Firing of transition split yields a token on places p_{19} and p_{20}. Then, both branches can be executed independently of each other, finally yielding a marking where places p_{23} and p_{24} have a token. Firing transition join then synchronizes both branches.

For the sake of simplicity, this example is rather atypical in the sense that an implementation usually tends to have much more internal tasks than the public view. However, N'_{reg} allows for behavior not possible in N_{reg}, viz. a fingerprint can be stored before the registration form is received. That is, in N_{reg} transition o can never fire before transition j but in N'_{reg} transition v can fire before transition t.

The question we have to answer is *whether the public view of the* registration office *can be substituted by its private view while still guaranteeing correctness of the contract*. To this end, we will formalize our model for service contracts and provide an *accordance notion* between public and private views in the next sections.

It is important to note that we consider only a *single instance* of the registration process. More precisely, in Fig. 2a, there are three tokens in the three source places p_1, p_8 and p_{15}. It is assumed that these tokens correspond to the same instance. Therefore, we do not need to correlate tokens on the interface places to tokens in the local processes. If there were multiple instances, this could be handled separately (e.g. using colored tokens or a private open net for each instance). We acknowledge that multiple instances are an important topic, but in this paper we assume that correlation is taken care of elsewhere, for example, by using the correlation mechanisms in the context of Web Services Business Process Execution Language (WS-BPEL) [11]. Hence, we only need to consider a single instance in isolation.

3. FORMALIZING CONTRACTS

To design an overall interorganizational process, the involved parties specify a public view of this process together with a distribution of responsibilities for the activities among them (i.e. a partitioning). The global public view and its partitioning serve as a *contract*.

For specifying the overall contract as well as the processes it consists of, we use the concept of *open nets* [8]. Open nets are a more liberal notion of the well-known concept of *workflow nets* (WFNs) [12], which have been proven successful for the modeling of business processes and workflows. As a substantial difference, open nets introduce interface places to communicate with other open nets. This idea is based on the module concept for Petri nets which was proposed by Kindler [13]. However, we consider the overall process (i.e. the contract) as being *self-contained*. This fact is reflected by its representation as an open net *with empty interface*. An immediate alternative to our approach would be to represent the contract itself within the much more established framework of WFNs. However, by using open nets only, we avoid duplicate definitions and some of the technicalities related to having unique source and sink places.

The basis of open nets are classical (place/transition) Petri nets (see [9], for instance). A Petri net $N = (P, T, F)$ consists of two finite sets P and T of places and transitions and a flow relation $F \subseteq (P \times T) \cup (T \times P)$. We use the standard notation to denote the pre- and postsets of a place or a transition: ${}^{\bullet}x = \{y \mid (y, x) \in F\}$ and $x^{\bullet} = \{y \mid (x, y) \in F\}$.

The *behavior* of a Petri net is defined using the standard Petri net semantics [9]: a transition is enabled if each place of its pre-set holds a token; an enabled transition t can fire in a marking m by consuming tokens from the pre-set places and producing tokens for the post-set places, yielding a marking m'. The firing of t is denoted by $m \xrightarrow{t} m'$, the successively firing of a sequence of transitions is denoted by $m \xrightarrow{*} m'$.

Definition 3.1 (Open net). An *open net* $N = (P, T, F, I, O, m_0, \Omega)$ *consists of a Petri net* (P, T, F) *together with*

(i) *an interface defined as a set* $I \subseteq P$ *of input places such that* ${}^{\bullet}p = \emptyset$ for any $p \in I$ and a set $O \subseteq P$ of *output places such that* $p^{\bullet} = \emptyset$ for any $p \in O$ and $I \cap O = \emptyset$,
(ii) *a distinguished initial marking* m_0 and
(iii) *a set* Ω *of final markings such that no transition of* N *is enabled at any* $m \in \Omega$.

We further require that $m \in \Omega \cup \{m_0\}$ *implies* $m(p) = 0$ *for all* $p \in I \cup O$; *that is, in the initial and the final markings, the interface places are not marked.*

Our formalism of open nets is quite liberal and serves as a model for the overall process of the contract as well as for the processes of the parties participating in the contract. All these processes are assumed to have a control structure with a unique starting point (modeled by the corresponding initial marking) and one or more final, i.e. terminal, states (modeled by the set of final markings).

Communication between the party's processes is achieved by asynchronous message passing via message channels (modeled by the interface places). Asynchronous message passing means that after a party has sent a message it can continue its execution and does not have to wait until this message is received. Furthermore, messages can 'overtake' each other; that is, the order in which the messages are sent is not necessarily the order in which they are received.

In order to assign an intuitively consistent meaning to *final* states, we restrict our approach to such open nets where a final marking in Ω does not enable any transition.

We use indices to distinguish the constituents of different open nets (e.g. I_j refers to the set of input places of open net N_j).

As an example, the whole process shown in Fig. 2a represents an open net N with empty interface $I = O = \emptyset$. N has the initial marking $m_0 = [\mathsf{p}_1, \mathsf{p}_8, \mathsf{p}_{15}]$ (i.e. the marking shown in Fig. 2a) and we define $\Omega = \{[\mathsf{p}_7, \mathsf{p}_{14}, \mathsf{p}_{17}]\}$ (i.e. there is only one final marking and this marking marks all sink places). The final marking mirrors the fact that all parties have been completed. N_{print}, also shown in Fig. 2a, is an open net with nonempty interface $I_{\mathsf{print}} = \{\mathsf{printDoc}\}$ and $O_{\mathsf{print}} = \{\mathsf{sendDoc}\}$.

It is easy to check in the open net N of Fig. 2a that from any marking reachable from the initial marking, the final marking $[\mathsf{p}_7, \mathsf{p}_{14}, \mathsf{p}_{17}]$ is reachable. This means that it is always possible to terminate properly. This property is formalized in the following definition.

Definition 3.2 (Weak termination). *An open net* $N = (P, T, F, I, O, m_0, \Omega)$ *weakly terminates iff for each* m *with* $m_0 \xrightarrow{*} m$ there is an $m_f \in \Omega$ with $m \xrightarrow{*} m_f$.

Weak termination of an open net N ensures that N is *free of deadlocks* and *free of livelocks*. Deadlock-freedom means that each nonfinal marking has at least one successor, whereas livelock-freedom means that each cycle of nonfinal markings can be left to reach a final marking.

The open net N representing the contract of Fig. 2a is *composed* of multiple open nets representing the parties. For the composition of open nets, we assume that all constituents (except for the interfaces) are pairwise disjoint. This can be achieved easily by renaming. In contrast, the interfaces intentionally overlap. For a reasonable concept of composition of open nets it is, however, convenient to require that all communication is bilateral and directed; that is, every interface place $p \in I \cup O$ has only one party that sends into p and one party that receives from p. For a third party C, a communication taking place inside the composition of parties A and B is internal matter. We will refer to open nets that fulfill these properties as *composable*. Composition is only defined for composable open nets. These considerations lead to the following definition of composition.

Definition 3.3 (Composition of open nets). *Let* $N_1, \ldots, N_k$ *be open nets with pairwise disjoint constituents, except for the interfaces.* $N_1, \ldots, N_k$ *are composable if, for all* $i \in \{1, \ldots, k\}$,

(i) $p \in I_i$ *implies that there is no* $j \neq i$ *such that* $p \in I_j$ *and there is at most one* j *such that* $p \in O_j$, *and*
(ii) $p \in O_i$ *implies that there is no* $j \neq i$ *such that* $p \in O_j$ *and there is at most one* j *such that* $p \in I_j$.

For markings $m_1 \in N_1, \ldots, m_k \in N_k$ *which do not mark interface places, their composition* $m = m_1 \oplus \cdots \oplus m_k$ *is defined by* $m(p) = m_i(p)$ *if* $p \in P_i$.

If $N_1, \ldots, N_k$ *are composable, their composition is the open net* $N = N_1 \oplus \cdots \oplus N_k$ *defined as follows*:

(i) $P = P_1 \cup \cdots \cup P_k$,
(ii) $T = T_1 \cup \cdots \cup T_k$,
(iii) $F = F_1 \cup \cdots \cup F_k$,
(iv) $I = (I_1 \cup \cdots \cup I_k) \setminus (O_1 \cup \cdots \cup O_k)$,

(v) $O = (O_1 \cup \cdots \cup O_k) \setminus (I_1 \cup \cdots \cup I_k)$,
(vi) $m_0 = m_{0_1} \oplus \cdots \oplus m_{0_k}$ *and*
(vii) $\Omega = \{m_1 \oplus \cdots \oplus m_k | m_1 \in \Omega_1, \ldots, m_k \in \Omega_k\}$.

Composition of two open nets M and N results in an open net again. Composing M and N means merging input places of M with equally labeled output places of N (and vice versa). Therein, bilateral and directed communication between the components is guaranteed. The initial marking of the composition is the sum of the initial markings of the components, and the set of final markings of the composition is the Cartesian product of the sets of final markings of the components. This is reasonable because Definition 3.1 ensures that the only shared constituents of the components (i.e. the interface places) are not marked in the initial or final markings.

Consider again the three open nets N_{cust}, N_{reg}, and N_{print} of Fig. 2a. We define the sets of final markings of these nets as $\{[\mathsf{p}_7]\}$, $\{[\mathsf{p}_{14}]\}$ and $\{[\mathsf{p}_{17}]\}$, respectively. Clearly, these open nets are composable: $N = N_{\mathsf{cust}} \oplus N_{\mathsf{reg}} \oplus N_{\mathsf{print}}$ is the overall process shown in Fig. 2a. It is easy to see that the composition has the final marking $[\mathsf{p}_7, \mathsf{p}_{14}, \mathsf{p}_{17}]$.

