Using Linear Logic to Verify Requirement Scenarios in SOA Models based on Interorganizational WorkFlow Nets Relaxed Sound

Kênia Santos de Oliveira, Vinícius Ferreira de Oliveira and Stéphane Julia
Computing Faculty, Federal University of Uberlândia, Uberlândia, Brazil

Keywords: Interorganizational WorkFlow Net, Petri Nets, Linear Logic, Service-oriented Architecture.

Abstract: This paper presents a method for requirement verification in Service-Oriented Architecture (SOA) models based on Interorganizational WorkFlow nets which are not necessarily deadlock-freeness. In this method, a requirement model corresponds to a public model that only specify tasks which are of interest of all parties involved. An architectural model is considered as a set of private processes that interact through asynchronous communication mechanisms in order to produce the services specified in the corresponding requirement model. Services can be seen as scenarios of WorkFlow nets. For each scenario that exists in the requirement model a proof tree of Linear Logic can be produced, and for each scenario correctly finalized, a precedence graph that specifies the task sequence can be derived. For each scenario of the architectural model, similar precedence graphs can be produced. The precedence graphs of the requirement and architectural model are then compared in order to verify if all existing scenarios of the requirement model also exist at the architectural model level. The comparison of behavior between distinct discrete events models is based on the notion of branching bisimilarity that prove behavioral equivalence between distinct finite automata. The example used to illustrate the proposed approach, shows that the method can be effective to identify if a SOA-based system satisfy the business needs specified by a model of public requirements.

1 INTRODUCTION

To represent the collaboration processes that involve multiple organizations, consideration has been given to interorganizational workflow processes. An interorganizational workflow processes allows organizations with complementary skills to perform jobs that are not within the range of a single organization. Interorganizational workflows can be easily represented by Interorganizational WorkFlow nets (van der Aalst, 2000).

To verify the correctness of an interorganizational workflow process, the soundness property is often used. However, a wide variety of interorganizational workflow processes are not completely sound and may lead the interorganizational business process to deadlock situations, for example (Fahland et al., 2011). Therefore, in (Passos and Julia, 2014), an approach is presented for deadlock-freeness scenarios detection in interorganizational workflow processes modeled by Interorganizational WorkFlow nets.

An important issue in Software Engineering is to ensure that a software architecture proposal reproduces the behavior of the requirement analysis model. The verification that the behavior of the requirement model exists within the corresponding architectural model minimizes risks of failure in projects, increasing the guarantee of software quality and avoiding rework costs (Goknil et al., 2014). Only a few workstudies address this problem (Zernadji et al., 2015) in the specific context of Software-Oriented Architecture (SOA). It is then of great interest to propose an approach that verifies if the requirements defined in analysis models (in terms of dynamic behavior) are also present in SOA models. In a more specific sense in those models considered as not sound, it is of great interest to consider deadlock-freeness scenarios as the main features of the requirement analysis.

This article presents an approach based on a kind of comparative analysis between requirement and architectural models in context of SOA. Through this comparative analysis, it is possible to verify if the main business relationship between the organizations involved can be provided without leading the system to a deadlock situations. Such an analysis can be based on Linear Logic proofs produced from interorganizational workflow processes modeled by Interorganizational WorkFlow nets. As a matter of fact, some studies have already shown the relationship between the Petri net theory and Linear Logic, such as

Oliveira, K., Oliveira, V. and Julia, S.
Using Linear Logic to Verify Requirement Scenarios in SOA Models based on Interorganizational WorkFlow Nets Relaxed Sound.
DOI: 10.5220/0006290202540262
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 254-262
ISBN: 978-989-758-248-6

(Girault et al., 1997) and (Riviere et al., 2001), since there is an almost direct translation of the structure of a Petri net into formulas of Linear Logic. To show the equivalence of two distinct models (requirement and architectural models), a new definition of semantic equivalence for distinct models can be produced, as already performed, for example, in the context of process algebras with the notion of bisimulation (Basten, 1998). A new definition of semantic equivalence in the context of Linear Logic compatible with the notion of bisimulation was then be introduced in this work. This in order to verify if requirements specified in a kind of public model also exist in the corresponding SOA model. The architectural model is be based on Interorganizational WorkFlow net not necessarily sound. The proposed approach accept in particular relaxed sound processes (when all the activities of the system appeared in at least one process that ended correctly).

The remainder of the article is organized as follows. In section 2, the definition of Interorganizational WorkFlow nets nets are presented as well as an overview of Linear Logic and the notion of Branching Bisimilarity. The approach to formally detect requirements present in SOA models is proposed in section 3. Finally, section 4 concludes this work.

2 THEORETICAL BACKGROUND

2.1 Interorganizational WorkFlow Net

Collaboration processes that involve multiple organizations can be represented as interorganizational workflow processes. Essentially, an interorganizational workflow process is a set of loosely coupled 'local' workflow processes involved in a same 'global' workflow process (van der Aalst, 2000). In order to model an interorganizational workflow process, an Interorganizational WorkFlow net (IOWF-net) can be used. Each of the local workflow processes is described by a Petri net called WorkFlow net (WF-net). According to (van der Aalst, 1998), the formal definition of a IOWF-net is based on a tuple $IOWF-net = \{PN_1, PN_2, ..., PN_n, P_{AC}, AC\}$, where:

1. $n \in \mathbb{N}$ is the number of Local WorkFlow nets (LWF-nets);
2. For each $k \in \{1, ..., n\}$: PN_k is a WF-net with source place i_k and sink place o_k;
3. P_{AC} is the set of asynchronous communication elements (communication places).
4. AC corresponds to the asynchronous communication relation. It specifies a set of input transitions and a set of output transitions for each asynchronous communication element.

