# Inconsistency recovery in Business Processes using a possibilistic WorkFlow net

Leiliane Pereira de Rezende*, Stéphane Julia* and Janette Cardoso†
*Faculdade de Computação - Universidade Federal de Uberlândia, UFU
2160, av. João Naves de Ávila, 38400–902 Uberlândia/MG, Brazil
Email: leily.rezende@gmail.com, stephane@facom.ufu.br
†Institut Supérieur de l'Aéronautique et de l'Espace,ISAE
10, av. Édouard Belin, 31055 Toulouse, France
Email: cardoso@isae.fr

***Abstract*—In this paper, an approach based on WorkFlow nets and on possibilistic Petri nets is proposed to deal with non-conformance in Business Processes. Routing patterns existing in Business Processes are modeled by WorkFlow nets. To express in a more realistic way the uncertainty attached to human activities, possibilistic Petri nets with uncertainty on the marking and on the transition firing are considered. Combining both formalisms, a kind of possibilistic WorkFlow net is obtained. An example of inconsistency recovery at a process monitoring level due to human behavior in a "Handle Complaint Process" is presented.**

***Keywords*-WorkFlow net, possibilistic Petri net, process non-conformance, process monitoring.**

## I. Introduction

The purpose of Workflow Management Systems is to execute Business Processes. Over the last few years, Business Process Management has become important in order to raise service quality and performance of firms [1].

Many papers [2] have already considered Petri net theory as an efficient tool for the modeling and analysis of Workflow Management Systems. In [2], WorkFlow nets, which are acyclic Petri net models used to represent Business Processes, are defined. The main property that has to be proven when considering WorkFlow nets is the soundness property [2] which guarantees that no deviation from the process description will be allowed during the real time execution.

In fact, good properties of well defined formal models such as WorkFlow nets can easily be proven when Business Processes are following a rigid structure that does not allow deviations from the process description during the real time execution. However, recently, it was shown that Business Processes do not easily map to a rigid modeling structure. Some of the activities executed in a Business Process depend on human resources that do not necessarily respect the rigorous definition of WorkFlow Processes, due to the fact that tasks performed by humans are generally complex and follow rules that cannot always be transformed into computerized processes. As a matter of fact, in practice inconsistencies between the process model and the real execution of the process can easily occur.

Attempts to consider a certain level of flexibility in process definition have already been proposed by several authors.

In [1], the Yawl language which supports flexibility in the process definition is proposed. The principle is based on Worklet Services which allows for the construction of subprocess structures in such a way that they can be added dynamically to the whole Workflow Process during the real time execution. The possible process deviations are designed in an explicit way (additional routing structures in the model) and, as such, no guarantee of soundness in the process exists.

In [3], a deviation-tolerant approach in process execution is presented. The basic principle is based on two models coexisting during the monitoring of the process. The first one corresponds to the expected behavior and the second is dynamically built, based on real human actions. The two models are then continuously compared to detect possible deviations. The problem in dealing with two models is that the monitoring activity can easily be overloaded implying a decrease in the system's performance.

In [4], a kind of declarative implicit model, essentially based on rules, is used in order to detect non-conformant states. In such a methodology, the process model can only be seen as a simple set of constraints that can not be analysed from the point of view of the soundness property.

In [5] and [6], a process model based on temporal logic and finite state machines to capture and tolerate deviations in processes during execution is defined. For the authors, a process is correct if all the constraints given by the set of state machines are verified. In particular, two kinds of transition are created: normal ones and exported ones which depend on user requests to indicate abnormal behavior. In particular, when the process execution is corrupted, the state of the process is fixed manually. The problem with this kind of approach is again that the model of the process is given through a declarative form instead of a single graph representing a whole process that could be analyzed from the point of view of the soundness property, as is the case with WorkFlow nets. Another problem is the necessity for explicitly model alternative scenarios corresponding to abnormal behavior. The consequence is generally the increase of the

complexity in the set of constraints and on the underlying process model.

In [7], [8] and [9], an incremental approach to check the conformity of a process model and an event log is presented. Initially it evaluates if all the sequences recorded in log are possible execution sequences in regards to the process model. Then the accuracy between the process model and the expected behavior is verified. Following this, the analyst is assisted in finding the areas that are in non-conformance with the process (in the model or in the log file). In [10] the implementation of an algorithm to calculate several conformance metrics is proposed. The problem with this approach is that the verification is carried out after the process execution.

A very promising alternative to deal with non-conformance in Business Processes seems to be approaches based on uncertain knowledge as that presented in [11]. The model of the process is then given through fuzzy sets and possibilistic distributions that permit a natural representation of uncertain and imprecise information that exists when human type resources are involved in the activities of a process.

One of the first studies which combines possibilistic representation of information with the precise structure of a Petri net when considering discrete event systems is that described in [12] and [13]. The main feature of possibilistic/fuzzy Petri nets is to allow one to reason about the aspects of uncertainty and change in dynamic discrete event systems. Most of the examples presented by the authors of possibilistic Petri net were applied to flexible manufacturing systems.

In this paper, an approach based on WorkFlow nets and possibilistic Petri nets is proposed to deal with non-conformance in Business Processes. In particular, a kind of possibilistic WorkFlow net will be defined to treat non-conformant states.

In section II, the definition of WorkFlow nets and soundness correctness criterion are provided. In section III, the definition of the objects Petri nets is presented. In section IV, an overview of possibilistic Petri nets is given. In section V, the possibilistic WorkFlow net and the algorithm for inconsistency recovery are defined and an example based on a "Handle Complaint Process" illustrates the approach. Finally, section VI concludes this work with a short summary, an assessment based on the approach presented and an outlook on future work proposals.

