Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN APPROACH OF OPTIMISATION AND FORMAL VERIFICATION OF WORKFLOW PETRI NETS

[1]OUTMAN EL HICHAMI, [2]MOHAMMED AL ACHHAB, [3]ISMAIL BERRADA, [4]RACHID OUCHEIKH, [5]BADR EDDINE EL MOHAJIR

[1,5]University of Abdelmalek Essaâdi, Faculty of Sciences, Tetouan, Morocco

[2]University of Abdelmalek Essaâdi, National School of Applied Sciences, Tetouan, Morocco

[3,4]University of Sidi Mohamed Ben Abdellah, Faculty of Sciences and Technology, Fez, Morocco

E-mail: [1]el.hichami.outman@taalim.ma, [2]alachhab@ieee.ma, [3]iberrada@univ-lr.fr, [4]cheikh.rachid09@gmail.com, [5]b.elmohajir@ieee.ma,

ABSTRACT

In this paper, we are interested in the verification of Workflows (*WF*). *WF* are increasingly used for modeling and improving the quality of business processes (*BP*). In fact, the BP are becoming more complex because of the process nature which can be decomposed into sub-processes that run in parallel and/or sequentially. In this paper, we are particularly interested in verifying a class of properties known as response properties. Our second aim is to optimise the verification process of complex *WF*. Therefore, we first propose a method to compute and extract WF_φ which is a part of *WF* concerned with the property φ to verify, and second we apply an abstraction method which optimises the size of WF_φ. This optimisation is interesting so that the verification process can be done just on the optimise WF_φ, and we deduce the preservation of the satisfaction of φ in *WF*.

Keywords: *Workflow Petri nets, response properties, abstraction, formal methods*

1. INTRODUCTION

In order to achieve a quality approach, the process approach is recommended for companies. In fact, business processes (*BP*) are typically associated with operational objectives and business relationships such as an engineering development process, insurance claims process, and web services [21]. The representation of a *BP* in a form which supports automated manipulation like modeling or enactment by a workflow system.

Workflows (*WF*) [7] have been established to manage the automatic execution of *BP*. A *WF* process is a set of activities which contains executable tasks that consume and produce well defined data. In this study, we are interested in using the formal methods of *WF* process validation. We distinguish two main existing verification interests. On the one hand, the *WF* analysis axes that consist of verifying general properties like absence of deadlocks, livelocks or no dead activities [5][9]. On the other hand, the use of verification framework to validate the dynamic behaviors of *BP* [11][18][19][20]. The last approach needs a translation framework from the *WF* schema towards a format and content that are recognized by the verification tool. The specification of the properties should also be expressed within a logic that is understood and supported by the verification tool.

In this work, we focus on an automatic and direct verification of a class of properties related to a *WF* processes. We are particularly interested in responses properties. This type of properties can be written as follows:

- Task *A* will always be followed by task *B*;
- Task *A* and task *B* will be executed in parallel and after task *C*;
- Task *C* will be executed after tasks *A* and *B*.

This paper discusses the use of Workflow Petri nets (*WF-net*) introduced by van der Aalst [6]. *WF-net* is an established tool based on Petri nets for modeling and analyzing the correctness of *WF*.

In the specification phase, we aim that the designer can use a graphical user interface based on the same concepts as established in *WF-net* to specify the properties to be verified. An optimal verification approach is proposed.

Our aim is the integration of the formal verification techniques in the design phase. First, the designer starts with modeling the Workflow, then he specifies the properties to verify, in this way constraint specifications can be automatically and

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

intuitively validated by the designer at this early stage. The famous drawback of this approach is the verification processes complexity which is still a problem when we use formal methods. In order to optimise the verification process, we propose to use the abstraction notion defined in [13]. This approach allows to group the tasks/places that run in parallel and/or sequentially. In [13] the authors state that this abstraction process preserves the liveness and boundedness properties in *WF-net*.

Our verification process based on the abstraction and optimisation of *WF-net* is defined as follows: first we specify the property φ to be verified with the graphical user interface. The verification algorithm starts with an optimisation of *WF-net* model in order to keep just the part $WF\text{-}net_\varphi$ concerned with φ. Then, the abstraction process will be done only in this part ($WF\text{-}net_\varphi$). Our approach aims to verify the property φ on the optimised submodel of $WF\text{-}net_\varphi$, and conclude the satisfaction (or not) of φ in the global *WF-net*.

The organization of this paper is as follows: *Section 2* discusses the related work. *Section 3* and *4* provide formal definitions of basics *WF-net* that are required in the rest of this paper. *Section 5* describes a set of reduction for *WF-net*. We develop our approach of verification in *Section 6*. We also show the efficiency of our approach with experiments and analysis in *Section 7*. *Section 8* concludes the paper, and draws perspectives.

2. RELATED WORK

Several techniques and methods are proposed to validate *WF-net*. Verbeek et al. [1] have developed a tool, called *Woflan*[1], to verify soundness. However, the verification of large *WF-net* can be intractable due to the space explosion problem. The current initiative intends to remedy to this situation by proposing an abstraction and optimisation method of *WF-net*. YAMAGUCHI et al. [11] have also developed a tool which consists of *WfNetEditor* and *WfNetAnalyzer* that generates a *WF-net* as a PNML[2] file which is an intermediate representation of *WF-net*. The check process converts the PNML file into PROMELA and uses LTL formulas to specify the properties. Finally, the method uses the SPIN model checker. This approach drastically reduces the complexity of verification processes. However, its use does not deal with the dynamic behavior of *WF-net*, and its limited only by the acyclic *WF-net*.

Some works have contributed to the verification of dynamic behavior of *BP* [15] [12]. This verification process is always based on existing verification tool, which implies the transformation of *WF-net* in the language of verification supported by the verification tool and specification of properties to be verified by temporal logic.

In [16] the authors have implemented a concept proof of their approach with existing software namely the open modeling platform *Oryx* [14] and the *BPMN-Q* query language [10]. This approach is based on the decomposition of *BPMN-Q*. However, these approaches fail because they do not satisfy all properties of a query and the decomposition of a *BPMN-Q* query is complicated.

The above works show that the verification phase comes after the design phase, and the knowledge of the logic used for the specification of properties is needed.