To clarify the concepts defined above, consider the synthetic example presented in Figure 1. This IOWF-net has two LWF-nets: A and B. Each one has only one source place (iA for LWF-A and iB for LWF-B) and one sink place (oA for LWF-A and oB for LWF-B). The places PC1, PC2 and PC3 are the communication places.

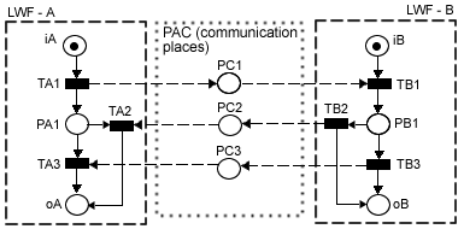

Figure 1: An IOWF-net example.

In (van der Aalst, 1998), Unfolded Interorganizational WorkFlow nets U(IOWF-net) are defined. The unfolding of an IOWF-net corresponds to the transformation of an IOWF-net into a simple WF-net. In the U(IOWF-net), all the LWF-nets are included into a single workflow process considering a start transition and a termination transition. A global source place 'i' and a global sink place 'o' have then to be added in order to respect the basic structure of a simple WF-net, and the asynchronous communication elements existing between the different LWF-nets are mapped into ordinary places. In section 3 an example of U(IOWF-net) can be observed (Figure 4).

The correctness of a workflow process is associated to the verification of the soundness property (van der Aalst, 1996). This property ensure that all the process instances of a WF-net will be correctly treated and finalized and will not have tasks not executed in the process verified. Many studies have already considered the analysis of the soundness property in the interorganizational workflow process case as in (van der Aalst, 1998) and (Passos and Julia, 2013). It is interesting for the organizations, in cases where the classical soundness correction criterion is not satisfied for a given IOWF-net, to identify scenarios that satisfy the business requirements. In this context, variants of the soundness criterion were proposed. In (Passos, 2016), it is defined an approach to identify if a IOWF-net is sound, relaxed sound or weak sound. An IOWF-net is relaxed sound if each process task is considered in at least one of the runs that finishes correctly. An IOWF-net is weak sound if there is no deadlock in the model and the correct completion of the process is guaranteed, even in cases where the analyzed process presents dead tasks.

2.2 Linear Logic

Instead of emphasizing truth, as in classical logic, or proof, as in intuitionistic logic, Linear Logic (Girard, 1987) emphasizes the role of formulas as resources.

In the Linear Logic there are several connectives, but, in this paper only tow are used. The *times* connective (denoted by '$\otimes$') represents simultaneous availability of resources; for instance, '$A \otimes B$' represents the simultaneous availability of resources 'A' and 'B'. The *linear implies* connective (denoted by '$\multimap$') represents a state change; for instance, '$A \multimap B$' denotes that consuming 'A', 'B' is produced; after the production of 'B', 'A' will not be available anymore.

The following definition presented in (Riviere et al., 2001) shows how to translate a Petri net model into Linear Logic formulas:

- A marking M is a monomial in $\otimes$ and is represented by $M = A_1 \otimes A_2 \otimes ... \otimes A_k$ where A_i are place names.
- A sequent $M, t_i \vdash M'$ represents a scenario where M and M' are respectively the initial and final markings, and t_i is a list of non-ordered transitions. For instance, in the LWF-A shown in Figure 1, the sequent $iA, TA1, TA2 \vdash oA$ represents one possible scenario of this WorkFlow net.

A sequent can be proven by applying the rules of the sequent calculus. In this paper, the following rules are used: $\multimap_L$ rule - expresses a transition firing and generates two sequents (the right sequent represents the subsequent remaining to be proved and the left sequent represents the consumed tokens by this firing); $\otimes_L$ rule - is used to transform a marking in an atoms list; $\otimes_R$ rule - transforms a sequent such as $A, B \vdash A \otimes B$ into two identity ones $A \vdash A$ and $B \vdash B$.

Linear Logic proof tree is read from the bottom-up. The proof stops when the identity sequent $o \vdash o$ ('o' correspond to a sink place) is produced, when there is not any rule that can be applied or when all the leaves of the proof tree are identity sequents.

The Linear Logic proof trees can be transformed into precedence graphs, as shown in (Diaz, 2009) by labeling the corresponding proof trees. To label a proof tree, each time the $\multimap_L$ rule is applied, the corresponding transition t_i label the application of the rule, as well as the atoms produced and consumed. Furthermore, the initial event must be labeled by i_i and the final event must be labeled by f_i. The labels are shown in the proof tree above the atoms and below the rules $\multimap_L$. In a precedence graph, the vertices are events and the arcs are identity sequent, i.e relation between the event that produced the atom and one that consumed the atom (Diaz, 2009). The subsection 3.2 shows how proof trees and the corresponding labelings are built.