Note that any subset of a set of composable open nets is composable as well. Furthermore, we have $N_1 \oplus N_2 \oplus N_3 = (N_1 \oplus N_2) \oplus N_3 = N_1 \oplus (N_2 \oplus N_3)$, and $N_1 \oplus N_2 = N_2 \oplus N_1$; that is, the composition of open nets is associative and commutative. Thus, composition of a set of open nets can be broken into single steps without affecting the final result.

Basically, we see a *contract* as an open net where every activity is assigned to one of the involved parties. We impose only one restriction: if a place is accessed by more than one party, it should act as a directed bilateral communication place. In the following, $|X|$ denotes the cardinality of a set X.

Definition 3.4 (Contract). *Let $\mathcal{A}$ be a set representing the parties involved in a contract.*

Then, a contract $[N, r]$ consists of an open net $N = (P, T, F, I, O, m_0, \Omega)$ with an empty interface ($I = O = \emptyset$) (the agreed public view of the process) and a mapping $r : T \to \mathcal{A}$ (the partitioning) such that, for all places $p \in P$, $|\{r(t) \mid t \in {}^\bullet p\}| \leq 1$ and $|\{r(t) \mid t \in p^\bullet\}| \leq 1$.

For technical purposes, we further require that the set of final markings Ω is Cartesian closed. That is, for two final markings $m_1, m_2 \in \Omega$ and a party $A \in \mathcal{A}$, the marking m_{12} with $m_{12}(p) = m_1(p)$ if $r(p) = A$ and $m_{12}(p) = m_2(p)$ if $r(p) \neq A$ is also a final marking (i.e. $m_{12} \in \Omega$). We also require that $m \in \Omega \cup \{m_0\}$ implies $m(p) = 0$ for all $p \in P$ such that $\exists t_1 \in {}^\bullet p : \exists t_2 \in p^\bullet : r(t_1) \neq r(t_2)$; that is, in the initial and the final markings, the internal interface places are not marked.

The overall open net N shown in Fig. 2a is an example for a contract involving the parties $\mathcal{A} = \{$customer, registration office, print office$\}$. The dotted lines in the figure show the partitioning of transitions over the parties involved in the contract; $r(\mathsf{a}) =$ customer, $r(\mathsf{k}) =$ registration office and $r(\mathsf{q}) =$ print office, for instance.

A contract can be cut into parts, each representing the agreed share of a single party. If $[N, r]$ is a contract of parties $\mathcal{A} = \{A_1, \ldots, A_k\}$, these parts are modeled by the open nets $N_{A_1}, \ldots, N_{A_k}$. For this purpose, the definition of a contract guarantees that every part is an open net again (satisfying the properties for initial and final markings), this time typically with a nonempty interface. It is obviously desirable that recomposing the parts afterwards, the resulting composed open net is the net N of the contract again. In this respect, the restriction that the set of final markings Ω of a contract is Cartesian closed indeed crucial, as otherwise $N_{A_1} \oplus \ldots \oplus N_{A_k}$ could have a final marking that results from recombining final markings of different parties but which is not a final marking of N. Therefore, the contract definition ensures that the composition of a contract's parts is the contract itself again.

In accordance with terminology of SOC [1, 2], we consider the contribution of a party to an interorganizational process as a *service*. Correspondingly, the agreed version (specification) of the service is called *public view* while an actual local implementation is called *private view* of the service.

Definition 3.5 (Public view). *Let $[N, r]$ be a contract with $N = (P, T, F, I, O, m_0, \Omega)$ and $r : T \to \mathcal{A}$, and let $A \in \mathcal{A}$ be a party.*

The public view of A's share in the contract is the open net $N_A = (P_A, T_A, F_A, I_A, O_A, m_{0_A}, \Omega_A)$ where

(i) $P_A = \{p \in P \mid \exists t \in {}^\bullet p \cup p^\bullet : r(t) = A\}$,
(ii) $T_A = \{t \in T \mid r(t) = A\}$,
(iii) $F_A = F \cap ((P_A \times T_A) \cup (T_A \times P_A))$,
(iv) $I_A = \{p \in P_A \mid \exists t \in {}^\bullet p : r(t) \neq A\}$,
(v) $O_A = \{p \in P_A \mid \exists t \in p^\bullet : r(t) \neq A\}$,
(vi) $m_{0_A} = m_{0|_{P_A}}$ (*i.e. the restriction of m_0 to the places in P_A*) *and*
(vii) $\Omega_A = \{m_{f|_{P_A}} | m_f \in \Omega\}$.

Informally spoken, a party's public view in a contract is the restriction of the open net modeling the contract to the constituents of this party. The open net modeling this public view is visualized by the dotted frame in Fig. 2. Technically, it is characterized by the set of transitions assigned to the respective party, the set of places in its pre- and postsets and the restriction of the flow relation, the initial and final markings to these places and transitions.

As described earlier, Fig. 2a shows the public views N_{cust}, N_{reg}, and N_{print} of the parties customer, registration office and print office, respectively.

A *private view* can now be seen as the implemented version N' of a public view N. Technically, N' is an (arbitrary) open net that has the same interface as N.

Figure 2b depicts a possible private view N'_{reg} of the registration office. N'_{reg} has the same interface as N_{reg} in Fig. 2a. N'_{reg} has one final marking, $[\mathsf{p}_{26}]$. Private views of the other parties are not provided here.

The next section is devoted to the question, whether a private view is a *correct* implementation of a public view, for example, can N_{reg} be replaced by N'_{reg} without making additional agreements?

4. ACCORDANCE BETWEEN PUBLIC AND PRIVATE VIEW

In this section, we define the notion of *accordance*. This criterion is used to compare the public view (agreed specification of the service) and the private view (actual implementation of the service) on a party's share of a contract. The goal of the accordance notion is to preserve weak termination (see Definition 3.2) of the overall process N. Formally, accordance is a *refinement relation* where weak termination of N and accordance of each private view N'_{A_i} with the corresponding public view N_{A_i} should imply weak termination of $N'_{A_1} \oplus \cdots \oplus N'_{A_k}$ which obviously models the overall process as actually implemented.

Consider, for example, the weakly terminating contract shown in Fig. 2a which is split into the three public views N_{cust}, N_{reg} and N_{print}. If each of these public views is substituted by a private view such that the private view *accords* with the public view, then the composition $N'_{\mathsf{cust}} \oplus N'_{\mathsf{reg}} \oplus N'_{\mathsf{print}}$ of these private views should weakly terminate, too.

To define a suitable notion of accordance, we introduce the concept of *strategies*.

Definition 4.1 (Strategy). *An open net M is a strategy for an open net N if $M \oplus N$ is weakly terminating. Strat(N) denotes the set of all strategies for N.*

If an open net M is a strategy for an open net N, then M and N together are 'well-behaving' with respect to weak termination. That is, the strategy notion allows to distinguish, for a given open net N, all those open nets M that are feasible as a partner in the composition with N from those open nets that introduce an error in the composition. Note that *Strat*(N) may correspond to a large (in fact infinite) set of open nets. Furthermore, the notion of a strategy is symmetric; that is, if M is a strategy for N, then N is also a strategy for M and vice versa.

The open nets N_{reg} in Fig. 2a and N'_{reg} in Fig. 2b are two example strategies for $N_{\mathsf{cust}} \oplus N_{\mathsf{print}}$.

In general, given a weakly terminating contract $[N, r]$ with parties $\mathcal{A} = \{A_1, \ldots, A_k\}$, each party's public view N_{A_i} is a strategy for the remaining part of the composition $N_{A_1} \oplus \cdots \oplus N_{A_{i-1}} \oplus N_{A_{i+1}} \oplus \cdots \oplus N_{A_k}$. This property of the strategy concept justifies the following definition of accordance.

Definition 4.2 (Accordance). *An open net N' accords with an open net N if N' has the same interface as N (i.e. $I' = I$ and $O' = O$) and N' has at least the strategies that N has, i.e. Strat(N') $\supseteq$ Strat(N).*

Accordance defines a refinement relation between open nets: if an open net N' accords with an open net N, then N' has at least the strategies of N. Therefore, accordance is well-suited for the scenario of implementing public views in a contract.

Note that Definition 4.2 defines accordance for arbitrary open nets N and N'. If N is a public view and N' is the corresponding private view, then interface equality is trivially fulfilled.

In our example, N_{reg} is the public view of the registration office and N'_{reg} is the corresponding private view. For this simple example, it is easy to see that N'_{reg} accords with N_{reg}. Hence, N'_{reg} can be used instead of N_{reg} in the contract while guaranteeing weak termination of the overall process.

It is important to note that accordance is a much weaker notion than most notions described in the literature (e.g. projection inheritance [14, 15]). Accordance exploits the fact that communication is asynchronous and, therefore, a weaker notion suffices.