Our approach has two advantages: (1) integrate the verification process in the design stage allowing a gradual validation of *BP*. This phase can be avoided by implementing a graphical specification interface. (2) exploit the abstraction and the optimisation of *WF-net* to remedy the problem of the complexity of the verification process.

3. BASICS OF PETRI NETS

Petri nets [2] are largely used as tool for representation, validation and verification of Workflows [5][6][9]. In this section, we give the basic definitions, notations of Petri nets and main results of their structure theory used in this work.

3.1 Petri nets

A Petri net is a tuple $N = (P,T,F)$ where:

1. $P \neq \emptyset$ is a finite set of places;
2. $T \neq \emptyset$ is a finite set of transitions with $P \cap T \neq \emptyset$;
3. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation.

A place can contain zero or more tokens. A token is represented by a black dot. The global state of a Petri net, also called a marking, is the distribution of tokens over places. Formally, a marking of a Petri net N is a function $M : P \rightarrow IN$. The initial marking of N is denoted by M_0.

3.2 Place and Transition

Let $N = (P,T,F)$ be a Petri net, we define a function $f : F \rightarrow \{0,1\}$ such that :

$$f(x,y) = \begin{cases} 1 \; if (x,y) \in F \\ 0 \; otherwise \end{cases}$$

[1] www.win.tue.nl/woflan/doku.php
[2] http://www.pnml.org

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

For each transition or place x we call a set $\bullet x = \{y \in P \cup T \mid f(y,x) = 1\}$ the *preset* of x, and a set $x\bullet = \{y \in P \cup T \mid f(x,y) = 1\}$ the *postset* of x.

We now introduce some additional notations used in the present paper.

A transition $t \in T$ is *enabled* at a marking M if $\forall p \in \bullet t : M(p) \geq 1$. If a transition t is *enabled* in marking M, it can be fired leading to a new marking M' written as $M \xrightarrow{t} M'$ such that:

$$M'(p) = \begin{cases} M'(p) = M(p) + 1 \text{ } if \text{ } p \in t\bullet \\ M'(p) = M(p) - 1 \text{ } if \text{ } p \in \bullet t \\ M'(p) = M(p) \text{ } otherwise \end{cases}$$

Given $w = t_1 \ldots t_n \in T^n$, we write $M_0 \xrightarrow{w} M$ if $M = M_0$ or there exist markings $M_1, \ldots, M_n$ such that $M_n = M$ and $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \ldots M_{n-1} \xrightarrow{t_n} M_n$. Then we say that M is *reachable*. The set of reachable markings of the Petri net N is denoted by $[M\rangle$ and defined by $[M\rangle = \{M : \exists w \in T \mid M_0 \xrightarrow{w} M\}$.

A Petri net N is *live iff* for every *reachable* marking M_i and every transition t, there is a marking M_j *reachable* from M_i which *enables* t.

A Petri net N is *bounded iff* for each place p there is a natural number n such that for every *reachable* marking, the number of tokens in p is less than n.

3.3 Routing constructs

The Petri net process definition defines three routing constructs used to specify the relationships between tasks during the process execution: sequential, parallel, and conditional [6], (see Figure 1).

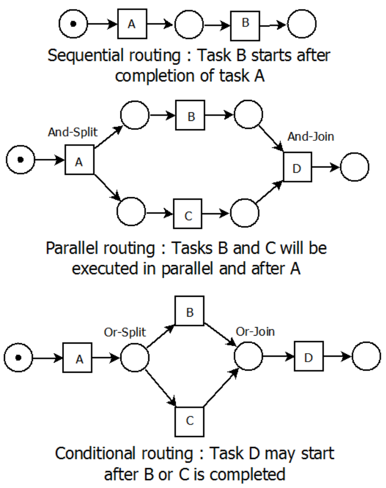

Figure 1: Routing constructs

4. WORKFLOW PETRI NETS (WF-NET)

WF-net is a restriction of Petri net for modeling, verifying and performance evaluation of workflows.

4.1 Definition

A Petri net is called a *WF-net* [6] if it has one input place i and one output place o without input and output transitions. For every transition or place $x \in P \cup T$, there exists a path from i to x and a path from x to o.

Formally, a Petri net $N = (P,T,F)$ is a *WF-net* if: $\exists i, o \in P$ such that $\bullet i = o\bullet = \emptyset, M_0(i) = 1$ and $\forall p \in P - \{i\} \mid M_0(p) = 0$ and $\forall x \in P \cup T, (i,x) \in F$ and $(x,o) \in F$. The resulting Petri net is *strongly connected*.

Normally, a *WF-net* should have a soundness property which guarantees a logical correctness of the modeled workflow.

A *WF-net* is intuitively said to be *sound iff*, for any case, the initial place is transformed to the final place and there are no dead transitions [17].

4.2 Definition

For further clarification, we give a simple example of the handling of a questionnaire [17]:

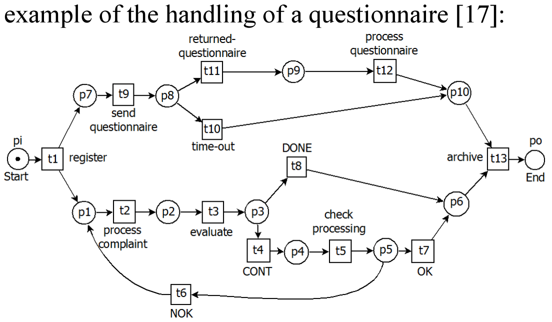

Figure 2: Example of WF-net: handling of a questionnaire

In this example, the *WF-net* starts with a registration step after which two parallel branches are started. The top branch is concerned with the handling of a questionnaire. After sending the questionnaire to the customer who submitted the complaints, there are two possibilities. The customer may return the questionnaire on time and subsequently it is processed. Otherwise, a timeout occurs and this step is skipped. In the lower branch of the *WF-net*, the complaint is first processed. After this, the result is evaluated. Based on this evaluation, the complaint is either checked or not. If it is checked, the result may be OK or not. If it is not OK, the complaint is processed again. This is repeated until no check is needed or the check is

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OK. Finally, after completing both parallel branches, the complaint is archived.