In (Passos and Julia, 2013), the Linear Logic proof trees was used to analyze the soundness criterion in an IOWF-nets. For this, it is necessary to verify soundness for each LWF-net that composes the IOWF-net and for the unfolded net, U(IOWF-net). A WorkFlow net is sound if it respects the restriction 1 and 2 shown below.

1. For each scenario of the analysed WorkFlow net, at the end of the proof tree: (a) there is not any available atom for consumption (all places of the WorkFlow net are empty); (b) there is not any available transition formula that was not fired; (c) just one identity sequent $o \vdash o$ was produced (just one token appears in the sink place).
2. Considering all scenarios for the WorkFlow net analyzed, each transition t_i has to appear in, at least, one scenario (the activities of the WorkFlow net will appears in at least one process).

A similar approach can be used to verify variants of the soundness criterion (relaxed and weak sound) presented in subsection 2.1. According to (Passos, 2016), an IOWF-net is relaxed sound if the proof trees of the U(IOWF-net) scenarios respect the restriction 1 (a and c), and for each scenario that satisfy this restriction, the restriction 2 is also respected. An IOWF-net is weak sound if all the proof trees of the U(IOWF-net) scenarios respect the restriction 1 (a and c).

2.3 Branching Bisimilarity

The idea of bisimilarity was first introduced in (Park, 1981) and can be interpreted in the following manner: two processes are equivalent if and only if they can always copy or simulate the actions of each other. Bisimilarity does not make the distinction between external (observable) actions and internal (silent) actions; therefore it is not a suitable equivalence concept for processes with internal behavior.

Branching bisimilarity was first introduced in (van Glabbeek and Weijland, 1996) and is a variant of bisimilarity; however, branching bisimilarity distinguishes external behavior from internal behavior. The distinction between external and internal behavior captures the idea that an environment observing two processes might not be able to see any differences in their behavior while internally the two processes perform different computations (Basten, 1998). Therefore, to be able to make a distinction between external and internal behavior (hidden events), silent actions can be introduced. Silent actions are actions that cannot be observed. Usually, silent actions are denoted

with the action label τ. The Figure 2 presented in (Basten, 1998) shows the essence of branching bisimulation.

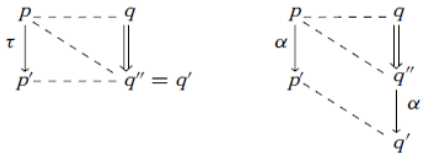

Figure 2: The essence of branching bisimulation.

In Figure 2, τ represents a silent action, α represents an observable action, and p, q, p', q', q'' represent processes. On the left side of Figure 2, the process p can evolve into another process p' by executing a silent action (τ) and the process q can evolve into another process q'' by executing a sequence of zero or more τ actions (such a fact represented by relation '$\Longrightarrow$'). Then, it is possible to observe on the left side of the figure that the process p has an equivalence relation with the processes q and q'' and the process p' has an equivalence relation with the process q''. These facts clearly state that two equivalent processes will continue equivalent after the introduction of some additional silent actions in one of the processes or even in both.

On the right side of Figure 2, the process p can evolve into another process p' by executing an observable action (α), the process q can evolve into another process q'' by executing a sequence of zero or more τ actions (relation '$\Longrightarrow$') and the process q'' can evolve into another process q' by executing an observable action (α). Then, it is possible to observe on the right side of the figure that the process p has an equivalence relation with the processes q and q'', and the process p' has an equivalence relation with the process q'. These facts clearly state that two equivalent processes will continue equivalent after the introduction of some additional observable actions in one of the processes only if the same observable actions also exist in the other process and respect the same sequence constraints in both processes.

3 VERIFICATION OF DEADLOCK-FREENESS REQUIREMENT SCENARIOS IN SOA MODELS

In (van der Aalst, 2003), Aalst defined the P2P-WorkFlow nets. The approach is used to represent interorganizational workflow processes and was proposed to ensure that the local implementation of a workflow does not create all kinds of anomalies over organizational borders. In such an approach, two views of the system are considered: the public one and the private one. A public WF-net specifies the expected system requirements the parties involved will have to perform. A private WF-net typically contains several tasks which are only of local interest and which do not appear in the public WF-net model.

The Figure 3 represents a public WF-net example and Figure 4 represents the corresponding private models (LWF - contractor and LWF - subcontractor) modeled by an IOWF-net. In Figure 4, a global source place i and a global sink place o were added in order to respect the basic structure of a simple WF-net as explained in subsection 2.1. Therefore, this model is a U(IOWF-net).

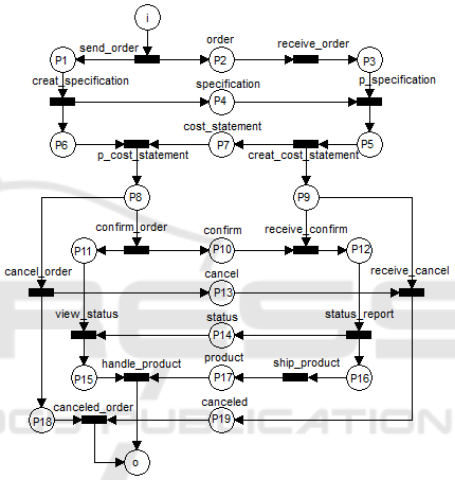

Figure 3: Public Model (requirement).