To illustrate our accordance relation, consider Fig. 3 that shows two fragments of two open nets N_1 and N_2. Assume that the only connections between the fragments and the rest of the open net are through places R and S; that is, the ovals correspond to placeholders for sets of places, connecting the fragment to the rest of the net. Assume that N_1 and N_2 are identical except for the fragments shown. In this case, *Strat*(N_1) = *Strat*(N_2); that is, any environment that can cooperate with N_1 can cooperate with N_2 and vice versa. In other words, N_1 accords with N_2 and vice versa. This is, however, nontrivial, because N_2 can receive a message via b before sending a message via a. This is not the case in N_1. Figure 4 shows another example illustrating the notion of accordance. In an

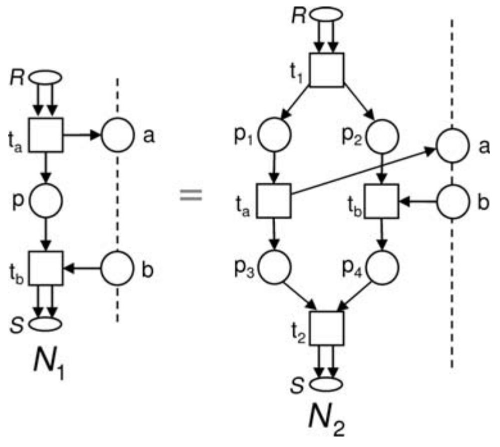


FIGURE 3. N_2 accords with N_1 and vice versa.

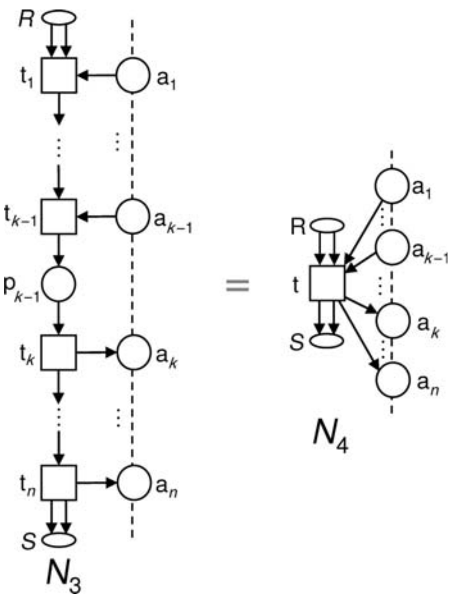


FIGURE 4. N_4 accords with N_3 and vice versa.

open net, the left-hand-side fragment can be substituted by the right-hand-side fragment without jeopardizing correctness; that is, N_3 accords with N_4 and vice versa. Generally, a sequence of receiving events followed by a sequence of sending events can be executed simultaneously while preserving accordance in both directions, i.e. $Strat(N_3) = Strat(N_4)$. This shows that sending and receiving events can be ordered arbitrarily as long as the receiving events proceed the sending events. Both Figs. 3 and 4 show that the asynchronous nature of communication allows partners to implement the private view in a way that deviates from the public view without creating problems such as deadlocks or inability to terminate. This makes accordance an original and interesting equivalence notion in the context of SOC.

The following theorem shows that *each* party can substitute its public view by a private view *independently*. If each of the private views accords with the corresponding public view, then weak termination of the implemented contract is guaranteed.

THEOREM 4.3 (IMPLEMENTATION OF A CONTRACT). *Let* $[N, r]$ *be a contract between parties* $\{A_1, \ldots, A_k\}$ *where* N *is weakly terminating. If, for all* $i \in \{1, \ldots, k\}$, N'_{A_i} *(the private view of* A_i*) accords with* N_{A_i} *(the public view of* A_i*), then* $N' = N'_{A_1} \oplus \cdots \oplus N'_{A_k}$ *(the actual implementation) is weakly terminating.*

Proof. Let $\{A_1, \ldots, A_k\}$ be the set of involved parties and $\mathcal{N}(j) = N'_{A_1} \oplus \cdots \oplus N'_{A_j} \oplus N_{A_{j+1}} \oplus \cdots \oplus N_{A_k}$ for $j \in \{0, \ldots, k\}$. Note that $\mathcal{N}(0) = N_{A_1} \oplus \cdots \oplus N_{A_k} = N$ and $\mathcal{N}(k) = N'_{A_1} \oplus \cdots \oplus N'_{A_k} = N'$.

We show by induction that $\mathcal{N}(j)$ is weakly terminating for any $j \in \{0, \ldots, k\}$.

Clearly, this holds for $j = 0$: $\mathcal{N}(0) = N$ is weakly terminating. Assume that $\mathcal{N}(j)$ is weakly terminating and $0 \leq j < k$. Let $\mathcal{N}' = N'_{A_1} \oplus \cdots \oplus N'_{A_j} \oplus N_{A_{j+2}} \oplus \cdots \oplus N_{A_k}$; that is, $\mathcal{N}(j)$ without $N_{A_{j+1}}$. $\mathcal{N}'$ is a strategy for $N_{A_{j+1}}$, because $\mathcal{N}(j) = \mathcal{N}' \oplus N_{A_{j+1}}$ is weakly terminating; that is, $\mathcal{N}' \in Strat(N_{A_{j+1}})$. Because $N'_{A_{j+1}}$ (the private view) accords with $N_{A_{j+1}}$ (the public view), $Strat(N_{A_{j+1}}) \subseteq Strat(N'_{A_{j+1}})$. Hence, $\mathcal{N}' \in Strat(N_{A_{j+1}}) \subseteq Strat(N'_{A_{j+1}})$, indicating that $\mathcal{N}'$ is a strategy for $N'_{A_{j+1}}$. Therefore, $\mathcal{N}' \oplus N'_{A_{j+1}} = \mathcal{N}(j + 1)$ is weakly terminating. By induction this implies that $\mathcal{N}(j)$ is weakly terminating for any j including $j = k$. Hence, $N' = \mathcal{N}(k)$ is weakly terminating. □

The value of the theorem is that it gives each party a criterion (accordance of N'_{A_i} with N_{A_i}) that can be locally verified for asserting a global property (weak termination of the overall process as actually implemented).

For example, any combination of arbitrary private views N''_{cust}, N''_{reg} and N''_{print} according with the corresponding public view (i.e. N''_{cust} accords with N_{cust}, N''_{reg} accords with N_{reg}, and N''_{print} accords with N_{print}) yields a weakly terminating realization of the contract shown in Fig. 2a.

5. CHECKING ACCORDANCE

In this section, we demonstrate that the property of *accordance* can be verified automatically, presently subject to restrictions. Checking accordance relies on the concept of an *operating guideline* (OG) [10, 16]. The purpose of an operating guideline OG_N of a service N is to characterize the set of all services M such that the composition of M and N behaves 'correctly'. Thereby, 'correctly' means weak termination in [16] or deadlock-freedom in [10]. The results in [16] are restricted to services with acyclic and finite behavior while there are only marginal restrictions for the approach in [10]. As we are interested in weak termination in this paper, we use the approach in [16] thus inheriting the restriction to acyclic finite-state services. Such a service may have, e.g. *while* loops in its operational description as long as every iteration produces states that are different from other iterations, for example, states with different counter values. In ongoing research, we work on an extension of the approach in [10] to weak termination, so the current restriction to acyclic finite-state services may only be temporary. First results support this assumption.

With the help of OGs we are able to represent the set of all strategies M for an open net N in a compact way. Technically, an OG is a special annotated automaton. Such an annotated automaton A^{Φ} consists of an underlying finite and deterministic automaton A and a function Φ that assigns to each state q of A a negation-free Boolean formula $\Phi(q)$ over transition labels. For instance, $!a \vee (?b \wedge ?c)$ is a valid Boolean formula that is satisfied if $!a$ is set to *true* and/or both $?b$ and $?c$ are set to *true*.

An annotated automaton A^{Φ} is used to represent the set of open nets that *match* with A^{Φ}. We continue by first defining the notions of annotated automata and matching in general and then introducing OGs.

Definition 5.1 (Annotated automaton). $A^{\Phi} = [Q, C, \delta, q_0, \Phi]$ is an *annotated automaton iff Q is a nonempty finite set of states, C is a set of labels,* $\delta \subseteq Q \times C \times Q$ *is a transition relation such that every state* $q \in Q$ *is reachable from* q_0 *via transitive applications of* δ, $q_0 \in Q$ *is the initial state and* Φ *is an annotation function, where, for all* $q \in Q$, $\Phi(q)$ *is a Boolean formula over literals in C.*

Annotated automata very similar to ours have been first published in [17]. The original goal of such an annotated automaton A^{Φ} is to represent *a set of automata*: an automaton B is represented by A^{Φ} iff (1) A (weakly) simulates B and (2) if a state r of B is simulated by a state q of A, then the arcs leaving r—interpreted as an assignment assigning *true* to the corresponding literals of the formula $\Phi(q)$—satisfy $\Phi(q)$.

Intuitively, this means that the underlying automaton A of A^{Φ} has (at most) richer behavior than B (expressed by the simulation relation [18]), but B has behavior at least as rich as required by the Boolean formulae of Φ (expressed by the satisfaction requirement). For more details, we refer to [19], for example.

In this paper, we use annotated automata as a representation of a set of *open nets*. Therefore, we take a service described in terms of an open net M with the interface $I \cup O$ and an annotated automaton A^{Φ} with Boolean formulae over the interface places of M and a special literal *final* (i.e. $C = I \cup O \cup \{final\}$) and define when M *matches* with A^{Φ}.

Intuitively, M matches with A^{Φ} if the *behavior* of M is simulated by A and each marking of M touched by the simulation satisfies the corresponding formula.

In order to simplify the presentation, we assume the open nets under consideration are *normal*. An open net is normal if each transition is connected to at most one interface place. This assumption does, however, not restrict generality as every open net can be transformed into an equivalent normal one [10].