5. ABSTRACTION OF WF-NET

In this section, we present the rules which we use for the abstraction of some routing constructs of *WF-net* [13]. Our goal is to apply these reduction rules in order to group some transitions and places with preserving the essential properties of *WF-net*. As a result, the complexity of the verification process is reduced and can be performed more efficiently.

In the following, we denote $AbsN = (P_a, T_a, F_a)$ the abstract *WF-net* of $N = (P, T, F)$, and we use the following four operations of abstraction:

5.1 Abstraction of Series Transitions

The rule allows for the merging of two sequential transitions t and u with one place p in between these two transitions into only one transition v (see Figure 3).

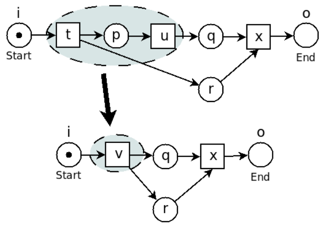

Figure 3: Abstraction of Series Transitions

Formally, let $p \in P$ be a place and $t, u \in T$ be two transitions, and $v \in T_a \setminus T$ be a transition such that:

Conditions on N:

1. $\bullet p = \{t\}$ (t is the only input of p);
2. $p\bullet = \{u\}$ (u is the only output of p);
3. $\bullet u = \{p\}$ (p is the only input of u);
4. $t\bullet \cap u\bullet = \varnothing$ (any output of t is not an output of u and vice versa).

Construction of $AbsN$:

1. $P_a = P \setminus \{p\}$;
2. $T_a = (T \setminus \{t, u\}) \cup \{v\}$;
3. $F_a = (F \cap ((P_a \times T_a) \cup (T_a \times P_a))) \cup (\bullet t \times \{v\}) \cup (\{v\} \times ((t\bullet \cup u\bullet) \setminus \{p\}))$.

5.2 Abstraction of Series Places

The rule allows for the merging of two sequential places p and q with one transition t in between them into a single place r (see Figure 4).

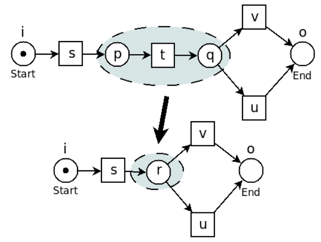

Figure 4: Abstraction of Series Places

Formally, let $t \in T$ be a transition and $p, q \in P$ be two places, and $r \in P_a \setminus P$ be a place such that:

Conditions on N:

1. $\bullet t = \{p\}$ (p is the only input of t);
2. $t\bullet = \{q\}$ (q is the only output of t);
3. $p\bullet = \{t\}$ (t is the only output of p);
4. $\bullet p \cap \bullet q = \varnothing$ (any input of p is not an input of q and vice versa).

Construction of $AbsN$:

1. $P_a = (P \setminus \{p, q\}) \cup \{r\}$;
2. $T_a = (T \setminus \{t\})$;
3. $F_a = (F \cap ((P_a \times T_a) \cup (T_a \times P_a))) \cup (((\bullet p \cup \bullet q) \setminus \{t\}) \times \{r\}) \cup (\{r\} \times q\bullet)$.

5.3 Abstraction of Parallel Transitions

The rule allows for the merging of multiple transitions (at least two) that have the same inputs and outputs into a single transition (see Figure 5).

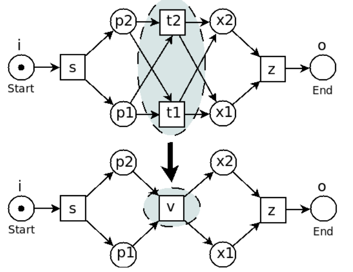

Figure 5: Abstraction of Parallel Transitions

Formally, let $\Gamma \subseteq T$ be the transitions where $|\Gamma| \geq 2$, and $v \in T_a \setminus T$ be a transition such that:

Conditions on N:

1. $\forall t_i, t_j \in \Gamma : \bullet t_i \equiv \bullet t_j$ (input places for all transitions in Γ are identical);
2. $\forall t_i, t_j \in \Gamma : t_i\bullet \equiv t_j\bullet$ (output places for all transitions in Γ are identical).

Construction of $AbsN$:

1. $P_a = P$;
2. $T_a = (T \setminus \Gamma) \cup \{v\}$;
3. $F_a = (F \cap ((P_a \times T_a) \cup (T_a \times P_a))) \cup (\{v\} \times \bullet t) \cup (t\bullet \times \{v\})$ where $t \in \Gamma$.

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.4 Abstraction of Parallel Places

The rule allows for the merging of multiple places (at least two) with the same inputs and outputs into a single place q (see Figure 6).

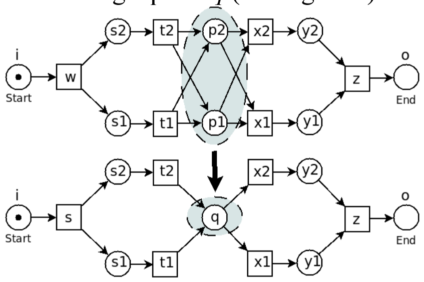

Figure 6: Abstraction of Parallel Places

Formally, let $Q \subseteq P$ be the places where $|Q| \geq 2$, and $q \in P_a \setminus P$ be a place such that:

Conditions on N:

1. $\forall p_i, p_j \in Q : \bullet p_i \equiv \bullet p_j$ (input transitions for all places in Q are identical);
2. $\forall p_i, p_j \in Q : p_i \bullet \equiv p_j \bullet$ (output transitions for all places in Q are identical).

Construction of *AbsN*:

1. $P_a = (P \setminus Q) \cup \{q\}$;
2. $T_a = T$;
3. $F_a = (F \cap ((P_a \times T_a) \cup (T_a \times P_a))) \cup (\bullet\, p \times \{q\}) \cup (\{q\} \times p \bullet)$ where $p \in Q$.

It is not difficult to see that the previous abstractions preserve the properties of liveness and boundedness [13]. That is, let $N = (P, T, F)$ and $AbsN = (P_a, T_a, F_a)$ be two *WF-net* before and after the previous reductions. Then *AbsN* is live (respectively bounded) *iff* N is live (respectively bounded).