The P2P approach presented by Aalst (van der Aalst, 2003), ensures that a IOWF-net obtained from a private WF-nets respecting the construction rules presented by Aalst should be deadlock-freeness (i.e. sound). In particular, such rules restrain the design patterns allowed for the construction of private models and are based on the concept of branching bissimilarity presented in subsection 2.3. But, in practice, organizations build theirs workflow processes without worrying too much about rules (Fahland et al., 2011). Considering this fact, Passos and Julia (Passos and Julia, 2014) presented an approach that verifies if the main business relationships between involved organizations can be provided safely. The approach identifies deadlock-freeness scenarios in interorganizational workflow processes modeled by IOWF-nets through the use of Linear Logic proof trees. In particular, the approach can be applied to models that respect relaxed versions of the soundness criterion as explained in subsection 2.2.

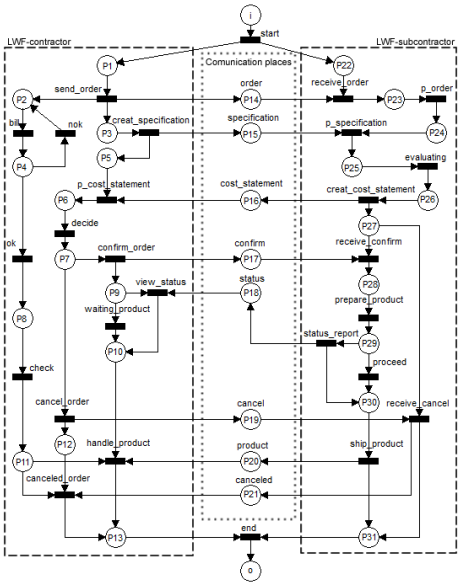

Figure 4: Private Models (architecture) composed in an U(IOWF-net).

Therefore, our approach is based on the ideas presented in (van der Aalst, 2003) and (Passos and Julia, 2014) whose main purpose is to verify, in the context of SOA, if the behavior defined in a requirement model (public model) is also present in the corresponding architectural proposal (private models) without depending on rigid construction rules that ensure the soundness of the process.

3.1 Proposed Method

The approach presented in this work considers that the requirement model corresponds to a public WF-net and the architectural model (type SOA) is the set of private WF-nets that interact through asynchronous communication mechanisms in order to produce the services specified by the requirement model. Therefore, the architectural model corresponds to an IOWF-net. The services of the SOA model are then the scenarios of the IOWF-net. A scenario in the context of a workflow process corresponds to a well defined route mapped into the corresponding WF-net and, if the WF-net has more than one route (places with two or more output arcs), more than one scenario has to be considered then. Because the qualitative analysis of the approach is based on relaxed soundness correction criteria, it is necessary then to identify all deadlock-freeness scenarios in the architectural model proposal that satisfy the scenarios specified in the requirement model.

To introduce the proposed approach, the examples presented in Figures 3 and 4 are used. It is important to note that the U(IOWF-net) of Figure 4 is not sound (the possibility to reach a deadlock situation within the existing scenarios of the model exists); as a consequence, such architectural proposal was not built based on the construction rules proposed by Aalst in (van der Aalst, 2003).

In this work, iterative routes of WF-net models are replaced by simple global tasks, as it is generally the case of hierarchical approaches based on the notion of well formed blocks (Valette, 1979). Figure 5 shows an example of iterative route constraint that exist in the WF-net of Figure 4 (LWF-contractor) and their transformation into single tasks.

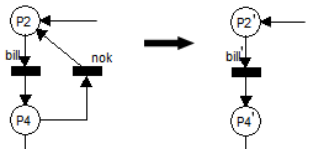

Figure 5: Transformation of iterative route constraints into single tasks.

The proposed method in this work respects the following sequence of steps: (1) build the Linear Logic proof trees for each scenario of the public WF-net correctly executed (requirement model) and transformed the obtained proof trees into the corresponding precedence graphs; (2) build the Linear Logic proof trees for each scenario of the private WF-net composed in an U(IOWF-net) correctly executed (architecture model) and transformed the obtained proof trees into the corresponding precedence graphs; (3) verify the equivalence between the precedence graphs of the public model and the precedence graphs of the private model.

Because the precedence graphs show in a formal way the sequencing constraints of a set of activities performed by a WF-net, they can be seen as a king of operational semantic associated to a workflow process. The equivalence between two WF-nets (the public and the private one) can consequently be verified using the branching bisimilarity concept presented in subsection 2.3 whose purpose is to compare operational semantics of distinct formal behavioral models. In our approach, the activities of the private model that do not appear in the public model are then considered as silent actions. When removing the silent actions of the precedence graphs of the private models, the obtained reduced graphs should be the same of the public WF-net when the behavior of the architectural model reproduces the behavior of the requirement model.