Definition 5.2 (Matching). *Let M be a normal open net and let X be the set of all reachable markings of the Petri net M* obtained by removing all interface places of M. Let* $A^{\Phi} = [Q, C, \delta, q_0, \Phi]$ *be an annotated automaton with* $C = I_M \cup O_M \cup \{final\}$. *Then M matches with* A^{Φ} *iff there is a mapping* $\rho : X \to Q$ *such that the following conditions hold:*

(i) $\rho(m_{0_M}) = q_0$;
(ii) *If t is an internal transition of M (i.e. t is not connected to any interface place),* $m, m' \in X$, *and* $m \xrightarrow{t} m'$, *then* $\rho(m) = \rho(m')$;
(iii) *If t is a receiving transition of M with* $c \in I_M$, $c \in {}^{\bullet}t$, $m, m' \in X$, *and* $(m + [c]) \xrightarrow{t} m'$, *then there is a state* $q' \in Q$ *with* $(\rho(m), c, q') \in \delta$ *and* $\rho(m') = q'$;
(iv) *If t is a sending transition of M with* $c \in O_M$, $c \in t^{\bullet}$, $m, m' \in X$, *and* $m \xrightarrow{t} (m' + [c])$, *then there is a state* $q' \in Q$ *with* $(\rho(m), c, q') \in \delta$ and $\rho(m') = q'$;
(v) *For all* $m \in X$, *at least one of the following properties holds*:
   (a) *An internal transition t is enabled at m; or*
   (b) $\Phi(\rho(m))$ *evaluates to true for the following assignment* β:
      (1) $\beta(c) = true$ *if* $c \in O_M$ *and there is a transition t with* $c \in t^{\bullet}$ *that is enabled at m*;
      (2) $\beta(c) = true$ *if* $c \in I_M$ *and there is a transition t with* $c \in {}^{\bullet}t$ *that is enabled at* $m + [c]$;
      (3) $\beta(c) = true$ *if c = final and* $m \in \Omega_M$;
      (4) $\beta(c) = false$, *otherwise*.

In the above definition, ρ represents the informally described (weak) simulation relation. Items (i)–(iv) in Definition 5.2 define the simulation relation between the markings of M and the states of the annotated automaton A^{Φ}. The assignment used for evaluating an annotation represents transitions t of M that leave the considered marking m of M^*. That way, we assure that M has behavior at most as rich as the underlying automaton A of A^{Φ} but at least as rich as required by the formulae of Φ. The evaluation of the annotation is defined by item (v).

An operating guideline OG_N of an open net N now is a special annotated automaton, such that an open net M matches with OG_N if and only if M is a strategy for N.

Definition 5.3 (Operating guideline). *An annotated automaton is an operating guideline* OG_N *of an open net N iff Strat(N) is exactly the set of all open nets matching with* OG_N.

Together with the matching procedure, an operating guideline OG_N of an open net N characterizes the set *Strat*(N) of strategies for N. Although *Strat*(N) itself is semantically defined, OG_N is an equivalent operational description of *Strat*(N).

In [16], we were able to show that an operating guideline OG_N of an open net N is always finite and presented an algorithm to compute OG_N. The algorithm is implemented in our tool Fiona [20].[1]

The operating guideline $\mathrm{OG}_{N_{\mathsf{reg}}}$ of the open net N_{reg} of Fig. 2a is an annotated automaton that consists of 11 states and 19 transitions and is depicted in Fig. 5. The annotation of the initial state of $\mathrm{OG}_{N_{\mathsf{reg}}}$ is $\mathsf{fingerprint} \vee \mathsf{noFingerprint} \vee \mathsf{reqID} \vee \mathsf{reqPass}$ reflecting the possible choices of a strategy for N_{reg}.

The open net $N_{\mathsf{cust}} \oplus N_{\mathsf{print}}$ in Fig. 2a is an example for a strategy for N_{reg}. It is easy to see that $N_{\mathsf{cust}} \oplus N_{\mathsf{print}}$ matches with $\mathrm{OG}_{N_{\mathsf{reg}}}$. In the matching, the initial marking $[\mathsf{p}_1, \mathsf{p}_{15}]$ of $N_{\mathsf{cust}} \oplus N_{\mathsf{print}}$, for instance, is simulated by the state q_0 (cp. Definition 5.2):

(i) $\rho([\mathsf{p}_1, \mathsf{p}_{15}]) = \mathsf{q}_0$;
(ii) trivially fulfilled;

[1]Available at http://service-technology.org/fiona.

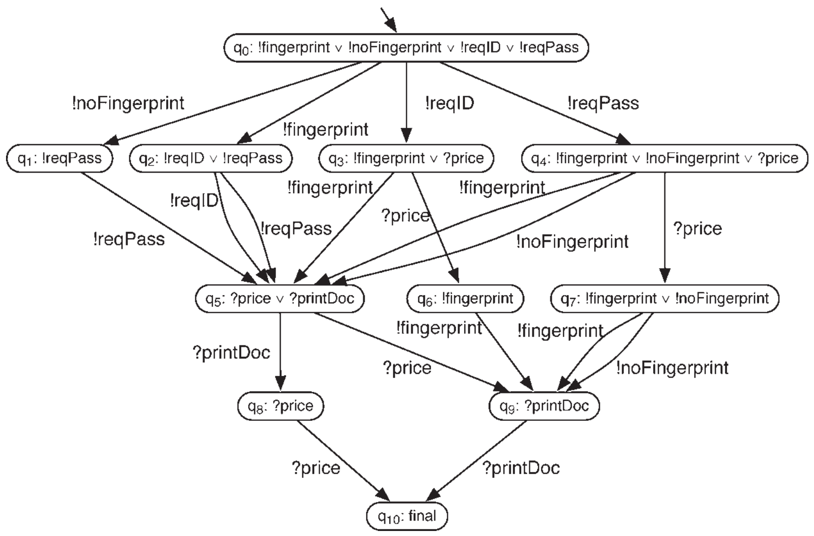


FIGURE 5. The operating guideline $OG_{N_{reg}}$ of the public view of the registration office of Fig. 2a. For better readability, we add a leading '!' ('?') to a literal x in the graphics of an OG_N if x is an output (input) place of a strategy M for N.

(iii) trivially fulfilled;

(iv) $[\mathsf{p}_1, \mathsf{p}_{15}]$ enables the transitions

(a) a with $c = \mathsf{reqPass} \in O_{N_{cust}} \oplus N_{print}$, $\mathsf{reqPass} \in \mathsf{a}^\bullet$ and $[\mathsf{p}_1, \mathsf{p}_{15}] \xrightarrow{\mathsf{a}} [\mathsf{p}_2, \mathsf{p}_{15}, \mathsf{reqPass}]$ and $(\mathsf{q}_0, \mathsf{reqPass}, \mathsf{q}_4) \in \delta_{OG_{N_{reg}}}$, and q_4 simulates the marking $[\mathsf{p}_2, \mathsf{p}_{15}]$

(b) b with $c = \mathsf{reqID} \in O_{N_{cust}} \oplus N_{print}$, $\mathsf{reqID} \in \mathsf{b}^\bullet$ and $[\mathsf{p}_1, \mathsf{p}_{15}] \xrightarrow{\mathsf{b}} [\mathsf{p}_3, \mathsf{p}_{15}, \mathsf{reqID}]$ and $(\mathsf{q}_0, \mathsf{reqID}, \mathsf{q}_3) \in \delta_{OG_{N_{reg}}}$, and q_3 simulates the marking $[\mathsf{p}_3, \mathsf{p}_{15}]$;

(v) the formula $\Phi(\mathsf{q}_0) = \mathsf{fingerprint} \vee \mathsf{noFingerprint} \vee \mathsf{reqID} \vee \mathsf{reqPass}$ is satisfied by the assignment β that assigns *true* to the literals reqPass (because of transition a) and reqID (because of transition b) and *false* to all other literals.

As an OG of an open net N represents the set of strategies for N, it is natural to use OG_N and $OG_{N'}$ for comparing $Strat(N)$ with $Strat(N')$. Informally, N' accords with N if and only if OG_N can be embedded into $OG_{N'}$ such that the annotations in OG_N imply the annotations of $OG_{N'}$.

THEOREM 5.4 (ACCORDANCE CHECK WITH OGs) *Let N and N′ be two open nets with acyclic and finite behavior and let* $OG_N = [Q, C, \delta, q_0, \Phi]$ *and* $OG_{N'} = [Q', C, \delta', q_0', \Phi']$ *be the corresponding OGs.*

Then, $Strat(N') \supseteq Strat(N)$ *iff there is a mapping* $\xi: Q \to Q'$ *such that*

(i) $\xi(q_0) = q_0'$;

(ii) *if* $\xi(q) = q'$ *and* $(q, c, q_1) \in \delta$, *then there is a* q_1' *such that* $(q', c, q_1') \in \delta'$ *and* $\xi(q_1) = q_1'$; *and*

(iii) *for all* $q \in Q$, *the formula* $\Phi(q) \Rightarrow \Phi'(\xi(q))$ *is a tautology.*

The main value of this theorem is that it allows for a finite, operational approach to check an inclusion relation of two potentially infinite sets.

For the proof of this theorem, we rely on a fact about OGs as constructed in [16]. As we cannot repeat the whole approach of [16], we only include the following proposition and then sketch the proof of Theorem 5.4.