## II. WorkFlow Nets

A Petri net that models a Workflow Process is called a WorkFlow net [2], [14]. A WorkFlow net satisfies the following properties [14]:

- It has only one source place, named *Start* and only one sink place, named *End*. These are special places such that the place *Start* has only outgoing arcs and the place *End* has only incoming arcs.
- A token in *Start* represents a case that needs to be handled and a token in *End* represents a case that has been handled.
- Every task t (transition) and condition p (place) should be on a path from place *Start* to place *End*.

Soundness is a correctness criterion defined for WorkFlow nets. A WorkFlow net is sound if, and only if, the following three requirements are satisfied [2]:

- For each token put in the place *Start*, one and only one token appears in the place *End*.
- When the token appears in the place *End*, all the other places are empty for this case.
- For each transition (task), it is possible to move from the initial state to a state in which that transition is enabled, i.e. there are not any dead transitions.

A method for the qualitative analysis of WorkFlow nets (soundness verification) based on the proof trees of linear logic is presented in [15].

### A. Process

A process defines which tasks need to be executed and in which order [14]. Modeling a Workflow Process in terms of a WorkFlow net is rather straightforward: transitions are active components and model the tasks, places are passive components and model conditions (pre and post), and tokens model cases [2], [14].

To illustrate the mapping of a process into a WorkFlow net, the process for handling complaints that is shown in [2] can be considered as follows: an incoming complaint is first recorded. Then the client who has complained and the department affected by the complaint are contacted. The client is approached for more information. The department is informed of the complaint and may be asked for its initial reaction. These two tasks may be performed in parallel, i.e. simultaneously or in any order. After this, the data is gathered and a decision is made. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed. Fig. 1(a) shows a WorkFlow net that correctly models this process.

### B. Routing constructs

Tasks can be optional, i.e. there are tasks that just need to be executed for some cases, and the order in which tasks will be executed can vary from case to case [2]. Four basic constructions for routing are presented in [2] and [14]:

- *Sequential*: tasks are executed one after another sequentially, clearly demonstrating dependence among these tasks: one needs to finish for the other to start;
- *Parallel*: if more than one task can be executed simultaneously or in any order. In this case, both tasks can be executed without the result of one interfering in the result of the other;
- *Conditional* (or selective routing): when there is a choice between two or more tasks;

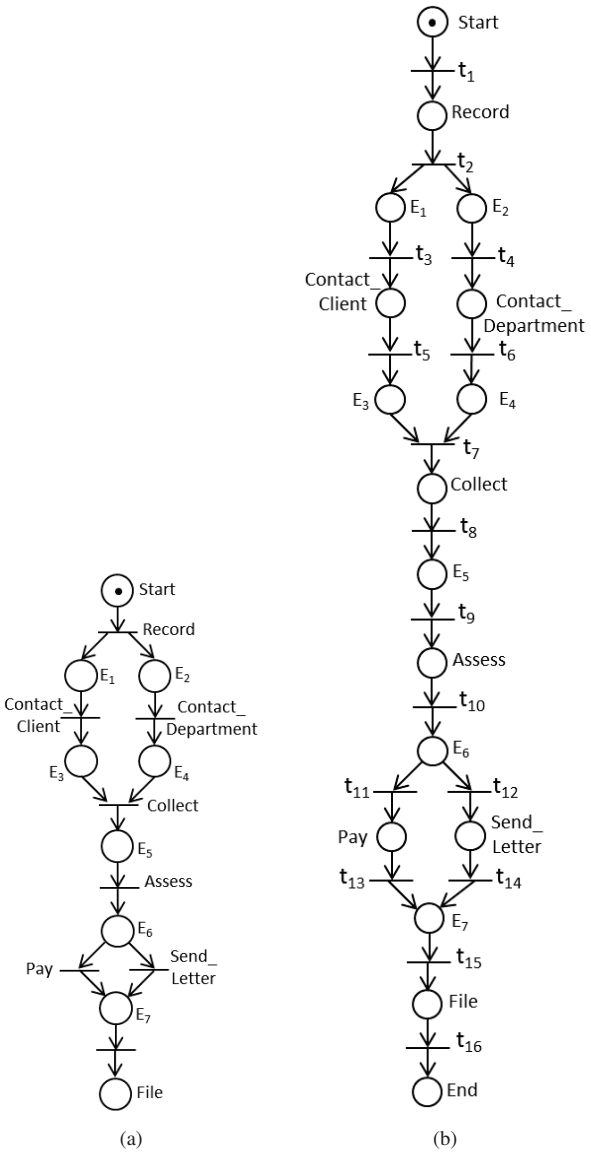


Figure 1. Handle Complaint Process: (a)Tasks are associated directly to simple transitions. (b)Tasks are associated directly to places.

- *Iterative*: when it is necessary to execute the same task multiple times.

Some variations of these four basic constructions can be found in [2] and [14].

Considering the "Handle Complaint Process" shown in Fig. 1(a), tasks "Contact Client" and "Contact Department" are an example of parallel routing. Tasks "Collect" and "Assess" are an example of sequential routing. And tasks "Pay" and "Send Letter" are an example of conditional routing.

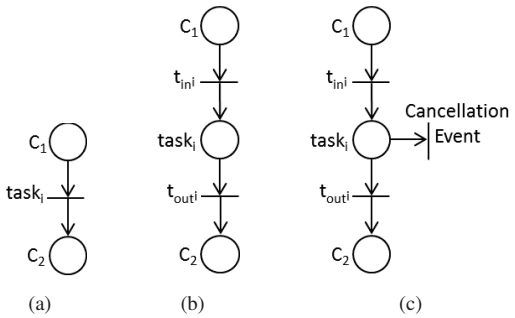


Figure 2. (a) Traditional WorkFlow net; (b) WorkFlow net with explicit task execution; and (c) WorkFlow net with cancellation event.

### C. Process Monitoring

In [1], WorkFlow nets were revisited in terms of their suitability for monitoring Business Processes. The authors showed that some patterns were not easily captured, in particular patterns dealing with cancellation and multiple concurrently executing instances of the same task.