3.2 Method Application

To illustrate the approach, we use the examples presented in Figures 3 and 4. The transitions of the mod-

els are named according to the initial letters of the activities associated to them. For example, *send_order* and *receive_order* are represented by t_{so} and t_{ro}, respectively. The transitions (activities) of the requirement model (WF-net of Figure 3) are represented by the following formulas of Linear Logic:

$t_{so} = i \multimap P_1 \otimes P_2$, $t_{ro} = P_2 \multimap P_3$,
$t_{cs} = P_1 \multimap P_4 \otimes P_6$, $t_{ps} = P_3 \otimes P_4 \multimap P_5$,
$t_{pcs} = P_6 \otimes P_7 \multimap P_8$, $t_{ccs} = P_5 \multimap P_7 \otimes P_9$,
$t_{cfo} = P_8 \multimap P_{10} \otimes P_{11}$, $t_{rcf} = P_9 \otimes P_{10} \multimap P_{12}$,
$t_{sr} = P_{12} \multimap P_{14} \otimes P_{16}$, $t_{vs} = P_{11} \otimes P_{14} \multimap P_{15}$,
$t_{co} = P_8 \multimap P_{13} \otimes P_{18}$, $t_{rc} = P_9 \otimes P_{13} \multimap P_{19}$,
$t_{sp} = P_{16} \multimap P_{17}$, $t_{cdo} = P_{18} \otimes P_{19} \multimap o$,
$t_{hp} = P_{15} \otimes P_{17} \multimap o$.

The requirement model contains two scenarios called respectively Sr1 and Sr2. Before the construction of the precedence graph, it is necessary to prove the sequent corresponding to each scenario of the public WorkFlow net are syntactically correct in accordance with Linear Logic theory.

Considering the scenario Sr1 of the public WorkFlow net of Figure 3, the following sequent needs to be proven:

$i, t_{so}, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o$.

The corresponding proof tree for scenario Sr1 is the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{i \vdash i}\quad \dfrac{}{P_1 \otimes P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L}{i, i \multimap P_1 \otimes P_2, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L}\quad \dfrac{\dfrac{}{P_3 \vdash P_3}\quad \dfrac{}{P_4 \vdash P_4}}{P_3, P_4 \vdash P_3 \otimes P_4}\otimes_R}{P_6, P_5, P_5 \multimap P_7 \otimes P_9, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L}{P_3, P_4, P_6, P_3 \otimes P_4 \multimap P_5, t_{ccs}, t_{pcs}, t_{co}, t_{rcl}, t_{cdo} \vdash o}\otimes_L}{P_1 \vdash P_1 \quad P_3, P_4 \otimes P_6, P_3 \otimes P_4 \multimap P_5, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L}{P_2 \vdash P_2 \quad P_1, P_3, P_1 \multimap P_4 \otimes P_6, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L}{P_1, P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L}{i \vdash i \quad P_1 \otimes P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L}{i, i \multimap P_1 \otimes P_2, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L$$

To generate the precedence graph, the proof tree must be labeled as explained in subsection 2.2. For reasons of space, only part of the proof trees presented in this work are labeled. For example, for the proof tree of scenario Sr1, part of the labeling is the following one:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{i \vdash i}\quad \dfrac{}{P_1 \otimes P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L^{t_{so}}}{i, i \multimap P_1 \otimes P_2, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L^{t_{so}}\quad \dfrac{\dfrac{}{P_3 \vdash P_3}\quad \dfrac{}{P_4 \vdash P_4}}{P_3, P_4 \vdash P_3 \otimes P_4}\otimes_R}{P_6, P_5, P_5 \multimap P_7 \otimes P_9, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L^{t_{ro}}}{P_3, P_4, P_6, P_3 \otimes P_4 \multimap P_5, t_{ccs}, t_{pcs}, t_{co}, t_{rcl}, t_{cdo} \vdash o}\otimes_L^{t_{ps}}}{P_1 \vdash P_1 \quad P_3, P_4 \otimes P_6, P_3 \otimes P_4 \multimap P_5, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L^{t_{cs}}}{P_2 \vdash P_2 \quad P_1, P_3, P_1 \multimap P_4 \otimes P_6, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L^{i_i}}{P_1, P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L^{t_{so}\ t_{so}}}{i \vdash i \quad P_1 \otimes P_2, P_2 \multimap P_3, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\multimap_L^{i_i}}{i, i \multimap P_1 \otimes P_2, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{co}, t_{rc}, t_{cdo} \vdash o}\otimes_L^{t_{so}\ t_{so}}$$

The complete precedence graph of scenario Sr1 for the requirement model is presented in Figure 6. In this graph, the vertices represented the activities, and the arcs the conditions that activate the activities. This is noted, in particular, as the formal specification of the requirement (in terms of the behavior) expected from the SOA where such a service will be implemented. It represents too a possible view of the operational semantic associated to the corresponding workflow process.

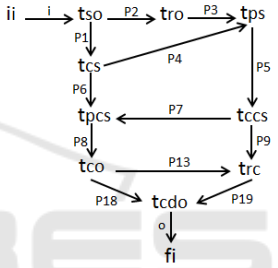

Figure 6: Precedence graph for scenario Sr1.

Considering the scenario Sr2 of the public WorkFlow net of Figure 3, the following sequent needs to be proven:

$i, t_{so}, t_{ro}, t_{cs}, t_{ps}, t_{ccs}, t_{pcs}, t_{cfo}, t_{rcf}, t_{sr}, t_{vs}, t_{sp}, t_{hp} \vdash o$

For reasons of space, we do not show the proof trees for scenario Sr2. The precedence graph for scenario Sr2 of the requirement model is presented in Figure 7.