PROPOSITION 5.5 ([16]). *For every operating guideline* $OG_N = [Q, C, \delta, q_0, \Phi]$ *(of some service N) and all* $q \in Q$, *the formula* $\Phi(q)$

(i) *uses only literals c where there is some* $q' \in Q$ *with* $(q, c, q') \in \delta$ *and*

(ii) *is satisfied for the assignment assigning true to all literals in* $\Phi(q)$.

Proof of Theorem 5.4 (Sketch). Implication. Let $OG_{N'} = [Q', C, \delta', q_0', \Phi']$ and $OG_N = [Q, C, \delta, q_0, \Phi]$, and let $Strat(N') \supseteq Strat(N)$.

We can construct an open net M whose behavior corresponds exactly to the transition system $[Q, C, \delta, q_0]$. This can be achieved by using $Q \cup C$ as set of places (with C being the interface of M), and having, for each $(q_1, c, q_2) \in \delta$ a transition t_{q_1, c, q_2} that moves a token from q_1 to q_2, and removes (produces, respectively) a token from (on) c if c is an output (input) place of N. Let m_q denote a marking of M where there is a token on place q and no token elsewhere. Let m_{q_0} be the initial marking of M. By induction, it can be shown that, for all $q \in Q$, m_q is reached by Definition 5.2, with $\rho(m_q) = q$.

As there is a transition for each $(q, c, q') \in \delta$, we can derive from Proposition 5.5 that all annotations evaluate to *true* when M is evaluated according to Definition 5.2. Consequently, M

matches with OG_N and hence M is a strategy for N and thus, by assumption, a strategy for N'.

Being a strategy for N', there is a mapping ρ' from the markings of M to Q'. Define ξ: $Q \to Q'$ such that $\xi(q) = q'$ iff $\rho'(m_q) = q'$. By the structural similarity of Definition 5.2 and Theorem 5.4, it is easy to see that ξ satisfies the first two items required in Theorem 5.4. For verifying the third item, let $q \in Q$ and let β be an arbitrary assignment to literals occurring in $\Phi(q)$ where $\Phi(q)$ is *true*. Remove from M all those transitions $t_{q_1,\ c,\ q_2}$ where $\beta(c)$ is *false*. By Definition 5.2, the resulting open net is still a strategy for N and thus a strategy for N', too. Using Definition 5.2 again, we can see that $\Phi'(\xi(q))$ is *true* as well. Thus, $\Phi(q) \Rightarrow \Phi'(\xi(q))$ is a tautology.

Replication. Consider a mapping ξ as required and let M be a strategy for N. We show that M is a strategy for N', too. By Definition 5.2, there is a mapping ρ from the markings of M to Q. Let $\rho'(m) = \xi(\rho(m))$. For all markings reached by Definition 5.2, $\Phi(\rho(m))$ evaluates to *true* for the assignment described in Definition 5.2, and by the third item of Theorem 5.4, so does $\Phi'(\rho'(m))$. Consequently, M is a strategy for N', too. □

The operating guideline $\mathrm{OG}_{N'_{\mathrm{reg}}}$ of the private view N'_{reg} of the registration office (cf. Fig. 2b) is depicted in Fig. 6. It consists of 8 states and 15 transitions and introduces new strategies that make use of the reorganization of the registration office into two independent departments: it is now possible for a strategy for N'_{reg} to send a reqID message followed by noFingerprint which causes a deadlock in the public view N_{reg} and hence there is no strategy for N_{reg}.

Using Theorem 5.4, we can easily verify that *each* strategy for N_{reg} is a strategy for N'_{reg}, too; that is, $\mathit{Strat}(N'_{\mathrm{reg}}) \supseteq \mathit{Strat}(N_{\mathrm{reg}})$:

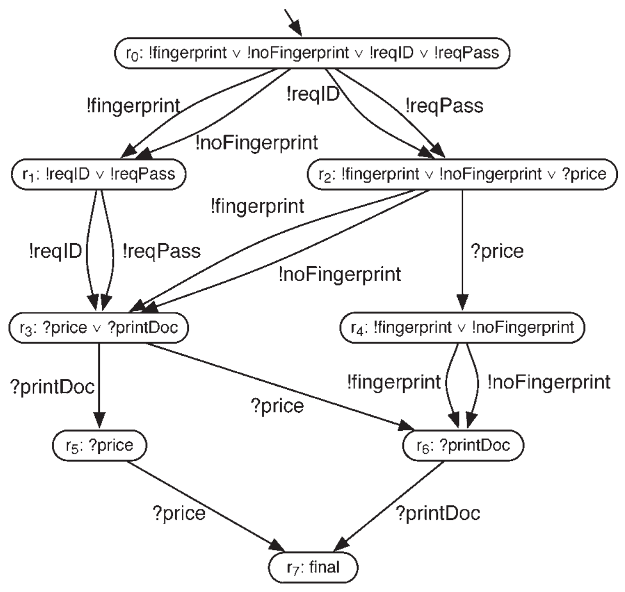


FIGURE 6. The operating guideline $\mathrm{OG}_{N'_{\mathrm{reg}}}$ of the private view of the registration office of Fig. 2b.

Obviously, the automaton underlying the private view's operating guideline $\mathrm{OG}_{N'_{\mathrm{reg}}}$ of Fig. 6 simulates the automaton underlying the public view's operating guideline $\mathrm{OG}_{N_{\mathrm{reg}}}$ of Fig. 5. For example, r_1 simulates the states q_1 and q_2; that is, $\xi(\mathsf{q}_1) = \xi(\mathsf{q}_2) = \mathsf{r}_1$.

Additionally, an annotation of a state of $\mathrm{OG}_{N'_{\mathrm{reg}}}$ implies the corresponding state's annotation in $\mathrm{OG}_{N_{\mathrm{reg}}}$. For instance, $\Phi(\mathsf{q}_1) = (\mathsf{reqPass}) \Rightarrow (\mathsf{reqID} \vee \mathsf{reqPass}) = \Phi'(\mathsf{r}_1)$. Therefore, we conclude that $\mathit{Strat}(N'_{\mathrm{reg}}) \supseteq \mathit{Strat}(N_{\mathrm{reg}})$.

As a counterexample we assume that N'_{reg} was the public view which all parties agreed upon. Then, using N_{reg} as an implementation is wrong: according to the public view N'_{reg}, the customer is allowed to send a reqID message followed by noFingerprint which is no strategy for the private view N_{reg} (cp. state q_3 of $\mathrm{OG}_{N_{\mathrm{reg}}}$ in Fig. 5).

For an implementation of the criteria in Theorem 5.4, finding the mapping ξ is the crucial task. As both OG_N and $\mathrm{OG}_{N'}$ are deterministic (i.e. in each state q there is at most one c-successor), this task actually amounts to a depth-first search through $\mathrm{OG}_{N'}$ which is mimicked in OG_N. The time and space required for finding ξ is thus linear in the number of states and edges of $\mathrm{OG}_{N'}$. This size, in turn, is equal to the number of states and edges of a particular strategy for N [19]. The accordance check based on Theorem 5.4 has been implemented in our tool Fiona, too.

6. CASE STUDY

The example contract depicted in Fig. 2a was derived from a real-life business process created by a customer of a German consulting company within their modeling tool. In cooperation with the consulting company, we analyzed the process for correctness.

In a first step, we exported the business process into a BPEL4Chor choreography [7] using the export function of the modeling tool. This BPEL4Chor choreography consists of multiple (abstract) WS-BPEL processes [11] and wiring information (the topology of the choreography). Thereby, each WS-BPEL process represents the public view of a party and the wiring information describes the processes' interplay.

In a second step, the processes were partially reorganized and all necessary implementation details were manually filled in. For example, activities were reordered, added or removed, and all opaque activities of the abstract WS-BPEL process were successively substituted by executable WS-BPEL activities. The resulting executable processes represent the corresponding private views of the parties. The wiring remained unchanged.

To check accordance between the private and the public views, we translated all (i.e. abstract and executable) WS-BPEL processes into open nets using our compiler

BPEL2oWFN[2] [20]. This compiler implements a feature-complete open nets semantics [21] for WS-BPEL and BPEL4-Chor. With feature-complete we mean, the open net semantics supports all concepts of WS-BPEL including the control flow, data flow, message flow, exception handling and compensation handling. As mentioned in Section 2, we only translate a single instance of each WS-BPEL process of the choreography and therefore do not need to correlate messages to process instances.

Although the semantics are expressed in terms of high-level open nets, the model generated by BPEL2oWFN abstracts from data to avoid generating an infinite state space. The implemented data abstractions [22] result in low-level open nets. Messages and the content of variables are modeled by undistinguishable tokens. Data-dependent branches (e.g. WS-BPEL's 'if' activity) are modeled by nondeterministic choices. In case a variable has a finite data domain (e.g. data of type Boolean), the resulting high-level open net can, however, be unfolded into a low-level open net without loss of information. The applied abstraction method guarantees that if two low-level open nets are related under accordance, then this holds for their corresponding high-level open nets, too.

On the open net level, we were finally able to compute the OGs and checked accordance between the public and the private views using Fiona. The translation and analysis procedure that has been described above is illustrated in Fig. 7.

It is worthwhile mentioning that the open net presented in Fig. 2a is a rather simplified version of the originally generated model. In particular, the negative control flow of the involved processes (i.e. fault, compensation and termination handling) is not depicted. Thus, the OGs of the original public and private views have considerably more edges and notes than the OGs of Figs 5 and 6. Consequently, tool support is mandatory. However, with the tool chain of BPEL2oWFN and Fiona, the check whether a change of the WS-BPEL process of a party is accordance-preserving can be automatized and, for example, integrated into WS-BPEL modeling tools.