The principal reason of limitation existing in WorkFlow nets for monitoring Business Processes is the fact that tasks are associated directly with simple transitions. As a consequence, once initiated, a task cannot be interrupted because it corresponds to the firing of a transition. If during the execution of a task, an event occurs in the system whose purpose is to interrupt the whole process, in traditional WorkFlow nets the current tasks of the process have to be completed first to be able to accept the cancellation. Of course, a proper model of the process should be able to accept interruption events in an asynchronous way in order to monitor the process in an efficient way.

The solution proposed in this work to consider a more realistic monitoring model of a Business Process is to transform the transitions of WorkFlow net into a structure based on the following pattern: a block corresponding to a task of a transition $t_i$ is composed of a place $P_{t_i}$ which represents the task $t_i$, an input transition $t_{in\,i}$ which represents the beginning of the task execution, and an output transition $t_{out\,i}$ which represents the end of the task execution.

The WorkFlow net of Fig. 2(a) will then be transformed into the Workflow Process given by the acyclic Petri net model of Fig. 2(b). As the new block (corresponding to the task in execution) can be substituted by a simple transition preserving the good properties of the initial model [16], the new process model will continue sound in most cases and will be adapted for monitoring activities, in particular if some events of cancellation need to be specified as shown in Fig. 2(c).

Finally, the Petri net model of Fig 1(b), corresponding to the WorkFlow net of Fig. 1(a), will then be produced.

## III. Objects Petri Nets

Ordinary Petri nets do not allow for the modeling of complex data structures. Many extensions have been proposed to model this specific aspect through high-level Petri net definitions.

The object Petri nets defined by Sibertin-Blanc [17] are based on the integration of predicate/transition Petri nets and the concept of an object oriented paradigm. The tokens are considered as *n-tuples* of instances for a class of objects and carries data structures defined as sets of attributes for specific classes. Pre-conditions and actions are associated with transitions, which respectively act on the attributes (eventually modifying their values) of the data structures transported by the tokens of the net. The object Petri nets can be formally defined as:

A marked Object Petri net can be defined by the 9-tuple:

$$N_0 =< P, T, C_{lass}, V, Pre, Post, A_{tc}, A_{ta}, M_0 > \quad (1)$$

where:

- $C_{lass}$ is a finite set of classes of objects: for each class a set of attributes is also defined;
- $P$ is a finite set of places whose types are given by $C_{lass}$;
- $T$ is a finite set of transitions;
- $V$ is a set of variables whose types are given by $C_{lass}$;
- $Pre$ is the function precedent place (an arc between a place and a transition which considers a formal sum of elements of $V$);
- $Post$ is the next function place (an arc between a transition and a place which considers a formal sum of elements of $V$);
- $A_{tc}$ is an application which associates to each transition a condition that involves the attributes of the formal variables associated with the input arcs of the transitions;
- $A_{ta}$ is an application which associated to each transition an action that involves the formal attributes of the variables associated with the input arcs of the transition and updates the attributes of the formal variables associated with the transitions' output arcs;
- $M_0$ is the initial marking which associates a formal sum of objects to each place (n-tuples of instances of classes that belong to $C_{lass}$);

An example of object Petri net is presented in Fig. 3. The set of classes is defined as:

$$C_{lass} = \{Product, Request\}$$

The $Product$ class has the attributes:

$$\begin{cases} name = identifier; \\ code = integer; \\ cost = float; \end{cases}$$

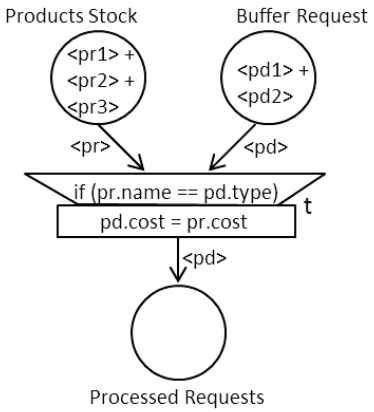


Figure 3. Specification of a Sale Transaction.

The $Request$ class has the attributes:

$$\begin{cases} code : integer; \\ cost : float; \\ type : identifier; \end{cases}$$

The variable $pr$ belongs to the class $Product$ and the variable $pd$ belongs to the $Request$ class. The place $Products\ Stock$ belongs to the $Product$ class, the place $Buffer\ Request$ belongs to the $Request$ class and the place $Processed\ Requests$ belongs to the $Request$ class. The initial marking $M_0$ is given by the objects that are in the places $Products\ Stock$ and $Buffer\ Request$ and is given by:

$$M_0 = \begin{bmatrix} < pr1 > + < pr2 > + < pr3 >, \\ < pd1 > + < pd2 >, \\ 0 \end{bmatrix}$$

For example, the attributes of the object (token) $pr1$ can be given by:

$$\text{Product pr1} \begin{cases} name : hometheater; \\ code : 567544; \\ cost : 278,50; \end{cases}$$

and the attributes of the object (token) $pd2$ can be given by:

$$\text{Request pd2} \begin{cases} code : 123440; \\ cost : 00,00; \\ type : hometheater; \end{cases}$$

The detailed definition of the dynamic behavior (firing rules) of the object Petri Net can be found in [17]. In Fig. 3, the transition $t$ is enabled by the initial marking. The attributes of the variable $pr$ associated with the arc connecting the place $Products\ Stock$ to the transition $t$ can be replaced by the attributes of the objects $pr1$ for example. Similarly, the attributes of the variable $pd$ associated with the arc connecting the place $Buffer\ Request$ to transition $t$ can be replaced by the attributed of the objects $pd2$ for