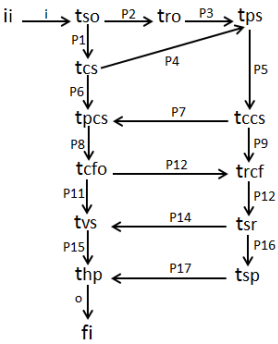

Figure 7: Precedence graph for scenario Sr2.

The U(IOWF-net) of Figure 4 (architectural model) contain four scenarios called respectively Sa1, Sa2, Sa3 and Sa4. The transitions of the architecture

model are represented by the following formulas of Linear Logic:
$t_s = i \multimap P_1 \otimes P_{22}$, $t_{so} = P_1 \multimap P_2 \otimes P_3 \otimes P_{14}$,
$t_{ro} = P_{22} \otimes P_{14} \multimap P_{23}$, $t_{po} = P_{23} \multimap P_{24}$,
$t_b = P'_2 \multimap P'_4$, $t_{ok} = P'_4 \multimap P_8$, $t_c = P_8 \multimap P_{11}$,
$t_{cs} = P_3 \multimap P_5 \otimes P_{15}$, $t_{ps} = P_{15} \otimes P_{24} \multimap P_{25}$,
$t_e = P_{25} \multimap P_{28}$, $t_{pcs} = P_5 \otimes P_{16} \multimap P_6$,
$t_{ccs} = P_{26} \multimap P_{16} \otimes P_{27}$, $t_d = P_6 \multimap P_7$,
$t_{co} = P_7 \multimap P_{12} \otimes P_{19}$, $t_{rc} = P_{27} \otimes P_{19} \multimap P_{21} \otimes P_{31}$,
$t_{cdo} = P_{11} \otimes P_{12} \otimes P_{21} \multimap P_{13}$, $t_p = P_{29} \multimap P_{30}$,
$t_{cfo} = P_7 \multimap P_9 \otimes P_{17}$ $t_{rcf} = P_{17} \otimes P_{27} \multimap P_{28}$
$t_{vs} = P_9 \otimes P_{18} \multimap P_{10}$, $t_{sr} = P_{29} \multimap P_{18} \otimes P_{30}$,
$t_{wp} = P_9 \multimap P_{10}$, $t_{hp} = P_{11} \otimes P_{10} \otimes P_{20} \multimap P_{13}$,
$t_{sp} = P_{30} \multimap P_{20} \otimes P_{31}$, $t_{end} = P_{13} \otimes P_{31} \multimap o$.

Considering each scenario of U(IOWF-net) of Figure 4, the following sequents needs to be proven:

- (scenario Sa1) $i, t_s, t_{so}, t_{ro}, t_b, t_{cs}, t_{ok}, t_c, t_{po}, t_{ps}, t_e, t_{ccs}, t_{pcs}, t_d, t_{cfo}, t_{rcf}, t_{wp}, t_{pp}, t_{tp}, t_{sp}, t_{hp}, t_{end} \vdash o$
- (scenario Sa2) $i, t_s, t_{so}, t_{ro}, t_b, t_{cs}, t_{ok}, t_c, t_{po}, t_{ps}, t_e, t_{ccs}, t_{pcs}, t_d, t_{cfo}, t_{rcf}, t_{vs}, t_{pp}, t_{sr}, t_{sp}, t_{hp}, t_{end} \vdash o$
- (scenario Sa3) $i, t_s, t_{so}, t_{ro}, t_b, t_{cs}, t_{ok}, t_c, t_{po}, t_{ps}, t_e, t_{ccs}, t_{pcs}, t_d, t_{co}, t_{rc}, t_{cdo}, t_{end} \vdash o$
- (scenario Sa4) $i, t_s, t_{so}, t_{ro}, t_b, t_{cs}, t_{ok}, t_c, t_{po}, t_{ps}, t_e, t_{ccs}, t_{pcs}, t_d, t_{cfo}, t_{rcf}, t_{wp}, t_{pp}, t_{sr}, t_{sp}, t_{hp}, t_{end} \vdash o$

For reasons of space, we do not show the proof trees for scenarios Sa1, Sa2 and Sa3. The precedence graph for scenario Sa1 is presented in Figure 8. In this graph, all sequences of activities performed in the U(IOWF-net) are clearly specified.

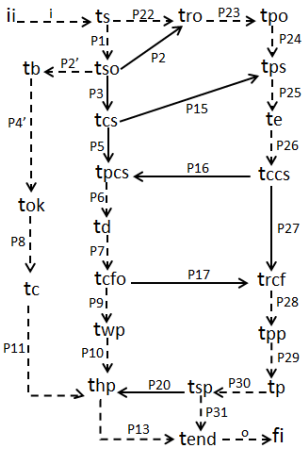

Figure 8: Precedence graph for scenario Sa1.

In the precedence graph for the private model, dashed lines are used when an arc is a link to a silent activity (one of its vertices corresponds to a silent activity). As a matter of fact, these activities are of interest only to their respective private WF-net. By removing a silent action from the precedence graph for scenario Sa1, it is necessary to connect the precedent activity that is connected to the silent activity to the successor activity of the silent activity. For example, by removing the activity t_d in Figure 8, a new directed arc is created between the activities t_{pcs} and t_{cfo}. By removing all the silent activities from the precedence graph in Figure 8, the precedence graph of Figure 9 is obtained.

