Formal approach for compliance rules checking in Business Process Models

Oussama Mohammed Kherbouche, Adeel Ahmad, Henri Basson

Formal approach for compliance rules checking in Business Process Models

Oussama Mohammed Kherbouche, Adeel Ahmad, Henri Basson
Université Lille Nord de France
Laboratoire d'Informatique, Signal et Image de la Côte d'Opale
BP-719 62228 CALAIS Cedex FRANCE
E-mail: {kherbouche, ahmad, basson}@lisic.univ-littoral.fr

***Abstract*-the business process models should comply with a set of rules describing the operations, policies and constraints that an organization must respect under financial authorities. However, the large number of rules and their frequency of changes make the traditionally used manual compliance checking a time-consuming task. As a result an automated compliance checking should be adopted. This paper proposes a formal approach for automated compliance checking. It proposes to map BPMN models directly to finite state machines (i.e., Kripke structures) and to express the compliance rules in a graphical language for better understandability. Subsequently, these are translated into linear temporal logic formulae for their integration. The compliance of business process models can be verified by means of model checking technology. The main goal is to increase the efficiency of the deployment of business process models while minimizing the risks and cost of the compliance inspection.**

Keywords—BPMN process model; compliance rules; Kripke structure; LTL; ProMeLa; SPIN model checker

I. INTRODUCTION

The Business Process Models (BPM) [1, 2] has been widely used for the past decade. The increasing popularity of BPM is due to its notational simplicity [3] and its expressiveness. However, the business process and their operations should satisfy a set of policies or constraints characterized by compliance rules. The rapidly changing nature of rules requires checking business processes each time a rule is added or changed [4]. The change in compliance rules can occur in line with the business goals, and also with legal regulations. For example, certain execution orders between the activities, new policies or regulations such as the risk assessment in the banking sector, etc. The increasing frequency of changing rules requires business processes to adopt an automatic compliance checking.

Different approaches have been proposed for the verification of some properties in different models [5-13]. An automated approach supported by a ProM framework allows detecting violations from workflow event logs using Linear Temporal Logic (LTL) checkers is proposed in [5]. The authors in [6] propose a method to check correctness properties of workflows implemented in Business Process Execution Language (BPEL). It maps the BPEL to dataflow network and the dataflow network is mapped to a ProMeLa model. In [7], the authors discuss a formal approach based on model checking to verify the business process models (defined in BPEL and formalized with pi-calculus) against compliance rules expressed in the Business Property Specification Language (BPSL). Another approach [8] is able to express constraints in PLTL (Past Linear Temporal Logic) rather than only LTL, which gives the approach more expressiveness over the others by using a method from their previous work [9] of defining BPMN-Q (BPMN-Query), which is a modeling tool to concentrate on gates. The authors, in [10], use π-calculus to represent the workflow patterns.

The benefit of using model checking is to better visualize the counter examples, which are produced in the case that the formula being checked is found to be non-valid [11]. The verification in π-calculus is done through checking the bi-simulation equivalence, sometimes results are not obtained in reasonable amount of time, even to prove the simple correctness requirements [12].

In this article, we focus on process models designed with Business Process Modeling Notation (BPMN) which is accepted as a standard in business process modeling community. Indeed, we propose an automated approach to verify the compliance of BPMN process models with the set of established rules using model checking technique. The approach transforms the BPMN process model directly into Kripke structures [13] without involving the intermediate step which is generally Petri-nets in order to express their behavior. The compliance rules are expressed into temporal logic (using LTL) formulae with the help of a graphical notation to obtain the same level of abstraction as the business process models. The proposed approach can help to better verify the compliance rules after each change.

The rest of the article is organized as follows: the section II discusses the proposed approach, in detail. The section III presents the tool design and the implementation details. Later, the section IV further elaborates the validation of the approach with the help of an example. Finally, the section V concludes our contribution and briefly highlights the future prospects.