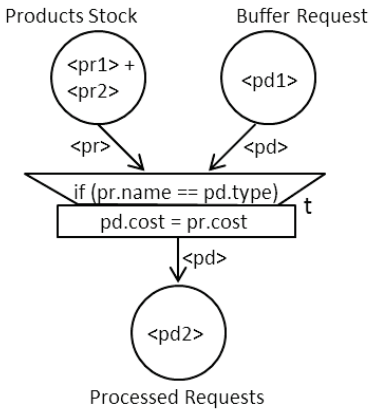


Figure 4. Execution of a Sale Transaction Simulation.

example. Considering that the attributes of the pair of objects *(pr1, pd2)* check the condition associated with the transition $t$, the transition can be fired. The action associated with the transition is then executed and a new object $pd2$ can be produced in the place *Processed Requests*, as shown in Fig. 4, with the following attributes:

$$\text{Request pd2} \begin{cases} code : 123440; \\ cost : 278,50; \\ type : hometheater; \end{cases}$$

In particular, when considering this new object $pd2$, the attribute $cost$ has been modified after the firing of $t$.

## IV. Possibilistic Petri Nets

Possibilistic Petri nets are derived from Object Petri nets [18]. In particular, in the approach presented in [12], a possibilistic Petri net is a model where a marked place corresponds to a possible partial state, a transition to a possible state change, and a firing sequence to a possible behavior. The main advantage in working with possibilistic Petri nets is that it allows for the updating of a system state at a supervisory level with ill-known information without necessarily reaching inconsistent states.

A possibilistic Petri net model associates a possibility distribution $\Pi_o(p)$ to the location of an object $o$, $p$ being a place of the net, thus allowing a possibilistic distribution to then model:

- *A precise marking*: each token is located in only one place (well-known state).
- *An imprecise marking*: each token location has a possibility distribution over a set of places. It cannot be asserted that a token is in a given place, but only that it is in a place among a given set of places.

$\Pi_o(p) = 1$ represents the fact that $p$ is a possible location of $o$, and $\Pi_o(p) = 0$ expresses the certainty that $o$ is not present in place $p$. Formally, a marking in a possibilistic Petri net is then a mapping:

$$M : O \times P \longrightarrow \{0, 1\} \tag{2}$$

where $O$ is a set of objects and $P$ a set of places. If $M(o, p) = 1$, there exists a possibility of having the object $o$ in place $p$. On the contrary, if $M(o, p) = 0$, there exists no possibility of having $o$ in $p$. A marking $M$ of the net allows one to represent:

- *A precise marking*: $M(o, p) = 1$ and $\forall p_i \neq p, M(o, p_i) = 0$.
- *An imprecise marking*: for example, if there exists a possibility at a certain time to have the same object $o$ in two different places, $p_1$ and $p_2$, them $M(o, p_1) = M(o, p_2) = 1$.

A possibilistic marking will correspond in practice to knowledge concerning a situation at a given time.

In a possibilistic Petri net, the firing (certain or uncertain) of a transition $t$ is decomposed into two steps:

- *Beginning of a firing*: objects are put into output places of $t$ but are not removed from its input places.
- *End of a firing*: that can be a firing cancellation (tokens are removed from the output places of $t$) or a firing achievement (tokens are removed from the input places of $t$).

A certain firing consists of a beginning of a firing and an immediate firing achievement. A pseudo-firing that will increase the uncertainty of the marking can be considered only as the beginning of a firing (there is no information to be sure whether the normal event associated with the transition has actually occurred or not). To a certain extent, pseudo-firing is a way of realizing forward deduction.

The interpretation of a possibilistic Petri net is defined by attaching to each transition an authorization function $\eta_{x_1,...,x_n}$ defined as followed:

$$\eta_{x_1,...,x_n} : T \longrightarrow \{False, Uncertain, True\} \tag{3}$$

where $x_1, ..., x_n$ are the variables associated to the incoming arcs of transition $t$ (when considering the underlying Object Petri net).

If $o_1, ..., o_n$ is a possible substitution to $x_1, ..., x_n$ for firing $t$, then several situations can be considered:

- $t$ is not enabled by the marking but the associated interpretation is true; an inconsistent situation occurs and a special treatment of the net is activated;
- $t$ is enabled by a precise marking and the interpretation is true; then a classical firing (with certainty) of an object Petri net occurs;
- $t$ is enabled by a precise marking and the interpretation is uncertain; then the transition is pseudo-fired and the imprecision is increased;
- $t$ is enabled by an uncertain marking; if the interpretation is uncertain, $t$ is pseudo-fired;

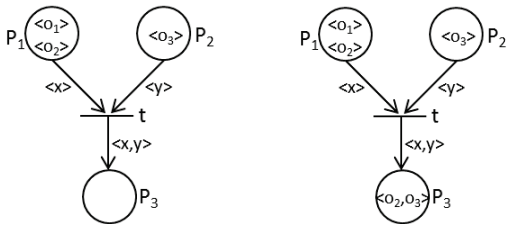


Figure 5. Marking (a) Before firing; and (b) After pseudo-firing.

- $t$ is enabled by an uncertain marking and the interpretation is true: a recovery algorithm, presented in [19], is called and a new computation of the possibility distribution of the objects involved in the uncertain marking is realized in order to go back to a certain marking.

The pseudo-firing (or uncertain marking) is detailed through the example illustrated in Fig. 5. The place $p_1$ belongs to $Class_1$, $p_2$ to $Class_2$ and $p_3$ to the composite class $(Class_1, Class_2)$. The object instances of $Class_1$ have an attribute $date$. The interpretation, given by possibilistic distributions $\eta_{xy}$ is:

$$\forall_y \begin{cases} uncertain & if(\tau < x.date) \wedge (signal(x)) \\ true & if(\tau \geq x.date) \wedge (signal(x)) \\ false & otherwise \end{cases}$$

where $signal(x)$ is $true$ when the associated sensor is active on a specific shop floor.