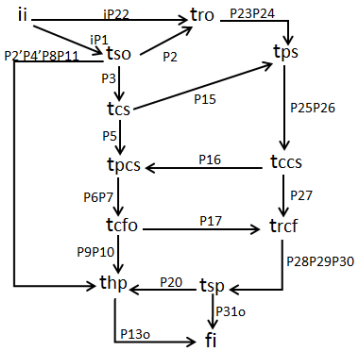

Figure 9: Reduced precedence graph for scenario Sa1.

The precedence graph for scenario Sa2 is presented in Figure 10. By removing all the silent activities from the precedence graph in Figure 10, the precedence graph of Figure 11 is obtained.

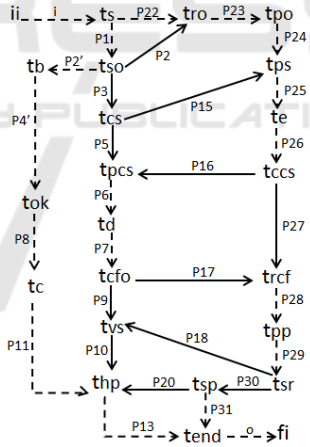

Figure 10: Precedence graph for scenario Sa2.

The precedence graph for scenario Sa3 is presented in Figure 12. By removing all the silent activities from the precedence graph in Figure 12, the precedence graph of Figure 13 is obtained.

The proof tree corresponding to scenario Sa4 is the following one (because of space limitation, only the beginning and the end of the proof are presented):

$$\dfrac{\dfrac{P_{31} \vdash P_{31} \quad P_{13} \vdash P_{13}}{P_{31}, P_{31} \vdash P_{31} \otimes P_{13}} \otimes_R \qquad P_{18}, o \vdash o}{\vdots} \multimap_L$$
$$\vdots$$
$$i, i \multimap P_1 \otimes P_{22}, t_{so}, t_{ro}, t_b, t_{cs}, t_{po}, t_{ps}, \ldots, t_{sr}, t_{sp}, t_{hp}, t_{end} \vdash o$$

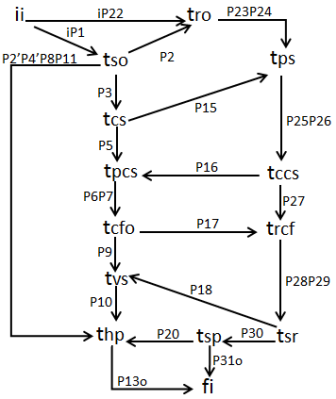

Figure 11: Reduced precedence graph for scenario Sa2.

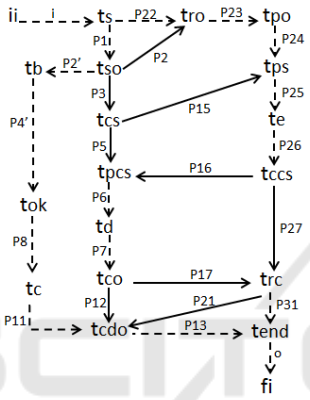

Figure 12: Precedence graph for scenario Sa3.

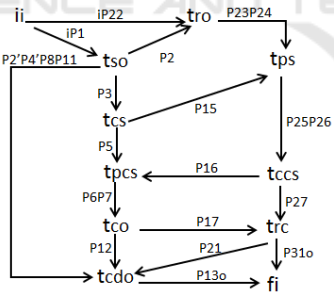

Figure 13: Reduced precedence graph for scenario Sa3.

The last line of the proof tree clearly shows that even with the last place 'o' of the U(IOWF-net) marked, another token is present in the process in P_{18}. This means that scenario Sa4, in case it is executed, will produce a kind of information duplication; such a scenario does not correspond then to a sound behavior and will not be considered as part of the scenarios necessary to cover the business requirement of the public model. As a direct consequence, the precedence graph for scenario Sa4 is not produced.

The last step of the approach is to compare the precedence graphs of the public model with the precedence graphs of the private one to verify if the requirements specified in term of behavior in the public WorkFlow net are also present in the architectural proposal specified in the private WorkFlow nets composed in a U(IOWF). The precedence graphs obtained from scenarios Sr1 and Sr2 have then to be compared with precedence graphs obtained from scenarios Sa1, Sa2 and Sa3.

The precedence graph for scenario Sr1 (Figure 6) and the reduced precedence graph for scenario Sa3 (Figure 13) execute the same activities respecting the same sequential constraints. The additional arcs iP_{22}, $P_2'P_4'P_8P_{11}$ and $P_{31}o$ that exist in the graph of Figure 13, and that do not exist in the graph of Figure 6 are simply redundant constraints that can be removed without modifying the requirement specification. For example, the arc $P_2'P_4'P_8P_{11}$ of Figure 13 simply states that the activity t_{so} has to happen before activity t_{cdo}. However, this statement already exists through the sequence of arcs P_3, P_5, P_6P_7, P_{12} for example. By removing the redundant arcs from the graph of Figure 13, the precedence graphs of Figure 6 and 13 are exactly the same.

The precedence graph for scenario Sr2 (Figure 7) and the reduced precedence graph for scenario Sa2 (Figure 11) execute the same activities respecting the same sequential constraints. By removing the redundant arcs from the graph in Figure 11, the precedence graphs of Figure 7 and 11 are exactly the same.