II. BPM ANALYSIS USING MODEL CHECKING

In this section, we formally explain the compliance checking in BPMN process models. The compliance rules,

as discussed earlier, are expressed as LTL formulae through a graphical notation. It allows the formalization on the same level of abstraction as the models. It also provides an intermediate interface between the compliance rules and the LTL formulae expressed in a particular tool like EpiSpin[1], as mentioned in the section IV, for non-technical users.

The approach, as illustrated in Fig.1 involves the BPMN transformation into Kripke structure and model checking. The steps involved in this procedure are detailed in the following:

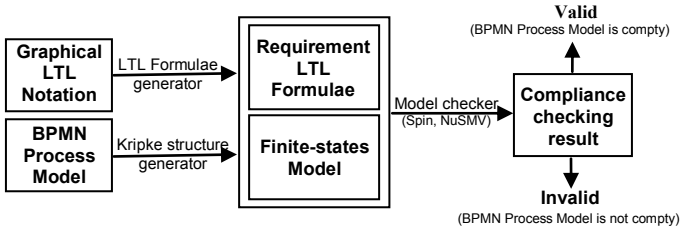


Fig. 1. Compliance checking approach

A. Mapping BPMN to Kripke structure

We primarily attempt to generate the finite states model from the given BPMN process models with the help of Kripke structures, such that the states of the Kripke structure express the behavior of the process model.

The objective of obtained Kripke structure is to facilitate the better verification of the desired temporal properties such as: $\mathcal{M} \models \phi$ iff $\mathcal{M}, \pi \models \phi$ (i.e. $\mathcal{M}$ satisfy ϕ for all paths π in a Kripke structure $\mathcal{M}$).

In this translation, the start event $\mathcal{E}^s$ of the BPMN process model represent the initial state ($I \subseteq S$) of Kripke structure. The set of flow objects in BPMN process which can be partitioned into Events $\mathcal{E}$, Activities $\mathcal{A}$ and Gateways $\mathcal{G}$, and the set of sequence flows $T \subseteq F \times F$ of this model represent the finite non-empty set of states (S) of a Kripke structure and the transition relations (R), respectively.

The mapping of BPMN to Kripke structure necessarily define the set of atomic proposition (AP) which later on, associates a labeling function with each generated state. A labeling function describes properties of a given state and can be represented as, $\mathcal{L}(s)$ holds in s, where $s \in S$

In this structure, each state $s \in S$ is labeled with *enabled* and *executed* transitions. Where, *E_A* signifies that the transition A is enabled and *ex_A* signifies that the transition A is executed (completed). A brief description of the mapping from a set of BPMN elements to Kripke structure is given in Tab. I. Once the Kripke structure is obtained, we then proceed to translate the compliance rules into LTL formulas [14] using a given set of graphical notations.

B. Compliance checking rules generation

The compliance rules are typically difficult to understand under a textual format for business process model, we propose a set of graphical notations called G-LTL, which

1 http://epispin.ewi.tudelft.nl/

TABLE I. MAPPING OF BPMN OBJECTS TO KRIPKE STRUCTURE

BPMN Object	Kripke Structure
Start s	Start
End e	Final
Message M	E_M, M, exM
Task T	E_T, T, exT
Looped Task T	E_T, T, exT
T, Ex, T_x	E_T, T, exT, E_T_x, T, exT_x
A, B, C	E_A, A, exA, E_B, B, exB exC, E_C, C
A, B, C	E_A, A, E_B, B, exA exB, E_C, C, exC
A, B, C	exA, E_B, B, exB, E_C, C, exC
A, B, C	E_A, A, exA, E_C, E_B, B, exB

represent the graphical version of the LTL formulae.

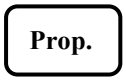
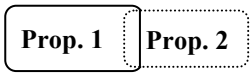
The objective of G-LTL definition is to facilitate the expressions of compliance rules and formalization on the same level of abstraction as the process models. The Tab. II summarizes the graphical notations of G-LTL.

The generator transforms automatically this graphical notation to corresponding LTL formula expressed in a particular tool like EpiSpin.