This function has the following semantics. Before the time $date$, the arrival of a message from the shop floor signaling that the object $< x >$ was involved in the event associated with the transition $t$, is *possible* but does not correspond to a normal behavior. Either the message is erroneous, or the representation of the shop floor state (the Petri net marking) is not consistent with the actual state. The imprecision concerning object $< x >$ will increase and the transition $t$ associated with the corresponding event will be pseudo-fired.

On the other hand, receiving the message after a time $date$ corresponds to normal behavior. So the firing of $t$ should be a normal firing and the update of the shop floor state should be realized with certainty.

Let us consider the initial marking of Fig. 5(a); two substitution are possible for $t$: $S_1 =< o_1, o_3 >$ and $S_2 =< o_2, o_3 >$. Let us assume that $o_1.date = 20$ and $o_2.date = 40$.

At time $\tau_1 = 25$ let us suppose that $signal(o_2) = true$ and at time $\tau_2 = 35$, $signal(o_1) = true$. Fig. 6 depicts the possibility distributions of instances $o_1$, $o_2$ and $o_3$ as a function of time (the black lines represent a possibility equal to 1 and the bright lines a possibility equal to 0):

- at time $\tau = 10$, the firing of transition $t$ is possible in the future for $y = o_3$ and for either $x = o_1$ or $x = o_2$ (transition is enabled and can be fired normally since $signal(o_i)$ is received);
- at time $\tau = 25$, $signal(o_2) = true$ is received but it does not correspond to a normal behavior ($o_2.date > 25$); $\eta_{o_2 o_3}(t) = uncertain$, and $t$ is pseudo-fired with substitution $S_2$ (Fig. 5(b));
- after date $\tau > 25$ the marking is imprecise and cover two alternatives:
  - the event corresponding to the firing of $t$ for tuple $< o_2, o_3 >$ has actually occurred; the information given by $signal(o_2)$ was right;
  - the event corresponding to the firing of $t$ for tuple $< o_2, o_3 >$ has not actually occurred. This transition can still be fired, either by $< o_2, o_3 >$ or by $< o_1, o_3 >$;
- at time $\tau = 35$ the receipt of $signal(o_1) = true$ corresponds to a normal behavior ($o_1.date \leq 35$) and $\eta_{o_1 o_3}(t) = true$. As explained before, this case corresponds to the one in which the recovery algorithm is called. The application of the algorithm *cancels* the pseudo-firing of $t$ for $< o_2, o_3 >$. As the marking is now precise and $\eta_{o_1 o_3}(t) = true$, transition $t$ is fired (with certainty) with the tuple $< o_1, o_3 >$. It assume that $signal(o_2) = true$ was due to noise.

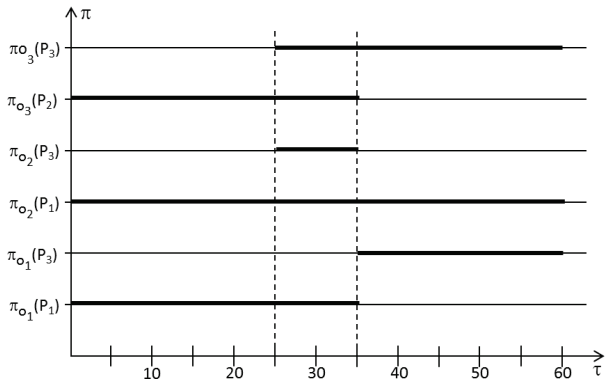


Figure 6. Possibility distribution of locations of $o_1$, $o_2$ and $o_3$.

## V. Possibilistic WorkFlow Nets

If a Petri net is used as a model for Business Processes in a Workflow Management System, transitions will represent the state changes of the process. In particular, each event occurring during the execution of the process (beginning and ending of activities) will be associated with a transition as a boolean variable. Such a variable will be essentially seen as an external value corresponding to a message received from an activity (or send to an activity). Possibly, internal values depending on certain token attributes will enable some transitions too.

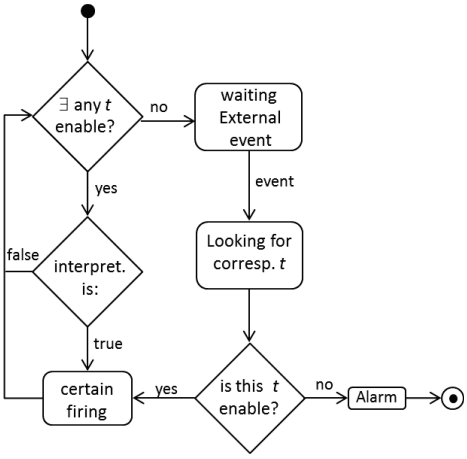


Figure 7. Token player of Petri net.

Petri net models can be directly executed using a specialized inference mechanism called "*token player* algorithm" that allows for a simplified monitoring of the represented processes. As pointed out in the introduction, the interaction of human behavior in Processes Management can introduce some uncertainty and should be taken into account in order to turn the model of the process more robust to human behavior. A classical token player algorithm, as the one in Fig 7, is only based on normal expected events. If an unexpected event occurs, an immediate inconsistency between the model of the process and the real process execution will happen (an external event received by a transition which is not enabled by the marking of the net) or, if some expected event never occurs, the model will reach a deadlock situation (an external event never received by an enabled transition).

A model of the process based on the routing structure of WorkFlow nets and on uncertain marking and firing of possibilistic Petri nets will then produce a kind of possibilistic WorkFlow net that will be able to deal with non-conformance in Business Processes monitoring. The "Handle Complaint Process" represented in Fig. 1(b) will be used to illustrate the approach.

The possibilistic WorkFlow net with objects in Fig. 8 represents the new model of the "Handle Complaint Process" where $< c_1 >$ is an object belonging to the class "Complaint", $x$, $y$ and $z$ are variables of the same class "Complaint" and all places of the model belong to the class "Complaint" too.