6. VERIFICATION PROCESS

In this section, we present our verification process of *WF-net*. After the modelisation of *BP* to *WF-net* which will be verified, the designer can use a graphical interface to specify the property φ to validate. First we apply our algorithm (see Algorithm 1) to keep just the part of $WF\text{-}net_\varphi$ concerned with this property. Second, we apply the abstraction rules in order to optimise $WF\text{-}net_\varphi$. After that, the existing verification algorithms can be applied. Finally, we conclude that the satisfaction (or not) of φ in the original *WF-net*. This approach is presented in Figure 6:

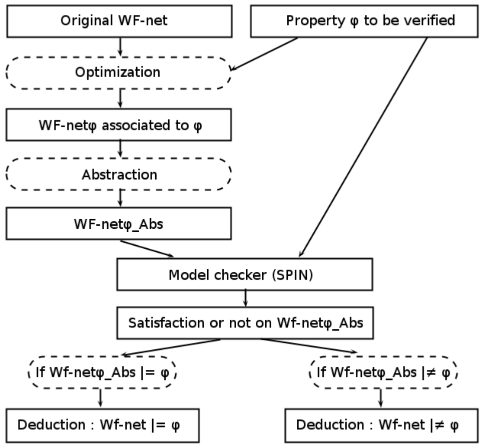

Figure 6: Process of WF-net verification

6.1 Response properties models

In this section, we first give three models of response properties to specify the dynamic behavior of the *BP*. Then we will also discuss some possible semantics of these models.

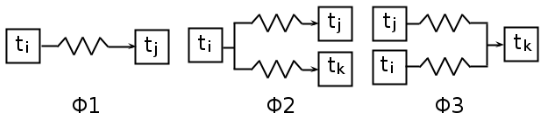

Figure 7: Examples of response properties

The first property (*Φ1*) states that task t_i will always be followed by task t_j. The second property (*Φ2*) states that task t_j and task t_k will be executed in parallel and after task t_i. The third property (*Φ3*) states that task t_k will be done after the end of task t_i and task t_j.

The semantics[3] of *Φ1*, *Φ2*, and *Φ3* can be interpreted in different ways. In this paper we focus on the implementation and the verification of LTL formulas [3]. This type of properties can be written as follows:

$$-(\Phi 1)\begin{cases} t_j \text{ is reachable from } t_i : \\ [\,](t_i \Rightarrow \Diamond t_j), (\text{LTL formula}) \end{cases}$$

$$-(\Phi 2)\begin{cases} t_j \text{ and } t_k \text{ will be reachable from } t_i : \\ [\,](t_i \Rightarrow [\,]\Diamond(\Diamond t_j \wedge \Diamond t_k)), (\text{LTL formula}). \end{cases}$$

$$-(\Phi 3)\begin{cases} t_k \text{ will be reachable from } t_i \text{ and } t_j : \\ [\,]((t_i \wedge t_j) \Rightarrow [\,]\Diamond t_k), (\text{LTL formula}). \end{cases}$$

[3] In temporal logic, ◊: Eventually in the future, []: Now and forever in the future, $\Diamond_n$: Eventually in the future after n steps, and $\Diamond_{m \geq n}$: Eventually in the future after m steps (m≥n).

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6.2 Optimisation and abstraction of WF-net

In this section, we present the process proposed to integrate the optimisation and abstraction in the formal verification of *WF-net*.

After the modelisation of the *WF-net*, the designer can specify the property Φ to be verified. The verification process starts with the optimisation of *WF-net* in order to keep just the part *WF-net*$_\Phi$ concerned with Φ. Then, the abstraction process will be done only in this part (*WF-net*$_\Phi$). Figure 8 schematizes this approach.

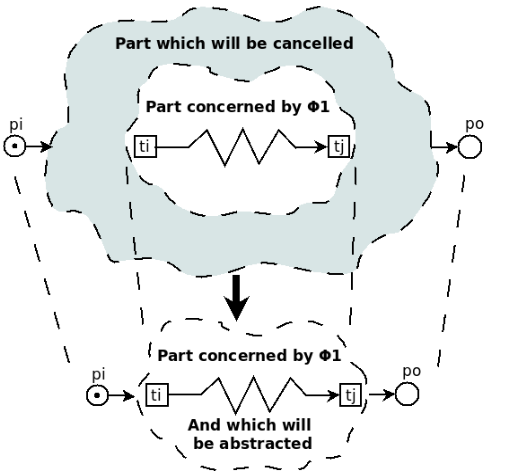

Figure 8: Our approach of optimisation and abstraction

6.2.1 Construction algorithm of the part concerned with Φ1

Algorithm1

```
Data:Original WF-net: N = (P, T, F), ti, tj : the
extremities tasks in Φ1
Result: Part of WF-net: NΦ1 = (PΦ1, TΦ1, FΦ1) related to Φ1

for ti ∈ T do
    Mark(ti) := false /*Mark ti as not visited*/
endfor
Function CreatePart(WF-net N, Task t)
begin
    Mark(t) /* Mark the t as explored */
    /* while: select only the p places
    adjacents to t */
    while p such as f(t, p) = 1 do
        /* for: select only the
        transitions adjacent to p */
        for t' (postset p●) do
            if NotMark(t') then
                Mark(t')
                if t'≡ tj then
                    Add to PΦ1 all the places between (ti, tj)
                    Add to TΦ1 all the transitions between (ti, tj)
                    Add to FΦ1 all the relations between (ti, tj)
                endif
                CreatePart(N, t')
            endif
        endfor
    endwhile
end
```

After the construction of *WF-net* $N_{\Phi 1}$, and in order to keep the same behavior of *WF-net N*, it is necessary to add some transitions and/or places to *WF-net* $N_{\Phi 1}$. There are four cases to study: places (respectively transitions) which have the links with outside transitions (respectively places) and not in *WF-net* $N_{\Phi 1}$ (see bold arcs in Figure 9):

1. Or-split (arc 1 in Figure 9) which connects a place p in $P_{\Phi 1}$ to a transition t in T and not in $T_{\Phi 1}$. In this case, we must add t in $T_{\Phi 1}$ in order to keep the alternative tasks in *WF-net* $N_{\Phi 1}$.
2. Or-join (arc 2 in Figure 9) which connects a transition t in T and not in $T_{\Phi 1}$ to a place p in $P_{\Phi 1}$. In this case it is not necessary to add t in $T_{\Phi 1}$ because in our case, we are only interested in paths that start from t_i and which have t_j as target termination.
3. And-split (arc 3 in Figure 9) which connects a transition t in $T_{\Phi 1}$ to a place p in P and not in $P_{\Phi 1}$. In this case, it is not necessary to add p in $P_{\Phi 1}$ because (1) in our case, we are only interested in paths w between t_i and t_j and (2) we are not concerned with parallel routing which doesn't influence the w.
4. And-join (arc 4 in Figure 9) which connects a place p in P and not in $P_{\Phi 1}$ to a transition t in $T_{\Phi 1}$. In this case, it is not necessary to add p in $P_{\Phi 1}$. In fact, the transition will be executed because in our work we consider that *WF-net N* is sound.