C. Automated compliance checking

The model checking [15] is a technique for formal

TABLE II. THE MAPPING OF G-LTL NOTATIONS TO LTL FORMULAE

G-LTL Notation	LTL Formula
	G Prop.
	X Prop.
	F Prop.
	Prop. 1 $\mathcal{U}$ Prop. 2
	¬ Prop.
	Prop. 1 ∧ Prop. 2
	Prop. 1 ∨ Prop. 2
	Prop. 1 XOR Prop. 2
	Prop. 1 → Prop. 2
	Prop. 1 ↔ Prop. 2
	(Prop. 1 ∧ Prop. 2)

modeling and analysis of reactive systems that exhibit random or probabilistic behavior. It verifies the temporal formula ϕ holds for that finite state machines $\mathcal{M}$ or not. As a result, it confirms the compliance of the process models. Otherwise, it returns a counterexample in cases of violation of compliance rules.

D. Automated compliance checking

The model checking produce counterexamples if the verification of temporal properties are not satisfied [7, 16]. Generally, these counterexamples contain internal state transitions rather than the process models. It makes them difficult to understand by a non-technical user. It is therefore, the output of the model checker should be translated in the visual notation to benefit from the generated counterexamples. The mapping of a counterexample to the source BPMN process model can significantly support the determination of the causal flow that leads to the violation of compliance rules. We use model checker dependency, which contain a tool chain that translates the output of the model checker back to the process model notations. This allows us to map each state to the original BPMN process model element and colored as red to highlight the errors to modelers and business experts.

III. TOOL DESIGN AND IMPLEMENTATION

In order to validate the presented approach, we implement a tool using a set of Eclipse IDE plug-ins, also called *"CC4BPMN"* (Compliance Checking for BPMN). This plug-in is composed of two modules, *"BPMN2PROMELA"* and *"CR2LTL"* to translation BPMN to ProMeLa (PROcess MEta LAnguage) and Compliance Rules to LTL, respectively. BPMN2PROMELA transform automatically the BPMN process model (expressed using Eclipse BPMN 2.0 Modeler plug-in) to ProMeLa model by providing input ProMeLa model file (*.pml) (i.e. EpiSpin translates the ProMeLa model into Kripke structure). CR2LTL proposes a set of G-LTL notation to express the compliance rules, expected to be checked by providing input LTL file (*.prp).

The ProMeLa model file and the compliance rules expressed by G-LTL notation file are presented as input to EpiSpin plug-in which verifies the compliance of the process models. Finally, when the compliance rules to be checked are not satisfied by the ProMeLa model, the counterexample returned by EpiSpin are mapped to the source BPMN process model.

Indeed, the BPMN process model can be transformed into ProMeLa language, which maps the processes, sub-processes and activities into ProMeLa processes and connector paths into ProMeLa channels. The messages between processes are represented, without loss of generality using integers in ProMeLa. In this translation, we use the following notation: c will represent a channel and m the message sent or received in this one. cS will denote an array of channels and mS an array of messages to be sent or received in each channel in the array cS.

The generation of temporal logic formulae expresses the rules to be checked in EpiSpin model checker. In [17], we show the translation of some principal elements: Sequence, *AND-Split*, *AND-Join*, *XOR-Split*, *XOR-Join*, *OR-Split*, *OR-Join* to ProMeLa Language. ProMeLa does not include syntax for LTL formulae [18, 19]. EpiSpin, however, can translate such formulae into ProMeLa syntax, with command line option -f "[] ((P &&! q) U r)").

The translation is a never claim, encoding the Büchi-automata acceptance condition [20, 21]. The formulae must then express negative properties of "errors" (negation of errors): execution sequences that satisfy the formula can be reported as correctness violations in verification.

IV. CASE STUDY

The case study consists of a simple expense reimbursement process shown in Fig. 2.