The scenario Sa1 produces a behavior that was not specified in the public model. This fact is predictable since the architecture model is a set of private WF-net that contains several tasks which are only of local interest and which do not appear in the public WF-net model.

Although the architectural model (Figure 4) does not hold deadlock-freeness, the existing scenarios (Sa1, Sa2 and Sa3) are correctly executed and verify the requirements specified in the public model. According to the definition presented in subsection 2.2, we can verify that the model is relaxed sound, i.e each transaction (activity) appears in, at least, one scenario that finishes correctly. Therefore, even though an architectural model (type SOA) does not hold deadlock-freeness, it is possible to verify if, at least, it possesses the scenarios that correctly satisfy the business needs defined in the analysis model. For the example shown in this work, one concludes that the requirements defined in the analysis model are well defined in the architectural model. The requirements present in scenarios Sr1 and Sr2 of the analysis model are present respectively in the scenario Sa3 and Sa2 of the architectural model.

4 CONCLUSION

This paper presented an approach for requirement verification in SOA models based on Interorganizational WorkFlow nets relaxed sound and Linear Logic. Our purpose was to show that, in context of SOA, all scenarios present in a requirement model (public model) are also present in the corresponding architectural model (private models). This approach was based in particular on the construction of Linear Logic proof trees and of precedence graphs that show the operational semantic of distinct models. With the precedence graph it was possible to compare and check the behavioral equivalence between the public and private models, in particular if the models simulate each other's behavior, respecting the notion of branching bisimilarity. In this approach, precedence graphs are only built for the sequents of Linear Logic syntactically correct, i.e deadlock-freeness scenarios that ended correctly the modeled business process.

One of the main advantage of this approach is to define, through the use of Linear Logic and precedence graphs, a new kind of operational semantic associated to business processes that allows to verify in a formal way business requirements within the SOA context. The presented approach considers architectural models (type SOA) not necessarily sound. The organizations do not have then to be constrained by external actors to build their private workflow processes, as is the case in practical when considering existing enterprise systems. Therefore, the organizations involved in interorganizational workflow processes can simply verify if the set of requirement scenarios of an analysis model are present in an available SOA model candidate for the implementation of the required service. In this sense, the impacts and deviations generated by collaboration between different organizations can be minimized.

In this article, only a kind of functional requirement was verified in the models. As a future work proposal, we will associate explicit time constraints to the models to evaluate the performance of the models and apply a kind of quantitative analysis in the context of SOA.

REFERENCES

Basten, T. (1998). *In Terms of Nets System Design With Petri Nets and Process Algebra*. PhD thesis, Eindhoven University of Technology, Eindhoven, Netherlands.

Diaz, M. (2009). *Petri Nets: Fundamental Models, Verification and Applications*. Wiley-ISTE, Reading, Massachusetts.

Fahland, D., Favre, C., Koehler, J., Lohmann, N., Volzer, H., and Wolf, K. (2011). Analysis on demand: Instantaneous soundness checking of industrial business process models. *Data Knowledg Engineering*, pages 448–466.

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science*, pages 1–102.

Girault, F., Pradin-Chezalviel, B., and Valette, R. (1997). A logic for petri nets. *Journal européen des systèmes automatisés*.

Goknil, A., Kurtev, I., and Berg, K. V. D. (2014). Generation and validation of traces between requirements and architecture based on formal trace semantics. *Journal of Systems and Software*, pages 112 – 137.

Park, D. (1981). Concurrency and automata on infinite sequences. In *5th GI-Conference on Theoretical Computer Science*, pages 167–183. Springer Verlag, Berlin, Germany.

Passos, L. M. S. (2016). *A Metodology based on Linear Logic for Interorganizational Workflow Processes Analysis*. PhD thesis, Federal Univerty of Uberlandia.

Passos, L. M. S. and Julia, S. (2013). Qualitative analysis of interorganizational workflow nets using linear logic: Soundness verification. In *IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 667–673.

Passos, L. M. S. and Julia, S. (2014). Linear logic as a tool for deadlock-freeness scenarios detection in interorganizational workflow processes. In *IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 316–320.

Riviere, N., Pradin-Chezalviel, B., and Valette, R. (2001). Reachability and temporal conflicts in t-time petri nets. In *9th international Workshop on Petri Nets and Performance Models*.

Valette, R. (1979). Analysis of petri nets by stepwise refinements. *Journal of Computer and System Sciences*, pages 35–46.

van der Aalst, W. M. P. (1996). Structural characterizations of sound workflow nets. Computing science reports/23, Eindhoven University of Technology.

van der Aalst, W. M. P. (1998). Modeling and analyzing interorganizational workflows. In *International Conference on Application of Concurrency to System Design*, pages 262–272. IEEE Computer Society Press.

van der Aalst, W. M. P. (2000). Loosely coupled interorganizational workflows: modeling and analyzing workflows crossing organizational boundaries. *Information and Management*, pages 67–75.

van der Aalst, W. M. P. (2003). Inheritance of interorganizational workflows: How to agree to disagree without loosing control? *Information Technology and Management*, pages 345–389.

van Glabbeek, R. J. and Weijland, W. P. (1996). Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, pages 555–600.

Zernadji, T., Tibermacine, C., Cherif, F., and Zouioueche, A. (2015). Integrating quality requirements in engineering web service orchestrations. *Journal of Systems and Software*, pages 463 – 483.