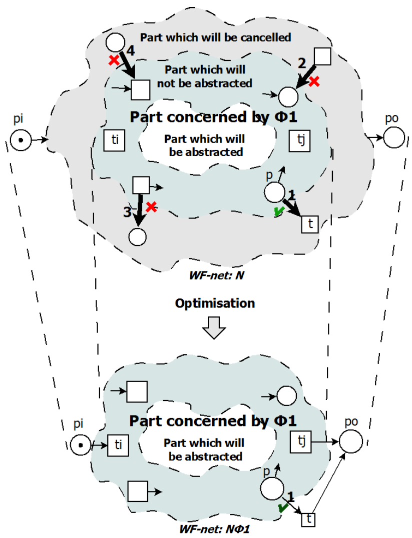

Figure 9: Construction of the WF-net $N_{\Phi 1}$

In the following, we formally show how to add these transitions to $N_{\Phi 1}$: Let *WF-net* $N = (P, T, F)$, $\Phi 1$: $t_i \rightsquigarrow t_j$ be the property to be verified, and $N_{\Phi 1} = (P_{\Phi 1}, T_{\Phi 1}, F_{\Phi 1})$ be the *WF-net* related to *WF-*

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

net N after the optimisation related to *Φ1*.

Formal definition of construction of *WF-net* $N_{\Phi 1}$:

1. Add the output transitions for all the Or-split places:

$$\underbrace{\forall p \in P_{\phi 1} \mid p\bullet = \{t'_1, \ldots, t'_m\} \in (T \setminus T_{\phi 1})}_{\Downarrow}$$

$$\begin{cases} T_{\phi 1} = T_{\phi 1} \cup t'_{i(1 \le i \le m)}; \\ F_{\phi 1} = F_{\phi 1} \cup (\{p\} \times t'_{i(1 \le i \le m)}). \end{cases}$$

2. Add the input place p_i and the output place p_o:

$$\begin{cases} P_{\phi 1} = P_{\phi 1} \cup \{p_i, p_o\}; \\ F_{\phi 1} = F_{\phi 1} \cup (p_i \times t_i) \cup (t_j \times p_o) \cup (t'_{i(1 \le i \le m)} \times p_o) \end{cases}$$

Remark: The construction of the WF-net concerned with both Φ2 and Φ3

To build the part concerned with the property *Φ2*, we apply the Algorithm 1 twice by changing the target extremities of *Φ2*. The same principle to create the part related to *Φ3*, but this time by modifying the source extremities of *Φ3*.

6.2.2 Example of Φ1

Let *φ1* be a property of type *Φ1* such that:

φ1: After "register", "OK" must be reachable.

We now present how to create $N_{\varphi 1} = (P_{\varphi 1}, T_{\varphi 1}, F_{\varphi 1})$ related to *WF-net N = (P, T, F) of* Figure 2 (the handling of a questionnaire). For this, we apply the Algorithm 1 to build the part concerned with *φ1*. After that, we add the transition t_8 to the Or-split place p_3. Finally, we add the input place p_i and the output place p_o. Therefore, the *WF-net* $N_{\varphi 1}$ is as follows:

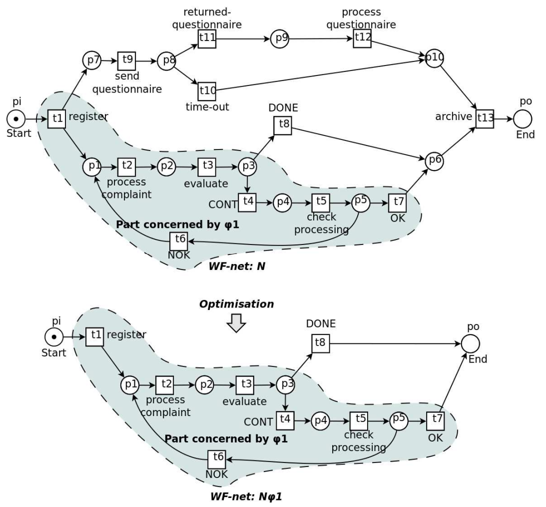

Figure 10: The WF-net $N_{\varphi 1}$ related to *φ1* after the optimisation of WF-net N

After the construction of *WF-net* $N_{\varphi 1}$, and in order to optimise more the size of *WF-net* $N_{\varphi 1}$, we perform the abstraction process in the part concerned with *φ1*. The result is the *WF-net* $AbsN_{\varphi 1}$ (presented in Figure 11).