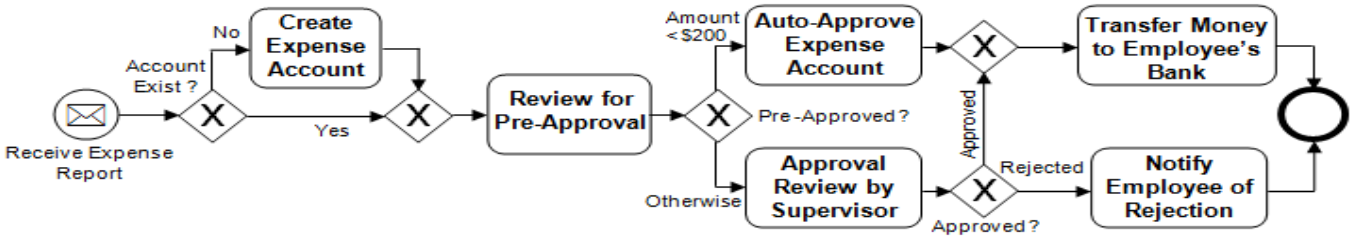


Fig. 2. Expense Reimbursement Process.

A. *Expense Reimbursement Process*

The employees of a company claim an expense reimbursement, for instance, buying the office supplies or software. After the expense report is received (Receive Expense Report), a new account is created if the employee does not already have one (Create Expense Account). The report is then reviewed for automatic approval (Review for Pre-Approval). If amount is less than $200 this one is automatically approved (Auto-Approve Expense Account). Otherwise if amount is equal or more than $200, then it requires an approval of the supervisor (Approval Review by Supervisor). In case of rejection, the employee receives a rejection notice through email (Notify Employee of Rejection), or otherwise, in case of acceptance, the reimbursement goes to the employee's direct deposit bank account (Transfer Money to Employees Bank).

The Expense Reimbursement process has to comply with several rules. We assume that the following rules among others should be verified for the process.

- *R1: if expense account does not exist then a new account will be created.*
- *R2: if the amount is lower than $200 then the expense account shall be auto-approved.*
- *R3: after request is approved, the money is transferred to the employee's bank; otherwise, a notification of rejection is sent to the employee.*

1) The PROMELA Model generation: in the following, we will generate the ProMeLa model corresponding to the expense reimbursement process shown in Fig. 2.

We define nine global channels denoted as cS to establish the communication between activities. We define also cS1, cS2 and cS3 as auxiliary array channels to simplify the exchanging of messages between activities.

Receive Expense Report - this event is initiated by receiving a request from an employee for expense reimbursement. 'exist' is 0 when the account doesn't exist or 1 when it exists. It is translated to ProMeLa process as shown in Listing 1.

LISTING 1. RECEIVE EXPENSE REPORT PROMELA PROCESS

```
proctype ReceiveExpenseReport() {
  int exist;
  chan cS1[2] = [1] of {int} ;
  cS1[0]= cS[0];/*Send to create expense Account*/
  cS1[1]= cS[1];/*Send to review pre-Approval*/
  R:
  if
   :: exist=0 /*Account doesn't exist*/
   :: exist=1 /* Account exist */
  fi;
  XORSplit(cS1,exist,1); goto R
}
```

Create Expense Account - This activity is chosen among one of the two possibilities in a nondeterministic manner without loss of generality, either the expense account is created or it does not exist. It is translated to ProMeLa process as shown in Listing 2.

LISTING 2. CREATE EXPENSE ACCOUNT PROMELA PROCESS

```
proctype createExpenseAccount() {
  int x;
  receive(cS[0],x);/*Receive from Expense Report*/
  send(cS[2],1);/*Send to Review for Preapproval*/
}
```

Review for Pre-Approval - This activity decides whether the expense account can be approved automatically or it requires a supervisor. As stated before, there are two possible decisions; once again, we choose non-deterministically one of them, without sacrificing the generality of verification. The translation of Review for Pre-Approval activity to ProMeLa process is shown in Listing 3.

LISTING 3. REVIEW FOR PRE-APPROVAL PROMELA PROCESS

```
proctype PreApproval() {
  mtype Amount;
  int x;
  chan cS2[2]=[1] of {int};
  chan cS3[2]=[1] of {int};
  cS2[0]= cS[1];
  cS2[1]= cS[2];
  cS3[0]= cS[3]; /*Send to Auto-approve Expense*/
  cS3[1]= cS[4]; /*Send to Approval review by
                                  supervisor*/
  P: XORJoin(cS2,x);/*Receive from Expense Report
                       or Create Expense Account*/
  if
   ::(Amount<200)-> x=0
   ::(Amount>=200)-> x=1
  fi;
  XORSplit(cS3,x,1); goto P
}
```

Auto-Approve Expense Account - This activity is chosen if the requested amount by an employee is less than $200. It is translated to ProMeLa process as shown in Listing 4.