After the firing of $t_1$ and the calling of the method $x.record()$ to the corresponding actor (here the human actor that corresponds to the secretary responsible in recording the complaint), an object $< c_1 >$ is produced in place $Record$ and is expecting for the end of the complaint recording that will happen when the corresponding condition $x.endRecord$ associated with the transition $t_2$ will become true.

If the secretary has a certain level of autonomy in making

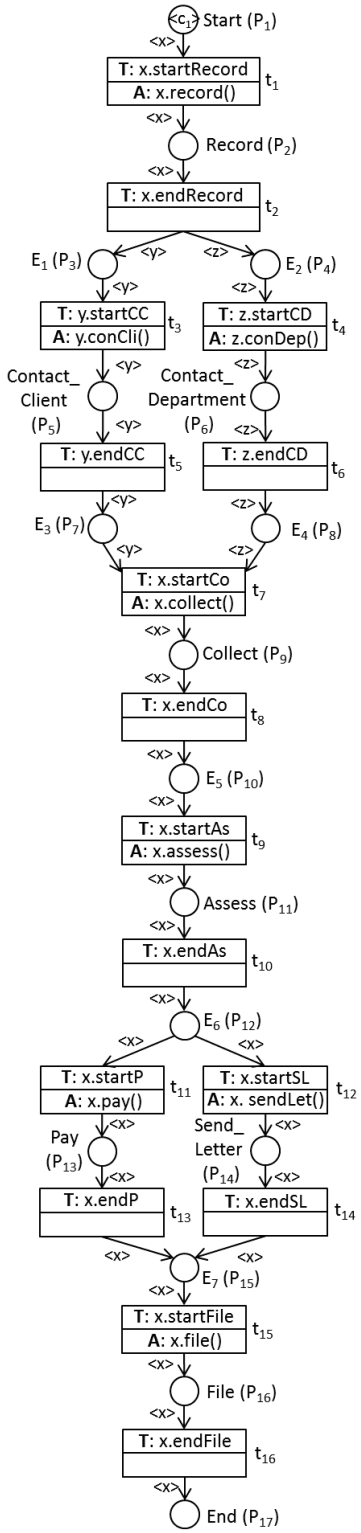


Figure 8. Handle Complaint Process: Possibilistic WorkFlow net.

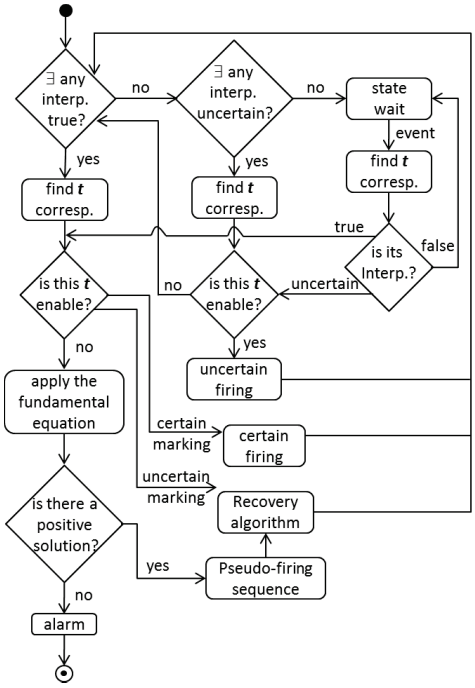


Figure 9. Token player of Possibilistic Petri net.

decisions in function of her own expert knowledge, she can decide that the complaint was not properly submitted for example and send directly a letter to the owner of the complaint without really following the complete set of activities specified by the model of the process. In this case, after sending a letter, instead of having the interpretation associated with the transition $t_2$ ($x.endRecord$) becoming true, it is the interpretation associated with the transition $t_{14}$ ($x.endSL$) that will become true. Because no object will exist in the place *Send_Letter* at this moment, neither transition $t_{14}$ will be enable nor transition $t_2$, the global state of the process resulting then in a deadlock situation. If the *token player* of Fig. 7 is applied to such a situation then an alarm will be activated and a kind of state recovery will have to be realized manually after diagnosing the motive of the corresponding process inconsistency.

If instead of applying a classic *token player*, the *token player* given by the activity diagram of Fig. 9 is considered, then, it will be possible to reach through a sequence of pseudo firings of a possibilistic Petri net a global state consistent with certain external events than can turn true the condition associated with transitions not necessarily enabled by the current marking.

The first step will be to verify through the fundamental equation of the Petri net theory [16] that there exists a non ordered sequence of transitions that permits the reaching of a marking such that the transition that received the external event will be enabled.

Considering the object WorkFlow net in Fig. 10(a) with an object $< c_1 >$ in $P_2$, if an external event is received by transition $t_{14}$, the corresponding equation that has to be solved is the following:

$$M' = M + W \times S \tag{4}$$

with $M'^T = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0]$ the final marking to be reached, $M^T = [0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0]$ the current marking, $W$ the incidence matrix of the WorkFlow net of Fig. 10(a), and $S$ a non ordered sequence of transition firings. The solution is then given by the firing sequence vector $S^T = [0\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0]$ that corresponds to the non ordered sequence $t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}$.

For any Petri net, the result produced by $S$ is only a necessary condition in the reachability problem of the Petri net theory. In the WorkFlow net case that respects the soundness property, $S$ will constitute a sufficient condition too because there does not exist any dead part in the corresponding net.