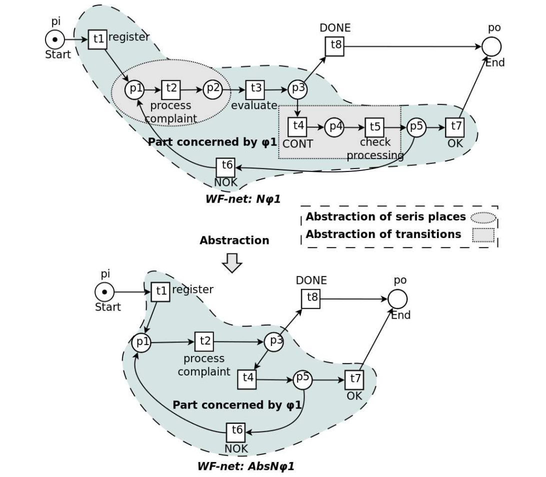

Figure 11: The WF-net $AbsN_{\varphi 1}$ *related to Figure 10 after the abstraction*

6.2.3 Example of Φ2 and Φ3

Let *φ2* (respectively *φ3*) be a property of type *Φ2* (respectively *Φ3*)

- *φ2: After "register", "returned questionnaire" and "evaluate" must be executed in parallel.*
- *φ3: "archive" is done after the end of "time-out" and "returned questionnaire".*

Figure 12 shows the parts concerned with *φ2* and *φ3*:

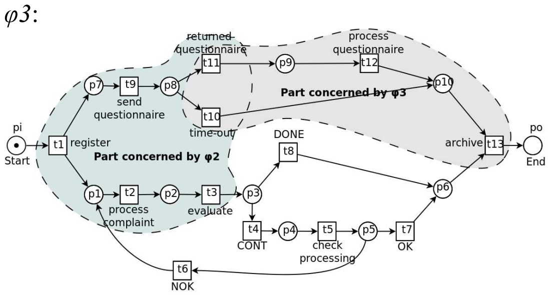

Figure 12: Parts concerned with φ2 and φ3

After the construction of *WF-net Nφ2* and *WF-net Nφ3*, the abstraction process can be done on the parts concerned with *φ2* and *φ3*. The result is the *WF-net AbsNφ2* and the *WF-net* Abs*Nφ3* (see Figure 13).

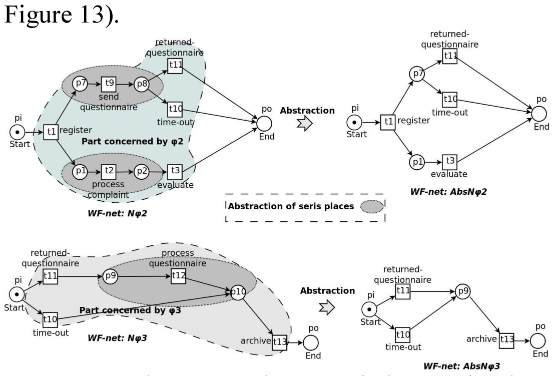

Figure 13: The WF-net AbsNφ2 and AbsNφ3 after the abstraction parts concerned with φ2 and φ3

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6.3 Proof of satisfaction of *Φ1*, *Φ2* and *Φ3* in The WF-net

In this section, we prove that if each of the three response properties models are satisfied in the *WF-net AbsN* after the optimisation and the abstraction of the original *WF-net N* related to these properties, then we conclude that the satisfaction in the original one.

In our context we consider that *WF-net N* is *sound*. So, every transition $t \in T$ will be enabled (because no deadlock transitions in *WF-net N* and every transition is reachable from M_0 the initial marking of *WF-net N*) which means that the source extremities of the transitions t_i, t_j and t_k in *Φ1*, *Φ2* and *Φ3* will be enabled.

6.3.1 Proof of Φ1

Let $N = (P, T, F)$ be a *WF-net*, *Φ1*: $t_i \rightsquigarrow t_j$ be a property to verify, and $AbsN_{\Phi 1} = (P_{\Phi 1a}, T_{\Phi 1a}, F_{\Phi 1a})$ be the *WF-net* after the optimisation and the abstraction of *WF-net N* related to *Φ1*.

We prove that if the property *Φ1* is satisfied in *WF-net* $AbsN_{\Phi 1}$, then we conclude that *Φ1* is satisfied in *WF-net N*.

Lemma1. *WF-net* $AbsN_{\Phi 1} \models \Phi 1 \Rightarrow$ *WF-net* $N \models \Phi 1$

Proof. According to the semantic of *Φ1* in LTL context, *WF-net* $AbsN_{\Phi 1} \models \Phi 1 \Rightarrow \forall$ paths $w = t_i \ldots t_j \in T_{\Phi 1a}$. *i.e.* $M(t_i \bullet) \xrightarrow{w} M(\bullet t_j)$. Because the optimisation and the abstraction keep all the possible paths between $t_i \ldots t_j$ in *WF-net* $AbsN_{\Phi 1}$. Furthermore, *WF-net N* is *sound* then no deadlock transitions $\Rightarrow \forall\, w' = t_i \ldots t_j \in T$ and $w \subseteq w'$. Hence, *WF-net* $N \models \Phi 1$.

6.3.2 Proof of Φ2

Let $N = (P, T, F)$ be a *WF-net*, *Φ2*: $t_i \rightsquigarrow <^{t_j}_{t_k}$ be a property to verify, and $AbsN_{\Phi 2} = (P_{\Phi 2a}, T_{\Phi 2a}, F_{\Phi 2a})$ be the *WF-net* after the optimisation and the abstraction of *WF-net N* related to *Φ2*.

We prove that if the property *Φ2* is satisfied in *WF-net* $AbsN_{\Phi 2}$, then we conclude that *Φ2* is satisfied in *WF-net N*.

Lemma2. *WF-net* $AbsN_{\Phi 2} \models \Phi 2 \Rightarrow$ *WF-net* $N \models \Phi 2$

Proof. According to the semantic of *Φ2* in LTL context, $AbsN_{\Phi 2} \models \Phi 2 \Rightarrow \forall$ paths $w_1 = t_i \ldots t_j$ and $\forall$ paths $w_2 = t_i \ldots t_K \mid (w_1, w_2) \in T^2_{\Phi 2a}$. *i.e.* $M(t_i \bullet) \xrightarrow{w_1} M(\bullet t_j)$ and $M(t_i \bullet) \xrightarrow{w_2} M(\bullet t_k)$.

Because the optimisation and the abstraction keep all the possible paths between $t_i \ldots t_j$ and between $t_i \ldots t_k$ in *WF-net* $AbsN_{\Phi 2}$. Furthermore, *WF-net N* is *sound* then no deadlock transitions $\Rightarrow \forall\, w' = t_i \ldots t_j$ and $\forall\, w'' = t_i \ldots t_K \mid (w', w'') \in T^2$ and $w_1 \subseteq w'$, $w_2 \subseteq w''$. Hence, *WF-net* $N \models \Phi 2$.