LISTING 4. AUTO-APPROVE EXPENSE ACCOUNT PROMELA PROCESS

```
proctype AutoApp() {
  int x;
  A:
  receive(cS[3],x);/*Receive from review for pre-
                                        Approval*/
  send(cS[5],1);/*Send to Transfer Money to
                                  Employee's Bank*/
  goto A:
}
```

Approval Review by Supervisor - This activity is chosen if the requested amount by an employee is more than $200. In this case the approval is reviewed by a supervisor. 'approved' is 1 when the review is approved, otherwise 0. It is translated to ProMeLa process as shown in Listing 5.

LISTING 5. APPROVAL REVIEW BY SUPERVISOR PROMELA PROCESS

```
proctype AppBSupervisor() {
  int approved, x;
  chan cS3[2]=[1] of {int}
  cS3[0]= cS[6];/*Send to Notify Employee of
                                        Rejection*/
  cS3[1]= cS[7];/*Send to Transfer Money to
                                    Employees Bank*/
  B:
  receive(cS[4],x);/*Receive from Review for pre-
                                        Approval*/
  if
   ::(approved=1) /*Approved*/
   ::(approved=0) /*Rejected*/
  fi;
  XORSplit(cS3,approved,1); goto B
}
```

Transfer Money to Employee's Bank - The money is transferred to employee's bank account when the expense reimbursement request is approved. It is translated to ProMeLa process as shown in Listing 6.

LISTING 6. TRANSFER MONEY TO EMPLOYEE'S BANK PROMELA PROCESS

```
proctype TransferMoney() {
  int x;
  T:
  XORJoin(cS,x);
  send(cS[8],1);/*Send to end process*/
  goto T;
}
```

Notify Employee of Rejection - The rejection is notified to employee when the expense reimbursement request is rejected. It is translated to ProMeLa process as shown in Listing 7.

LISTING 7. NOTIFY EMPLOYEE OF REJECTION PROMELA PROCESS

```
proctype Notification() {
  int x;
  N:
  receive(cS[6],x); /* Receive from Approval Review
                       by Supervisor * /
  send(cS[9],1);/* Send to end process */
  goto T;
}
```

The description corresponding to the *expense reimbursement process* described in *Expense reimbursement.pml* is shown in Listing 8.

LISTING 8. EXPENSE REIMBURSEMENT.PML

```
chan cS[9]=[1] of {int};
proctype ReceiveExpenseReport(){...}
proctype createExpenseAccount(){...}
proctype PreApproval(){...}
proctype AutoApp(){...}
proctype AppBSupervisor(){...}
proctype TransferMoney(){...}
proctype Notification(){...}
proctype end(){...}
/* Run processes */
init {
  atomic{
   run ReceiveExpenseReport();
   run createExpenseAccount();
   run PreApproval();run AutoApp();
   run AppBSupervisor();run TransferMoney();
   run Notification();run end() ;
  }
}
```

*2) Compliance checking rules generation:*The following description corresponds to the compliance rules expressed in LTL formulae, described in *Expense_reimbursement_CRules.prp.*

```
ltl R1 {[](( !ReceiveExpenseReport:exist==1) ->
<> createExpenseAccount@C )}
ltl R2 {[]((PreApproval: Amount<200)-> AutoApp@A)}
ltl R3 {[](<>( AppBSupervisor:approved == 1) ->
(( TransferMoney@T && ! Notification@N) U
(! TransferMoney@T && Notification@N )))}
```

The automatically generated ProMeLa models for *Expense Reimbursement process* and the compliance rules expressed in temporal logic formulae are fed into an EpiSpin model checker. It checked the compliance of the process models, or otherwise they obtain a counterexample, which signifies that, the given BPMN process model do not conform to the compliance rules. The Fig. 3.1 shows the input ProMeLa model file *Expense_reimbursement.pml* and the input LTL file *Expense_reimbursement_CRules.prp* provided to EpiSpin plug-in and the Fig. 3.2 shows the result of verification.