The lessons we learned from this case study can be summarized as follows:

The implementation of a party's public view is an error-prone and nontrivial task even for experts. Thus, tool support to check accordance between a public view and its private view is mandatory.

In order to check accordance of two open nets, their OGs have to be computed in advance. Afterwards the two criteria of Theorem 5.4 have to be checked for the computed OGs. Thereby, the calculation of the OGs is far more challenging because the time and space complexity to compute an OG is high. However, experimental results have shown that the calculation of an OG is feasible in most practical applications (see [10], for instance). For the case study, Fiona was able

[2]Available at http://service-technology.org/bpe12owfn.

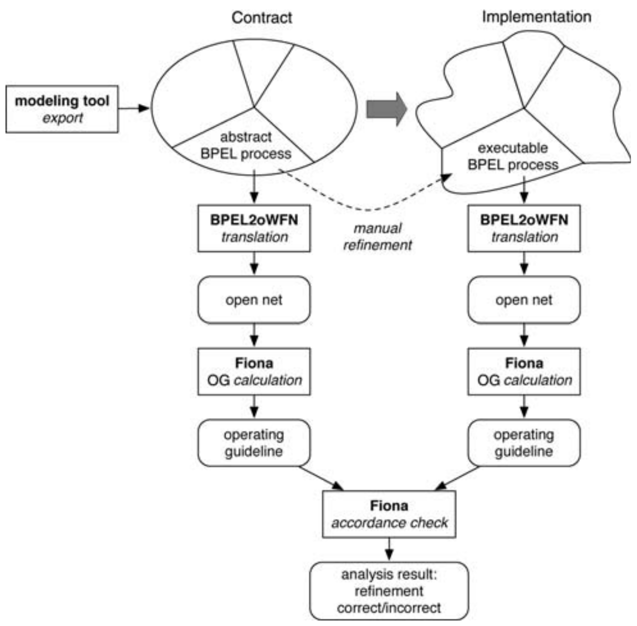


FIGURE 7. Application of the translation tool BPEL2oWFN and the analysis tool Fiona to check accordance between a modeled contract and an implemented BPEL4Chor choreography.

to compute the OGs of the nontrivial example processes and to verify accordance.

Finally, if two processes do not accord, it is hard to find the reason. Consequently, to apply our technique in practice, Fiona should provide the user with more diagnostic information.

7. RELATED WORK

7.1. Work based on Petri nets

The work presented in this paper mainly builds on results presented in [3, 4] where classical WFNs [12] are used as a formal model for a contract, and projection inheritance [14, 15] is used for relating public and private views. There, partitioning a contract into its parts may result in a public view that has no workflow structure. For instance, the public view can be an unconnected net. Thus, the approach of [3, 4] may require a 'massaging' step of the net. Using the more liberal notion of an open net, 'massaging' an unconnected net is not necessary in our approach. Another difference is caused by the different interaction models (synchronous versus asynchronous). In [3, 4], transition fusion (i.e. synchronous communication) is used to compose WFNs. This difference is reflected by the notions of projection inheritance and accordance. Two WFNs are related under projection inheritance iff they have the same observable behavior. As a result, two nets that differ in the order of sending two messages *a* and *b* are not related under projection inheritance, whereas the nets accord with each other. In [23], our notion of accordance has been proven to be more liberal than projection inheritance; that is, projection inheritance implies accordance.

Martens [24] presents a refinement notion for workflow modules. Workflow modules—like open nets—are a Petri net formalism to model (web) services. However, workflow modules are subject to several syntactic restrictions (similar to WFNs) and therefore less general than open nets. To decide refinement of acyclic workflow modules, Martens introduces a data structure called communication graph and a simulation relation on these graphs. A communication graph in some sense represents the communication behavior of a service and can be compared with a reduced version of our underlying automaton A of an operating guideline OG = A^{Φ} without any annotations. Due to the limitations of workflow modules and the loss of information by the reductions compared with our OGs, his structural simulation relation on communication graphs is only sufficient for refinement (accordance, in our terms) of services, whereas we were able to prove a criterion that is necessary and sufficient.

Vogler presents in [25] a livelock and deadlock preserving equivalence between Petri nets with interfaces. However, there is no direct implication in either direction between the equivalence of Vogler and accordance.

Bonchi *et al.* [26] model the behavior of services using a special kind of Petri nets, Consume-Produce-Read Nets. For their model they present saturated bisimulation as accordance relation. However, saturated bisimulation is too restrictive to allow reordering of sending messages. For example, two nets that differ in the order of sending two messages a and b are not saturated bisimular.

7.2. Work based on process calculi

Most process algebraic approaches define their refinement notions for synchronous communication. However, the translation of asynchronously communicating processes into synchronously communicating processes is well understood in the theory of process calculi. Therefore, the results presented in these papers can be applied for processes that communicate asynchronously.

Several authors propose refinement notions, called *conformance*, for processes specified in process calculi. These notions are similar to our accordance notion as they all guarantee independent refinement of each process while preserving a certain termination criterion. Some examples are given below.

An approach on service contracts similar to ours is presented by Bravetti and Zavattaro [27]. There, services are modeled as processes in the Calculus of Communicating Systems (CCS). The proposed conformance relation in [27] formalizes the correct contract composition property. This property is stricter than weak termination because besides the absence of deadlocks and livelocks, it also excludes infinite runs. To decide conformance, the authors prove 'should testing' [28] to be a sufficient criterion for their conformance notion. As the main difference, our algorithm to decide accordance, which is based on OGs, is complete in the sense that it is sufficient and necessary and it is implemented in our analysis tool Fiona. However, in contrast to our current decision procedure 'should testing' is not restricted to acyclic service models. In [29], Bravetti and Zavattaro enhance their correctness criterion by ensuring whenever a message can be sent, the other service is ready to receive this message. Systems that behave this way are called strongly compliant.

Castagna *et al.* [30] introduce a conformance notion that formalizes the absence of deadlocks and livelocks in finite-state systems. In contrast to accordance and other conformance notions, their conformance notion only demands the termination of the environment but not the termination of the process itself. In [31], this conformance notion has been proven equivalent to 'must testing' [32]. Note that 'must testing' is a weaker equivalence notion than 'should testing' as it cannot distinguish between a loop (i.e. a potentially infinite run) and a livelock.

Our accordance notion is also related to *testing* (see [33], for instance). A test for a process N is a process S such that S can always terminate in the composition with N (as required by the conformance notion of [30]). So it is easy to see that every strategy is a specific test that guarantees the termination of the test *and* the process. Our accordance criterion requires proper termination of both components. A detailed comparison of testing and accordance is, however, outside the scope of this paper.

Fournet *et al.* [34] consider CCS processes of asynchronous message passing software components. They present stuck-free conformance of such processes. Stuck-freedom formalizes the absence of deadlocks in the system. Our proposed notion of weak termination excludes deadlocks and also livelocks. To check conformance, the model checker Zing [35] is used. Stuck-free conformance requires among others that an implemented process N' simulates its original process N. Our approach, in contrast, requires a simulation relation between OGs of N and N'; that is, we do not compare N and N', but their strategies. Figure 8 depicts two processes N and N' where N' accords with N, but N' does not conform to N. The reason is, after having received b, the net N' can receive c and N does not. Thus, N' does not simulate N. Therefore, it seems that our notion of accordance is coarser than stuck-free conformance.

Busi *et al.* [36] present a notion of conformance between a choreography language based on WS-CDL and an orchestration language based on abstract WS-BPEL. This conformance notion is branching bisimulation. Conformance can be used to check if the implementation (i.e. the orchestrated system) behaves accordingly the conversation rules of the choreography. Like saturated bisimulation in [26], branching bisimulation is also too restrictive to allow reordering of messages.

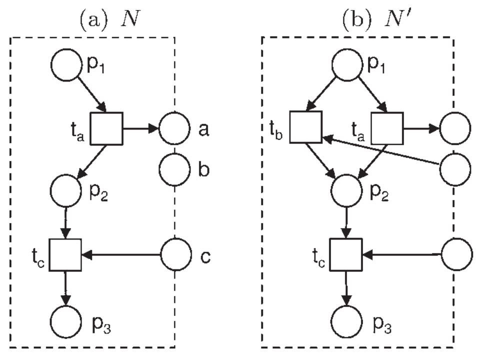


FIGURE 8. Accordance does not imply conformance of [34]: N' accords with N, but N' does not conform to N.

7.3. Work based on service substitutability

The concept of contracts is also related to the problem when a service can be substituted by another service. Most of this work, however, is restricted to synchronous communication [37–40] whereas we consider asynchronous message passing. Benatallah *et al.* [39] present four notions for substitutability of synchronously communicating automata models of services. In this paper, we cover two of them: *equivalence* and *subsumption*. Equivalence in our notion means that both services have the same set of strategies and subsumption means the inclusion of the set of strategies (accordance).

The *ComFoRT* framework [41] analyzes whether a software component S implemented in the programming language C can be substituted by another software component S'. S can be substituted by S' if the following two criteria hold: (i) every behavior possible in S must also be a behavior of S' and (ii) the new version of the software system must satisfy previously established correctness properties. Behavior inclusion is verified by trace comparison of the software components, which also does not allow the reordering of messages.