6.3.3 Proof of Φ3

Let $N = (P, T, F)$ be a *WF-net*, *Φ3*: $^{t_i}_{t_j} > \rightsquigarrow t_k$ be a property to verify, and $AbsN_{\Phi 3} = (P_{\Phi 3a}, T_{\Phi 3a}, F_{\Phi 3a})$ be the *WF-net* after the optimisation and the abstraction of *N* related to *Φ3*.

We prove that if the property *Φ3* is satisfied in $AbsN_{\Phi 3}$, then we conclude that *Φ3* is satisfied in *N*.

Lemma3. *WF-net* $AbsN_{\Phi 3} \models \Phi 3 \Rightarrow$ *WF-net* $N \models \Phi 3$

Proof. According to the semantic of *Φ3* in LTL context, *WF-net* $AbsN_{\Phi 3} \models \Phi 3 \Rightarrow \forall$ paths $w_1 = t_i \ldots t_k$ and $\forall$ paths $w_2 = t_j \ldots t_K \mid (w_1, w_2) \in T^2_{\Phi 3a}$. *i.e.* $M(t_i \bullet) \xrightarrow{w_1} M(\bullet t_k)$ and $M(t_j \bullet) \xrightarrow{w_2} M(\bullet t_k)$.

Because the optimisation and the abstraction keep all the possible paths between $t_i \ldots t_k$ and between $t_j \ldots t_k$ in *WF-net* $AbsN_{\Phi 3}$. Furthermore, *WF-net N* is *sound* then no deadlock transitions $\Rightarrow \forall\, w' = t_i \ldots t_k$ and $\forall\, w'' = t_j \ldots t_K \mid (w', w'') \in T^2$ and $w_1 \subseteq w'$, $w_2 \subseteq w''$. Hence, *WF-net* $N \models \Phi 3$.

7. EXPERIMENTS AND ANALYSIS

In this paper, we use the SPIN[4] tools to validate our proposal and to assure the theoretical results. In these experiments, we discuss the verification of the three models of response properties *Φ1*, *Φ2* and *Φ3*. The results of this analysis show the performance of our approach.

7.1 Transforming WF-net to PROMELA

In [8], the authors have proposed a method to describe a WF-net into PROMELA[5] that can be simulated and verified with the SPIN model checker. In this method, a *WF-net* system is represented as a single process. The process describes each firing of its transitions.

Program1 presents an outline of the PROMELA program for *WF-net* (handling of a questionnaire).

7.2 LTL formulas

The properties to be verified in SPIN have to be expressed as LTL formulas. LTL formulas correspond to the response properties *φ1*, *φ2* and *φ3* to be verified and can be rewritten as follows:

- *φ1:* After "*register*", "*OK*" must be reachable.
  []((M[1]>=1) -> <>(M[6]>=1)).
- *φ2:* After "*register*", "*returned questionnaire*" and "*evaluate*" must be executed in parallel.
  []((M[0]>=1)-><>(M[2]>=1&& M[8]>=1)).
- *φ3:* "*archive*" is done after the end of "*time-out*" or "*returned questionnaire*".
  []((M[9]>=1| |M[10]>=1)-> <>(M[11]>=1)).

[4] http://spinroot.com
[5] spinroot.com/spin/Man/promela.html

Journal of Theoretical and Applied Information Technology
31st March 2014. Vol. 61 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Program1

```
#define Place 12
#define Transition 13
/* Variables representing a state of
WF-net */
int M[Place]; /* Marking */
int X[Transition]; /* Firing count */
/* A firing of Transition t */
/* remove specifies the change of the
marking of •t */
/* add specifies the change of the
marking of t• */
/* fire (x) increments the element
corresponding to t in X[t] */
#define remove1(x)   (x>0        )    -> x--
#define remove2(x,y) (x>0 && y>0)  -> x--; y--
#define add1(x)        x++
#define add2(x,y)      x++; y++
#define fire(x)        x++
/* Process representing WF-net */
init
{
M[0]=1; /* Set the initial marking */
do
:: atomic{remove1(M[0]) -> fire(X[0]); add2(M[1],M[7])}
:: atomic{remove1(M[1]) -> fire(X[1]); add1(M[2])}
:: atomic{remove1(M[2]) -> fire(X[2]); add1(M[3])}
:: atomic{remove1(M[3]) -> fire(X[3]); add1(M[4])}
:: atomic{remove1(M[3]) -> fire(X[7]); add1(M[6])}
:: atomic{remove1(M[4]) -> fire(X[4]); add1(M[5])}
:: atomic{remove1(M[5]) -> fire(X[5]); add1(M[1])}
:: atomic{remove1(M[5]) -> fire(X[6]); add1(M[6])}
:: atomic{remove1(M[7]) -> fire(X[8]); add1(M[8])}
:: atomic{remove1(M[8]) -> fire(X[9]); add1(M[10])}
:: atomic{remove1(M[8]) -> fire(X[10]); add1(M[9])}
:: atomic{remove1(M[9]) -> fire(X[11]); add1(M[10])}
:: atomic{remove2(M[6],M[10]) -> fire(X[12]);
add1(M[11])}
od
}
```

7.3 Experimental results

In this section we give some statistics in order to show the performance of our approach. We compare the size, the memory[6] and the verification time[6] between the original *WF-net* related to Figure 2 and the *WF-net*: $AbsN_{\varphi1}$, $AbsN_{\varphi2}$ and $AbsN_{\varphi3}$.

We first give the results without optimisation and abstraction of *WF-net* related to Figure 2.

Table 1: Experiments without optimisation and abstraction.

LTL formulas	States, stored	Transitions	Memory (Mb)	Times (s)
$\varphi1$	15	61	11.876	0.18
$\varphi2$	16	52	12.364	0.21
$\varphi3$	12	40	11.485	0.18

[6] Verification with disable dead-variable elimination

To evaluate the effect of our approach, we perform the verification of properties $\varphi1$, $\varphi2$ and $\varphi3$ in the $AbsN_{\varphi1}$, the $AbsN_{\varphi2}$ and the $AbsN_{\varphi3}$. These experiments are given in Table 2.

Table 2: Experiments with optimisation and abstraction.