V. CONCLUSION

In this article, we present a formal approach for the compliance checking of BPMN process models. It proposes to translate the BPMN process model into finite-states machine (Kripke structure) and also to express the compliance rules in LTL formulae through intermediary G-LTL notation to obtain a same level of abstraction as the process models.

The proposed approach helps to check whether or not a process model satisfies the requested compliance rule, by means of model checking technique.

A counterexample is produced when the verification of temporal properties is not satisfied by the process model.

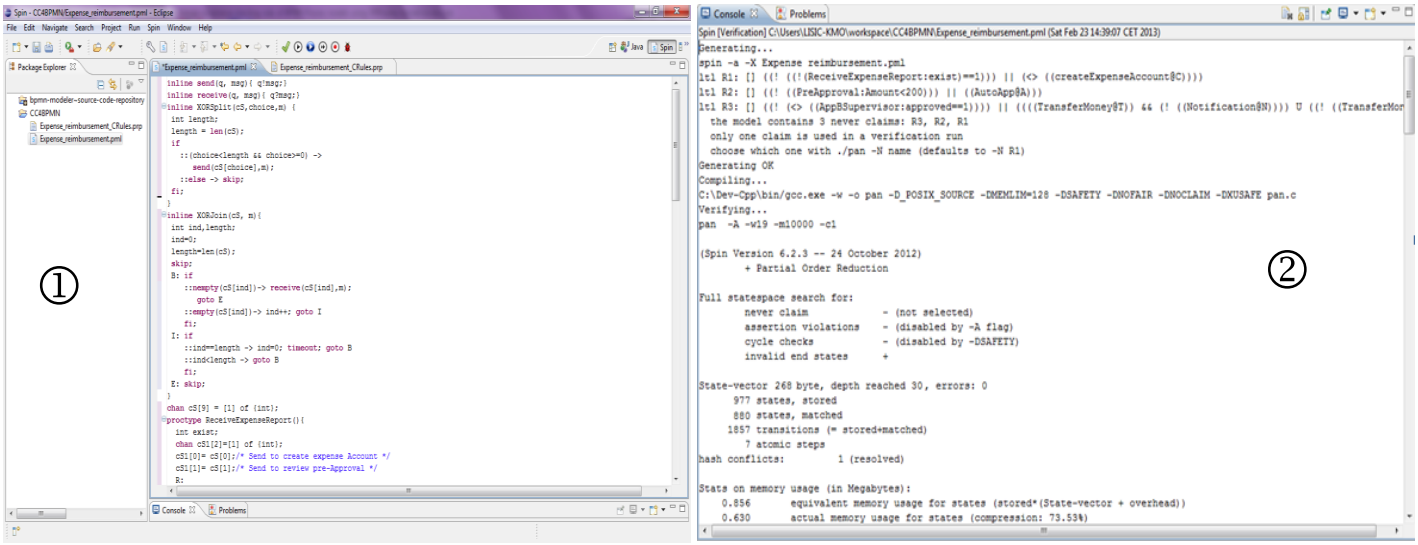


Fig. 3. The result of checking

The counterexamples may help to notify the actual causes of error generation in source BPMN model.

We have been validated the approach with the help of Eclipse IDE plug-ins development. In this article, the approach is evaluated using a basic example, which illustrates the verification of a given set of compliance rules.

Currently, our research work in the area of compliance checking is focused on verification of control flow aspects. In the future, we intend the verification of data objects and their states before and after each change in the process models, more specifically the BPMN process models.

REFERENCES

[1] R.Lu, "Constraint-Based Flexible Business Process Management," in School of Information Technology and Electrical Engineering, University of Queensland, 2008.

[2] W.M.P. Van der Aalst, A. ter Hofstede, M. Weske , "Business Process Management: A Survey," In Proceedings of Conference on Business Process Management (BPM 2003), Eindhoven, Netherlands 2003.