Once a solution for $S$ is found, it is then necessary to produce at least one ordered sequence of transitions that reaches $M'$ from $M$. The second step of the recovery procedure will be then to make the interpretations associated to all the transitions that occur in the solution $S$ uncertain and make all the other transitions of the object WorkFlow net false. Then, only during the recovery procedure, the interpretation of the transitions will be $\eta_{var}(t_i) = uncertain$ for the transitions $t_i$ that belongs to the non zero solutions of $S$, and $\eta_{var}(t_j) = false$ for the transitions $t_j$ that belongs to the zero solutions of $S$, less transition $t_{14}$ that corresponds to the transition with the true interpretation because of the received external event. In doing so, the following ordered sequence of pseudo firings will be produced (remembering that for a pseudo firing of a possibilistic Petri net, the actions associated with the fired transitions are not executed):

- for the marking $M$ corresponding to an object $< c_1 >$ in $P_2$ (Fig. 10(a)), the transition $t_2$ is pseudo enabled and can be pseudo fired producing new objects in $P_3$ and $P_4$ without removing the object in $P_2$. The resulting uncertain marking $M_{int_1}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$ and $P_4$.
- for the marking $M_{int_1}$, the transitions $t_3$ and $t_4$ are pseudo enabled and one of them can be pseudo fired. If $t_3$ is the chosen transition, a new copy of the object $< c_1 >$ is produced in $P_5$. The resulting uncertain marking $M_{int_2}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$ and $P_5$.
- for the marking $M_{int_2}$, the transitions $t_4$ and $t_5$ are pseudo enabled and one of them can be pseudo fired. If $t_4$ is the chosen transition, a new copy of the object $< c_1 >$ is produced in $P_6$. The resulting uncertain marking $M_{int_3}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$ and $P_6$.

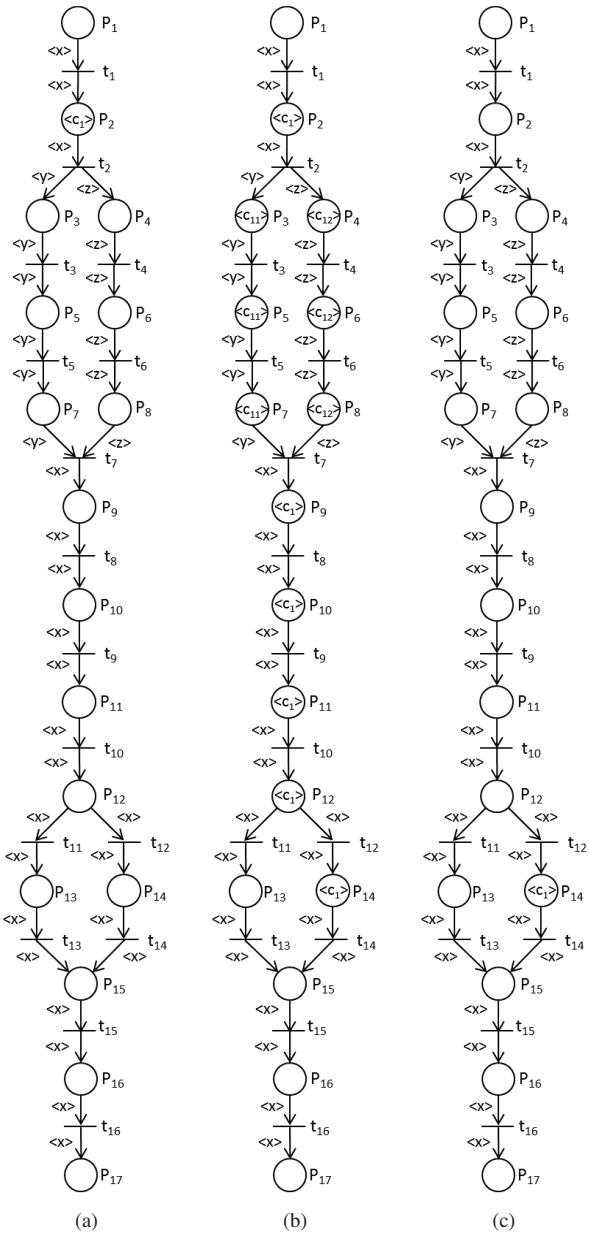


Figure 10. Simulation results: (a)Initial marking. (b)Pseudo-firing. (c)Final marking.

- for the marking $M_{int_3}$, the transitions $t_5$ and $t_6$ are pseudo enabled and one of them can be pseudo fired. If $t_5$ is the chosen transition, a new copy of the object $< c_1 >$ is produced in $P_7$. The resulting uncertain marking $M_{int_4}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$ and $P_7$.
- for the marking $M_{int_4}$, the transition $t_6$ is pseudo enabled and can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_8$. The resulting uncertain marking $M_{int_5}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$ and $P_8$.
- for the marking $M_{int_5}$, the transition $t_7$ is pseudo enabled and can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_9$. The resulting uncertain marking $M_{int_6}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$ and $P_9$.
- for the marking $M_{int_6}$, the transition $t_8$ is pseudo enabled and can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_{10}$. The resulting uncertain marking $M_{int_7}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$, $P_9$ and $P_{10}$.
- for the marking $M_{int_7}$, the transition $t_9$ is pseudo enabled and can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_{11}$. The resulting uncertain marking $M_{int_8}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$, $P_9$, $P_{10}$ and $P_{11}$.
- for the marking $M_{int_8}$, the transition $t_{10}$ is pseudo enabled and can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_{12}$. The resulting uncertain marking $M_{int_9}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$, $P_9$, $P_{10}$, $P_{11}$ and $P_{12}$.
- for the marking $M_{int_9}$, only the transition $t_{12}$ is pseudo enabled because the interpretation of transition $t_{11}$ is false. It can be pseudo fired producing a new copy of the object $< c_1 >$ in $P_{14}$. The resulting uncertain marking $M_{int_{10}}$ is given by copies of the object $< c_1 >$ in $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$, $P_9$, $P_{10}$, $P_{11}$, $P_{12}$ and $P_{14}$.