7.4. Work on open nets

Lohmann *et al.* [42] translate choreographies specified in the choreography language BPEL4Chor [7] into open nets. The model is then checked for deadlocks using the model checker Low Level Analyzer (LoLA) [43]. Because this approach can only be used if there is a party that can access the whole choreography, it is in general not applicable in practice, but can be used to check whether the public view of the overall contract is weakly terminating. The design of the public view can be further assisted: Lohmann [44] describes an algorithm to diagnose communication-related problems of service models. This diagnosis information can be used to support the automatic correction of deadlocking choreographies which is presented in [45].

In [23], we presented a number of transformation rules to incrementally build a private view of a party to a given public view. We proved that these rules preserve accordance between the public and the private views and hence the resulting private view is correct by design. Accordance-preserving transformation notions can also be applied directly on the BPEL code. In [46], some of these code transformation rules are presented.

7.5. Workflow perspectives, data and semantics

In our formal model, open nets, we focus on the so-called control-flow perspective and abstract from other perspectives such as data and resources. For example, messages are represented by undistinguishable (black) tokens and, as indicated before, we assume that correlation is handled in some way.

Data and other perspectives can be modeled with high-level Petri nets [47], for instance. In case the data domain is finite, the high-level Petri nets can be unfolded into an equivalent low-level Petri net with undistinguishable black tokens. However, unfolding might result in huge nets making the analysis hard or even impossible. Infinite data domains may be abstracted to a finite domain using techniques such as abstract interpretation [48]. If this is not possible, generating the OG for such a net is in general undecidable.

Andonoff *et al.* [49] propose Petri nets with objects [50], a special class of high-level Petri nets, as a process model. In this model, data can be adequately represented. A transition may have pre- and post-conditions. It is enabled if its precondition holds. By firing this transition, an action (i.e. a method call) can be executed. Petri nets with objects can be used to make comprehensive models covering multiple perspectives. For example, messages can be modeled by objects adequately reflecting their content. Moreover, in [49] algorithms are presented to transform Petri nets with objects [50] to OWL-S specifications [51], a quite popular semantic web service description language. Such an OWL-S specification can be published at a service broker and thus be used for service brokering. However, as with any high-level Petri net, the increased expressive power results in weaker analysis results.

Our approach is also restricted to the behavior of services and does not cover other important aspects such as semantical information and QoS. The reason for doing so is that our proposed approach builds on the concept of an OG and we have not integrated such concepts yet.

The commitment ontology proposed by Desai *et al.* [52] is an another approach for modeling interorganizational processes. Thereby a commitment specifies an interaction between two parties. The commitment ontology can be seen as a modeling language. It allows to specify pre- and post-conditions, data and semantical information. Processes specified in this ontology can, in principle, be translated, in automata or high-level Petri nets, for instance. To apply our techniques to such models, we need a finite data domain and thus have to deal with the same problems as already mentioned above.

As indicated above we acknowledge the importance of other perspectives (e.g. data) and see the value of using richer modeling notations. However, the focus of this paper is on the basic interaction structures between different parties. Our findings and results remain valid when incorporating additional perspectives or when using other process modeling notations (e.g. Business Process Modeling Notation (BPMN), WS-BPEL, etc.).

8. CONCLUSION

In interorganizational cooperations, the involved parties specify a public version of the overall process which serves as a contract. Later on, each party implements its part of the contract (i.e. the public view). Such a local modification is nontrivial as it may cause global errors such as deadlocks.

In this paper, we proposed a formal notion of a contract based on open nets. Our correctness criterion, weak termination, guarantees the ability to always be able to terminate properly (e.g. the overall process cannot get into a deadlock or livelock). To decide if the public view N of a party can be substituted by a modified version, i.e. the private view N', we presented the notion of accordance. N' accords with N if N and N' have the same interface and any environment that can cooperate with N can also cooperate with N'. The value of our accordance notion is that it can be checked locally and guarantees that the overall process preserves the weak termination property.

To check accordance automatically, we employed our concept of OGs. The OG of an open net N characterizes all open nets M (called strategies) such that the composition of M and N weakly terminates. We proved for open nets N and N' with acyclic and finite behaviors that N' accords with N if the OG of N' characterizes at least the strategies that are characterized by the OG of N.

Our compiler BPEL2oWFN enables us to derive an open net model of a contract from the contract's BPEL4Chor specification. On the open net level, our analysis tool Fiona computes the OGs of the private and the public views and checks accordance.

It is worthwhile mentioning that the presented general concepts of contracts and accordance are *not limited to open nets but can also be translated into other frameworks using message passing as a communication paradigm*. Furthermore, our approach can also be applied to services specified in a service description languages other than WS-BPEL as long as this language can be translated into Petri nets or at least into automata. For example, there exists tool support to translate languages like WS-CDL and BPMN into a formal model and therefore into automata.

If the service is described as a programming language like Java or C, things become more complicated. However, there are also approaches like the one of [41] that can translate such a service description into an automaton (or another formal model) using techniques that have proven themselves in the area of model checking [53] and static program analysis [54].

In ongoing work, we want to generalize our accordance check to open nets with cyclic behavior. For this purpose, our correctness check must also guarantee the absence of livelocks which is more challenging than checking deadlocks. In addition, we want to improve the diagnosis in case accordance does not hold between two open nets. Finally, we also want to investigate the relation between our notion of accordance and equivalences known from process algebra, in particular testing equivalence.

ACKNOWLEDGEMENTS

The authors are grateful to Arjan J. Mooij and Marc Voorhoeve for their help in pointing out differences between the presented notion of accordance and the existing conformance notions described in process calculi. The authors also thank Robert Danitz and Leonard Kern for their contribution to the implementation of Fiona.

FUNDING

N. Lohmann is funded by the DFG project 'Operating Guidelines for Services' (WO 1466/8-1). C. Stahl is funded by the DFG project 'Substitutability of Services' (RE 834/16-1).

REFERENCES

[1] Papazoglou, M.P. (2001) Agent-oriented technology in support of e-business. *Commun. ACM*, **44**, 71–77.

[2] Papazoglou, M.P. (2007) *Web Services: Principles and Technology*. Prentice Hall, Essex.

[3] Aalst, W.M.P.v.d. (2003) Inheritance of interorganizational workows: how to agree to disagree without loosing control? *Inf. Technol. Manage. J.*, **4**, 345–389.

[4] Aalst, W.M.P.v.d. and Weske, M. (2001) The P2P Approach to Interorganizational Workflows. *Proc. CAiSE'01*, Interlaken, Switzerland, August 12–14, Lecture Notes in Computer Science, Vol. 2068, pp. 140–156. Springer, Heidelberg.

[5] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y. and Barreto, C. (2005) *Web Services Choreography Description Language Version 1.0.* W3C Candidate Recommendation, 9 November. W3C, Cambridge, MA, USA.

[6] Zaha, J.M., Barros, A.P., Dumas, M. and ter Hofstede, A.H.M. (2006) Let's Dance: A Language for Service Behavior Modeling. *Proc. OTM Confederated Int. Conf., Part I*, Montpellier, France, October 29 –November 3, Lecture Notes in Computer Science, Vol. 4275, pp. 145–162. Springer, Heidelberg.

[7] Decker, G., Kopp, O., Leymann, F. and Weske, M. (2007) BPEL4Chor: Extending BPEL for Modeling Choreographies. *Proc. ICWS 2007*, Salt Lake City, UT, USA, July 9–13, pp. 296–303. IEEE Computer Society Press.

[8] Massuthe, P., Reisig, W. and Schmidt, K. (2005) An operating guideline approach to the SOA. *Ann. Math. Comput. Teleinf.*, **1**, 35–43.

[9] Reisig, W. (1985) Petri Nets, In *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, Heidelberg, New York.

[10] Lohmann, N., Massuthe, P. and Wolf, K. (2007) Operating Guidelines for Finite-state Services. *Proc. of ICATPN 2007*, Siedlce, Poland, June 25–29, Lecture Notes in Computer Science, Vol. 4546, pp. 321–341. Springer, Heidelberg.

[11] Alves, A. *et al*. (2007) *Web Services Business Process Execution Language Version 2.0*. OASIS Standard, 11 April. OASIS, Billerica, MA, USA.

[12] Aalst, W.M.P.v.d. (1998) The application of Petri nets to workflow management. *J. Circuits Syst. Comput.*, **8**, 21–66.

[13] Kindler, E. (1997) A Compositional Partial Order Semantics for Petri Net Components. *Proc. ICATPN 1997*, Toulouse, France, June 23–27, Lecture Notes in Computer Science, Vol. 1248, pp. 235–252. Springer, Heidelberg.

[14] Aalst, W.M.P.v.d. and Basten, T. (2002) Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, **270**, 125–203.

[15] Basten, T. and Aalst, W.M.P.v.d. (2001) Inheritance of behavior. *J. Log. Algebr. Program.*, **47**, 47–145.

[16] Massuthe, P. and Schmidt, K. (2005) Operating Guidelines—An Automata-theoretic Foundation for the Service-oriented Architecture. *Proc. QSIC 2005*, Melbourne, Australia, September 19–20, pp. 452–457. IEEE Computer Society Press.

[17] Wombacher, A., Fankhauser, P., Mahleko, B. and Neuhold, E.J. (2004) Matchmaking for business processes based on choreographies. *Int. J. Web Serv. Res.*, **1**, 14–32.

[18] Glabbeek, R.J.v. (2001) The Linear Time—Branching Time Spectrum I; The Semantics of Concrete, Sequential Processes. In Bergstra, J., Ponse, A. and Smolka, S. (eds), *Handbook of Process Algebra*, Chapter 1, pp. 3–99. Elsevier, Amsterdam, The Netherlands.