[3] I. Kitzmann, C. Konig, D. Lubke, L. Singer , "A simple algorithm for automatic layout of bpmn processes," In Proceedings of the IEEE Conference on Commerce and Enterprise Computing (CEC'09), 2009, pp. 391-398.

[4] M. El Kharbili, A.K. Alves de Medeiros, S. Stein, W.M.P. Van der Aalst, "Business Process Compliance Checking Current State and Future Challenges," In Proceedings of Modellierung betrieblicher Informations syteme (MobIS'08), 2008, pp. 107-113.

[5] W.M.P. Van der Aalst, H.T. de Beer, B.F. Van Dongen, "Process mining and verification of properties: An approach based on temporal logic," In Proceedings of the 2005 Confederated international conference on On the Move to Meaningful Internet Systems (OTM'05), 2005, pp. 130-147.

[6] M. Kovcs, L. Gnczy, "Simulation and Formal Analysis of Workflow Models," In Proceedings of the Fifth International Workshop on Graph Transformation and Visual Modeling Techniques, 2006, pp. 215-224.

[7] Y. Lui, S. Müller, K. Xu, "A static compliance-checking framework for business process models," In Proceedings of IBM Systems Journal 46(2), 2007, pp. 335-362.

[8] A. Awad, G. Decker, M. Weske, "Efficient Compliance Checking using BPMN-Q and Temporal Logic," In Proceedings of the 6th International Conference on Business Process Management (BPM'08), 2008, pp. 326-341.

[9] A. Awad , "BPMN-Q: A Language to Query Business Processes," In Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07), 2007, pp.115-128.

[10] F. Puhlmann, M. Weske, "Using the pi-calculus for formalizing workflow patterns," In Proceedings of the 3rd International Conference on Business Process Management (BPM'05), 2005, pp. 153-168.

[11] C Vaz, C. Ferreira, "Formal Verification of Workflow Patterns with SPIN," In Technical Report, April 2007. INESC-ID Tec. Rep. 12.

[12] H. Song H, K.J. Compton, "Verifying pi-calculus processes by Promela translation," In Technical Report CSE-TR-472-03, Univ. of Michigan, 2003.22.M.E.Stickel.

[13] M.C. Browne, E.M. Clarke, O. Grümberg, "Characterizing finite Kripke structures in propositional temporal logic" In Theoretical Computer Science - Volume 59 Issue 1-2, July 1988, pp. 115-131.

[14] K. Y. Rozier, "Linear Temporal Logic Symbolic Model Checking," In Computer Science Review -Volume 5, Issue 2, May 2011, pp. 163–203.

[15] G.J. Holzmann, "The Model Checker SPIN," In Proceedings of IEEE Transactions on software Engineering Special issue on formal methods in software practice, 1997, pp. 279-295.

[16] A. Foerster, G. Engels, T. Schattkowsky, "Activity diagram patterns for modeling quality constraints in business processes," In Proceedings of the 8th international conference on Model Driven Engineering Languages and Systems (MoDELS'05), 2005, pp. 2-16.

[17] M.O Kherbouche, A. Ahmad, H. Basson, "Using model checking to control the structural errors in BPMN models," In Proceedings of 7th IEEE International Conference on Research Challenges in Information Science (RCIS'13), 2013, pp. 1-12.

[18] Z. Manna, A. Pnueli, "The temporal Logic of Reactive and Concurrent Systems," In The temporal logic of reactive and concurrent systems, 1992, pp. 427.

[19] G.J. Holzmann, D. Peled , "The State of SPIN," In Proceedings of Computer Aided Verification (CAV'96), 1996, pp. 385-389.

[20] P. Gastin, D. Oddoux, "Fast LTL to Büchi Automata Translation," In Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01), 2001, pp. 53-65.

[21] F. Somenzi, R. Bloem, "Efficient Büchi Automata from LTL Formulae," In Proceedings of the 12th International Conference on Computer Aided Verification (CAV '00), 2000, pp. 247-263.