Finally, the final marking $M'$ corresponding to the object WorkFlow net of Fig. 10(b) is obtained. For this imprecise marking, the possibility of having a copy of the same object $< c_1 >$ in places $P_2$, $P_3$, $P_4$, $P_5$, $P_6$, $P_7$, $P_8$, $P_9$, $P_{10}$, $P_{11}$, $P_{12}$ and $P_{14}$ is 1. Due to the fact that the uncertain marking of the object $< c_1 >$ in $P_{14}$ enables the transition $t_{14}$(where the external event was detected), the recovery algorithm of possibilistic Petri nets presented in [19] allows for the marking of the WorkFlow net of Fig. 10(c) to became certain through the firing achievement of transition $t_2$, $t_3$, $t_4$, $t_5$, $t_6$, $t_7$, $t_8$, $t_9$, $t_{10}$ and $t_{12}$ (in fact, a new computation of the possibility distribution of the copies of object $< c_1 >$ is carried out in order to go back to certainty). Traditionally in possibilistic Petri nets, the recovery algorithm has the function of defuzzification of uncertain knowledge of the state of a Petri net when some certain events occur.

As a final result, the global marking of the WorkFlow net in Fig. 10(c) becomes consistent with the current situation of the process and the recorded pseudo firing sequences $t_2$, $t_3$, $t_4$, $t_5$, $t_6$, $t_7$, $t_8$, $t_9$, $t_{10}$ and $t_{12}$ informs the activities that were not correctly executed when considering the model of the process.

To take into account the kind of incident presented in this example with an ordinary Petri net based on the token player in Fig. 7, the process should be restarted manually or several new transitions should be created to consider all

possible abnormal scenarios. As a consequence, the global state of the process would go into a deadlock situation or the corresponding graph would rapidly become completely unreadable.

## VI. Conclusion

In this article, a possibilistic WorkFlow net model was presented with the purpose of dealing with non-conformance in Business Processes. Combining the routing structure of WorkFlow net with the uncertain reasoning of possibilistic Petri nets, it was possible to recover from certain kind of inconsistencies that can happen during the real time execution of the process when human actors are involved in the process activities. Such an approach was applied to a "Handle Complaint Process".

Comparing this approach with other works dealing with the problem of non-conformance, the main advantage is the fact that a formal process model allowing for the proving of some good properties, like the soundness property for example, was combined with a possibilistic approach which is very well adapted to the concept of flexibility and robustness in processes.

As a future work proposal, it will be interesting to present an inconsistency recovery approach that will be applied to a process not necessarily sound, knowing that in practice, the inherent flexibility of legacy systems does not always allow for the production of a process model that respects the soundness property.

## Acknowledgment

The authors would like to thank FAPEMIG, CAPES and CNPq for financial support.

## References

[1] A. Hofstede, W. van der Aalst, M. Adams, and N. Russell, *Modern Business Process Automation: YAWL and its Support Environment*. Springer Publishing Company, Incorporated, 2009.

[2] W. v. d. Aalst and K. v. Hee, *Workflow Management: Models, Methods, and Systems*, 1st ed., ser. MIT Press Books. The MIT Press, 2004, vol. 1.

[3] K. Mohammed, L. Redouane, and C. Bernard, "A deviation-tolerant approach to software process evolution," in *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, 2007, pp. 75 – 78.

[4] S. Thompson and T. Torabi, "An observational approach to practical process non-conformance detection," in *Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the*, 2009, pp. 62 –67.

[5] G. Cugola, E. Di Nitto, C. Ghezzi, and M. Mantione, "How to deal with deviations during process model enactment," in *Proceedings of the 17th international conference on Software engineering*, ser. ICSE '95. New York, NY, USA: ACM, 1995, pp. 265–273.

[6] G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, "A framework for formalizing inconsistencies and deviations in human-centered systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 3, pp. 191–230, jul 1996.

[7] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Information Systems*, vol. 33, pp. 64 – 95, 2008.

[8] W. M. P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H. M. W. Verbeek, "Choreography conformance checking: An approach based on bpel and petri nets," December 2005.

[9] J. Munoz-Gama, "Algorithms for process conformance and process refinement," Master's thesis, Universitat Politècnica de Catalunya (UPC), sep 2010.

[10] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst, "Towards robust conformance checking," in *Business Process Management Workshops'10*, 2010, pp. 122 – 133.

[11] S. Cîmpan and F. Oquendo, "Dealing with software process deviations using fuzzy logic based monitoring," *SIGAPP Appl. Comput. Rev.*, vol. 8, pp. 3 – 13, 2000.

[12] J. Cardoso, *Time Fuzzy Petri Nets*, ser. Studies in Fuzziness and Soft Computing. Physica-Verlag, 1999.

[13] J. Cardoso, R. Valette, and D. Dubois, "Possibilistic petri nets," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 5, pp. 573 –582, oct 1999.

[14] W. M. P. Van Der Aalst, "The application of petri nets to workflow management," *Journal of Circuits Systems and Computers*, vol. 8, pp. 21 – 66, 1998.

[15] L. Soares Passos and S. Julia, "Qualitative analysis of workflow nets using linear logic: Soundness verification," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 2009, pp. 2843 –2847.

[16] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, pp. 541 – 580, 1989.

[17] C. Sibertin-Blanc, "High level petri nets with data structure." in *Proceedings of the 6th european Workshop on Application and Theory of Petri Nets, Espoo, Finland*, Jensen, K., Ed., jun 1985, pp. 141–170.

[18] ——, "Cooperative objects: Principles, use and implementation." in *Concurrent Object-Oriented Programming and Petri Nets*, ser. Lecture Notes in Computer Science, G. Agha, F. de Cindio, and G. Rozenberg, Eds., vol. 2001. Springer, 2001, pp. 216–246.

[19] J. Cardoso, R. Valette, and D. Dubois, "Petri nets with uncertain markings." in *Applications and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed., vol. 483. Springer, 1989, pp. 64 – 78.