[19] Massuthe, P. and Wolf, K. (2007) An algorithm for matching non-deterministic services with operating guidelines. *Int. J. Bus. Process Integr. Manage.*, **2**, 81–90.

[20] Lohmann, N., Massuthe, P., Stahl, C. and Weinberg, D. (2006) Analyzing Interacting BPEL Processes. *Proc. BPM 2006*, Vienna, Austria, September 5–7, Lecture Notes in Computer Science, Vol. 4102, pp. 17–32. Springer, Heidelberg.

[21] Lohmann, N. (2008) A Feature-complete Petri Net Semantics for WS-BPEL 2.0. *Proc. WS-FM 2007*, Brisbane, Australia, September 28–29, Lecture Notes in Computer Science, Vol. 4937, pp. 77–91. Springer, Heidelberg.

[22] Lohmann, N., Massuthe, P., Stahl, C. and Weinberg, D. (2008) Analyzing interacting WS-BPEL processes using flexible model generation. *Data Knowl. Eng.*, **64**, 38–54.

[23] Aalst, W.M.P.v.d., Lohmann, N., Massuthe, P., Stahl, C. and Wolf, K. (2007) From Public Views to Private Views—Correctness-by-design for Services. *Proc. WS-FM 2007*, Brisbane, Australia, September 28–29, Lecture Notes in Computer Science, Vol. 4937, pp. 139–153. Springer, Heidelberg.

[24] Martens, A. (2005) Analyzing Web Service based Business Processes. In Cerioli, M. (ed.), *Proc. FASE 2005*, Edinburgh, UK, April 4–8, Lecture Notes in Computer Science, Vol. 3442, pp. 19–33. Springer, Heidelberg.

[25] Vogler, W. (1992) *Modular Construction and Partial Order Semantics of Petri Nets*, Lecture Notes in Computer Science, Vol. 625. Springer, Berlin.

[26] Bonchi, F., Brogi, A., Corfini, S. and Gadducci, F. (2007) A Behavioural Congruence for Web Services. *Proc. FSEN 2007*, Tehran, Iran, April 17–19, Lecture Notes in Computer Science, Vol. 4767, pp. 240–256. Springer, Heidelberg.

[27] Bravetti, M. and Zavattaro, G. (2007) Contract based Multi-party Service Composition. *Proc. FSEN 2007*, Tehran, Iran, April 17–19, Lecture Notes in Computer Science, Vol. 4767, pp. 207–222. Springer, Heidelberg.

[28] Rensink, A. and Vogler, W. (2007) Fair testing. *Inf. Comput.*, **205**, 125–198.

[29] Bravetti, M. and Zavattaro, G. (2007) A Theory for Strong Service Compliance. *Proc. COORDINATION 2007*, Paphos, Cyprus, June 6–8, Lecture Notes in Computer Science, Vol. 4467, pp. 96–112. Springer, Heidelberg.

[30] Castagna, G., Gesbert, N. and Padovani, L. (2008) A Theory of Contracts for Web Services. *Proc POPL 2008*, San Francisco, CA, USA, January 7–12, pp. 261–272. ACM Press.

[31] Laneve, C. and Padovani, L. (2007) The *must* Preorder Revisited. In Caires, L. and Vasconcelos, V.T. (eds), *Proc. CONCUR 2007*, Lisbon, Portugal, September 3–8, Lecture Notes in Computer Science, Vol. 4703, pp. 212–225. Springer, Heidelberg.

[32] Nicola, R.D. and Hennessy, M. (1984) Testing equivalences for processes. *Theor. Comput. Sci.*, 34, 83–133.

[33] Bruda, S.D. (2004) Preorder Relations. In Broy, M., Jonsson, B., Katoen, J.P., Leucker, M. and Pretschner, A. (eds), *Model-based Testing of Reactive Systems*, Dagstuhl, Germany, January, Lecture Notes in Computer Science, Vol. 3472, pp. 117–149. Springer, Heidelberg.

[34] Fournet, C., Hoare, C.A.R., Rajamani, S.K. and Rehof, J. (2004) Stuck-free Conformance. *Proc. CAV 2004*, Boston, MA, USA, July 13–17, Lecture Notes in Computer Science, Vol. 3114, pp. 242–254. Springer, Heidelberg.

[35] Andrews, T., Qadeer, S., Rajamani, S.K., Rehof, J. and Xie, Y. (2004) Zing: A Model Checker for Concurrent Software. *Proc. CAV 2004*, Boston, MA, USA, July 137–17, Lecture Notes in Computer Science, Vol. 3114, pp. 484–487. Springer, Heidelberg.

[36] Busi, N., Gorrieri, R., Guidi, C., Lucchi, R. and Zavattaro, G. (2005) Choreography and Orchestration: A Synergic Approach for System Design. *Proc. ICSOC 2005*, Amsterdam, The Netherlands, December 12–15, Lecture Notes in

Computer Science, Vol. 3826, pp. 228–240. Springer, Heidelberg.

[37] Bordeaux, L., Salaüun, G., Berardi, D. and Mecella, M. (2004) When are Two Web Services Compatible? *Proc. TES 2004*, Toronto, Canada, August 29–30, Lecture Notes in Computer Science, Vol. 3324, pp. 15–28. Springer, Heidelberg.

[38] Beyer, D., Chakrabarti, A. and Henzinger, T. (2005) Web Service Interfaces. *Proc. WWW 2005*, Chiba, Japan, May 10–14, pp. 148–159. ACM Press.

[39] Benatallah, B., Casati, F. and Toumani, F. (2006) Representing, analysing and managing Web service protocols. *Data Knowl. Eng.*, 58, 327–357.

[40] Kang, M., Froscher, J., Sheth, A., Kochut, K. and Miller, J. (1999) A Multilevel Secure Workflow Management System. *Proc. CAiSE 1999*, Heidelberg, Germany, June 14–18, Lecture Notes in Computer Science, Vol. 1626, pp. 271–285. Springer, Heidelberg.

[41] Sharygina, N., Chaki, S., Clarke, E. and Sinha, N. (2005) Dynamic Component Substitutability Analysis. *Proc. FM 2005*, Newcastle, UK, July 18–22, Lecture Notes in Computer Science, Vol. 3582, pp. 512–528. Springer, Heidelberg.

[42] Lohmann, N., Kopp, O., Leymann, F. and Reisig, W. (2007) Analyzing BPEL4Chor: Verification and Participant Synthesis. *Proc. WS-FM 2007*, Brisbane, Australia, September 28–29 Lecture Notes in Computer Science, pp. 46–60. Springer, Heidelberg.

[43] Schmidt, K. (2000) LoLA: A Low Level Analyser. *Proc. ICATPN 2000*, Aarhus, Denmark, June 26–30, Lecture Notes in Computer Science, Vol. 1825, pp. 465–474. Springer, Heidelberg.

[44] Lohmann, N. (2008) Why does my Service have no Partners? *Proc. WS-FM 2008*, Milan, Italy, September 4–5 Lecture Notes in Computer Science. Springer, Heidelberg. To appear.

[45] Lohmann, N. (2008) Correcting Deadlocking Service Choreographies using a Simulation-based Graph Edit Distance. *Proc. BPM 2008*, Milan, Italy, September 1–4, Lecture Notes in Computer Science, Vol. 5240, pp. 132–147. Springer, Heidelberg.

[46] König, D., Lohmann, N., Moser, S., Stahl, C. and Wolf, K. (2008) Extending the Compatibility Notion for Abstract WS-BPEL Processes. *Proc. WWW 2008*, Beijing, China, April 21–25, pp. 785–794. ACM Press.

[47] Jensen, K. (1997) Coloured Petri Nets (2nd edn). In *EATCS Monographs on Theoretical Computer Science.* Vol. 1–3. Springer, Berlin, Heidelberg, New York.

[48] Cousot, P. and Cousot, R. (1977) Abstract Interpretation: A unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. *Proc. POPL 1977*, Los Angeles, CA, USA, January, pp. 238–252. ACM Press, New York, NY.

[49] Andonoff, E., Bouzguenda, L. and Hanachi, C. (2005) Specifying Web Workflow Services for finding Partners in the Context of Loose Inter-organizational workflow. In van der Aalst, W.M.P., Benatallah, B., Casati, F. and Curbera, F. (eds), *Proc. BPM 2005*, Nancy, France, September 5–8, Lecture Notes in Computer Science, Vol. 3649, pp. 120–136. Springer, Heidelberg.

[50] Sibertin-Blanc, C. (2001) CoOperative Objects: Principles, Use and Implementation. *Concurrent Object-Oriented Programming and Petri Nets, Advances in Petri Nets*, Lecture Notes in Computer Science, Vol. 2001, pp. 216–246. Springer, Heidelberg, Germany.

[51] The OWL Services Coalition (2003). *OWL-S: semantic markup for web services*. http://www.daml.org/ services.

[52] Desai, N., Chopra, A.K. and Singh, M.P. (2007) Representing and Reasoning about Commitments in Business Processes. *Proc. AAAI 2007*, Vancouver, British Columbia, Canada, July 22–26, pp. 1328–1333. AAAI Press.

[53] Clarke, E.M., Grumberg, O. and Peled, D.A. (2000) *Model Checking*. MIT Press, Cambridge, MA.

[54] Nielson, F., Nielson, H.R. and Hankin, C. (2005) *Principles of Program Analysis* (2nd edn). Springer, Berlin.