LTL formulas	States, stored	Transitions	Memory (Mb)	Times (s)
$\varphi1$	8	12	3.770	0.02
$\varphi2$	10	23	3.672	0.03
$\varphi3$	5	8	3.522	0.02

For more clarification, Figure 14 illustrates this comparison between the original *WF-net* related to Figure 2 and the *WF-net*: $AbsN_{\varphi1}$, $AbsN_{\varphi2}$ and $AbsN_{\varphi3}$.

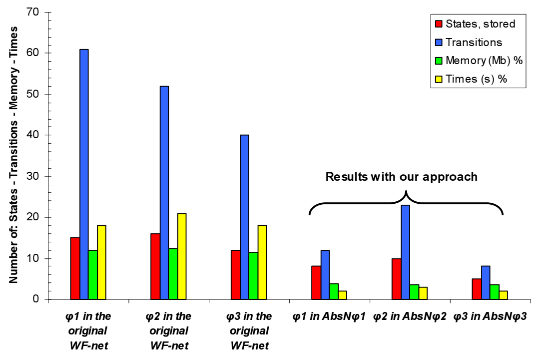

Figure 14: Statistics of the experimental results

8. CONCLUSION

This paper proposes a new approach to verify the *WF-net* in a *BP* context. We exploited the optimisation and the abstraction methods to remedy the verification processes complexity which is still a problem when we use a formal method. We are particularly interested in the response properties, and we present three class of these properties.

We propose an algorithm for extracting a part from *WF-net* in order to build an abstract *WF-net*. After this abstraction, we performed an optimisation in abstract *WF-net* to optimise more the size of *WF-net*. This method has the advantage of preserving the dynamic behaviors of *WF-net*.

Our approach is based on the verification of the response property on a part of the *WF-net* which is concerned by this property, and we proved that if the optimised *WF-net* satisfies this property, then we deduced the validation in the global *WF-net*. This proof is enriched by several practical experiences with the SPIN tools in order to show the performance of our approach.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

For now, we study the possibility to facilitate the conditions of abstraction in order to better optimise the size of abstract *WF-net*, and we will investigate the semantic expansions of the response properties.

REFRENCES:

[1] Henricus M.W. Verbeek. "Verification of WF-nets", PhD thesis, Eindhoven : Technische Universiteit Eindhoven. 2004.

[2] T. Murata, "Petri Nets: Properties, Analysis and Applications" an invited surve y paper, *Proceedings of the IEEE*, Vol.77, No.4 pp.541-580, 1989.

[3] Z.MANNA, A.PNUELI. "The temporal logic of reactive and concurrent systems", *Springer-Verlag*, New York, USA, 1992.

[4] Gerard J. Holzmann. "The Model Checker SPIN". *IEEE Trans*. Software Eng. 23(5), pp. 279-295, 1997.

[5] W.M.P. van der Aalst. "Verification of Workflow Nets", *ICATPN 97*, Volume 1248 of LNCS, pp. 407-426, 1997.

[6] W.M.P. van der Aalst. "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, Vol.8, No. 1, pp. 21-66, 1998.

[7] W.M.P. van der Aalst and K.M. van Hee. "Workflow Management: Models, Methods and Systems", *MIT Press*, Cambridge, p. 267, 2004.

[8] G.J. Holzmann, "The Spin Model Checker", Addison-Wesley, p. 596, 2003.

[9] Kamel B. and Rahma B. and Zohra S. "Workflow Soundness Verification based on Structure Theory of Petri Nets", *IJCIS Journal*, pp. 51-62, 2007.

[10] Awad, A.: BPMN-Q "A Language to Query Business Processes". In *EMISA*, pp. 115-128, 2007.

[11] Shingo Y., Munenori Y., and Minoru T.. "A Soundness Verification Tool Based on the SPIN Model Checker for Acyclic Workflow Nets", *ITC-CSCC*, pp. 285-288, 2008.

[12] Ahmed Awad, Gero Decker, Mathias Weske, "Efficient Compliance Checking Using BPMN-Q and Temporal Logic". (*BPM*) *Springer Verlag,* pp. 326-341, 2008.

[13] M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede, D. Edmond "Soundness-preserving reduction rules for reset workflow nets".*Information Sciences*, Vol. 179, No. 6, pp. 769-790, 2009.

[14] Decker, G., Overdick, H., Weske, M.: "Oryx - Sharing Conceptual Models on the Web", *International Conference on Conceptual Modeling (ER)*, LNCS 5231, pp. 536-537, Springer Verlag, 2008.

[15] M. Alam, M. Nauman, X. Zhang, T. Ali and P.C.K. Hung. "Behavioral Attestation for Business Processes", *Int. J. Web Service Res.*, Vol. 7, No. 3, pp. 52-72, 2010.

[16] Sherif Sakr, Ahmed Awad, Matthias Kunze. "Querying Process Models Repositories by Aggregated Graph Search", *Business Process Management Workshops*, Vol. 132, pp 573-585, Springer, 2012.

[17] W.M.P. van der Aalst, Kees M. van Hee, Arthur H. M. ter Hofstede, Natalia Sidorova, H. M. W. Verbeek, Marc Voorhoeve and Moe Thandar Wynn. "Soundness of workflow nets: classification, decidability, and analysis". *Formal Asp. Comput,* Vol. 23 No. 3, pp. 333-363, Springer-Verlag, 2011.

[18] Juan Carlos Polanco Aguilar, Koji Hasebe, Manuel Mazzara and Kazuhiko Kato, "Model Checking of BPMN Models for Reconfigurable Workflows", *Computing Science, Newcastle University,* Technical report series, No. CS-TR-1274, p. 6, 2011.

[19] Juan Carlos Polanco Aguilar, Koji Hasebe, Manuel Mazzara and Kazuhiko Kato. "Toward Design, Modelling and Analysis of Dynamic Workflow Reconfigurations - A Process Algebra Perspective". *WS-FM*, Vol. 7176, pp 64-78, Springer-Verlag, 2012.

[20] F Abouzaid, M Mazzara, J Mullins, N Qamar. "Towards a formal analysis of dynamic reconfiguration in WS-BPEL", *Intelligent Decision Technologies*, Vol. 7, No. 3, pp. 213-224, 2013.

[21] Mohammed AbuJarour, Ahmed Awad. "Web Services and Business Processes: A Round Trip", *Web Services Foundations*, pp.3-29, 2014.