# Ontology-Based Workflow Validation

Tuan Anh Pham, Thi Hoa Hue Nguyen, Nhan Le Thanh

# Ontology-Based Workflow Validation

Tuan Anh Pham
WIMMICS – I3S – CNRS - INRIA
University of Nice Sophia Antipolis
Sophia Antipolis, France
tuan-anh.pham@inria.fr

Thi-Hoa-Hue Nguyen
WIMMICS – I3S – CNRS - INRIA
University of Nice Sophia Antipolis
Sophia Antipolis, France
nguyenth@i3s.unice.fr

Nhan Le Thanh
WIMMICS – I3S – CNRS - INRIA
University of Nice Sophia Antipolis
Sophia Antipolis, France
nhan.le-thanh@inria.fr

***Abstract*—In order to ensure a workflow to be executed correctly, many approaches were introduced. But not many of them consider the semantic correctness of the workflow in the design time and the run time. In this paper, a solution to check the semantic correctness of the workflow automatically is presented. To do that, the workflow must be represented in a machine understandable form, an ontology-based approach to represent a workflow is proposed. In addition, we also provide a set of changed operations allowing the users to customize a workflow for using in their organizations. Their change can be made while ensuring the correctness of the workflow. Moreover, a verification method is proposed for checking the semantic correctness of workflow.**

***Keywords—Workflow; Ontology; Coloured Petri Net; Workflow verification***

## I. INTRODUCTION

A workflow is a set of related activities which can be executed in a certain order. For a long time, the workflow was studied by the company [1], [2]. They have been trying to make it more intelligent and more flexible, one of the most important point is to verify the correctness of the workflow automatically at the design time and during the run time to allow the workflow can be executed correctly.

Nowadays, many works have been done on workflow verification. However, a complete solution for checking the correctness of workflow is rarely considered. Many researchers focus on the control-flow aspect, such as [7], [8], [9], [10], [11] to prevent errors (e.g., avoiding deadlocks, infinite cycles) at the syntactic level. Nevertheless, they mainly check the conformance of a workflow process based on the principle that if the constraints on data and control flow are met during execution, the workflow is correct. There are some teams proposing only the idea, they do not consider the validation of workflow such as [19]. This team has the same ontological approach with us but they did not consider the semantic correctness and syntactic correctness [8] at the design time and the runtime. With our approach, we use Coloured Petri Net [21] to verify the syntactic correctness of a workflow at the design time.

We realize that there are not many researchs on checking the semantic correctness. Checking semantic correctness of workflow is to ensure that a workflow is designed and redesigned in compliance with some predefined rules in a domain. Let us take an example, a user creates a process of the Order Management Activity, he cannot create the task "Evaluated results" before the task "Order request" because at that time the request is not sent. This type of constraint is called "Semantic Constraint".

In this paper, we restrict ourselves to ensure the semantic correctness of a workflow processes and provide a set of changed operations to help the users customize a workflow to be corresponded to their requirements while their change does not affect to the semantic and syntactic constraints. Our contributions are:

- Giving a formal method to describe a variety of semantic constraints;
- Developing an ontology for annotating semantic constraints and representing control flow-based business workflow processes based on that ontology;
- Giving a set of changed operations to allow the users customize the workflow;
- Showing how to use the SPARQL query language [6] to check the semantic correctness of workflow processes.

This paper is organized as follows: A short introduction to the CPN Ontology, which is defined to represent Coloured Petri Nets (CPNs) [21] with OWL DL, is given in Section II. Section III proposes a formal definition of semantic constraints for business processes. We then develop a semantic conformance-oriented ontology. In Section IV, we present the creation of correspondences between these two ontologies to develop workflow processes. In section V, we propose a set of changed operations to customize the workflow. Five semantic verification issues of a workflow process are introduced in Section VI. Finally, Section VII concludes the paper with an outlook on the future research.

## II. REPRESENTATION OF COLOURED PETRI NET WITH OWL DL ONTOLOGY

In this Section, we introduce the Coloured Petri Net Ontology [21] defined for business processes modelled with Coloured Petri Net (CPNs). The purpose of this ontology is to ensure the syntactic correctness of workflow [8] processes and to facilitate business process models for sharing and reusing.

On one hand, Coloured Petri Nets (CPNs) have been developed into a full-fledged language for the design, specification, simulation, validation and implementation of large software systems. Consequently, modelling business

processes with CPNs supports workflow designers easy to verify the syntactic correctness of workflow processes [8]. On the other hand, OWL DL, which stands for OWL Description Logic, is equivalent to Description Logic SHOIN(D). OWL DL supports all OWL language constructs with restrictions (e.g., type separation) and provides maximum expressiveness while keeping always computational completeness and decidability. Therefore, we choose OWL DL language to represent the CPN Ontology. We believe that the combination of CPNs and OWL DL provides not only semantically rich business process definitions but also machine-processable ones. Fig. 1 depicts the core concepts of the CPN ontology.

The CPN Ontology comprises the concepts: "CPNOnt" defined for all possible CPNs: "Place" defined for all places; "Transition" defined for all transitions; "InputArc" defined for all directed arcs from places to transitions; "OutputArc" defined for all directed ares from transitions to places; "Token" defined for all tokens inside places (we consider the case of one place containing no more than one token at one time); "GuardFunction" defined for all transition expressions; "CtrlNode" defined for occurrence condition in control nodes; "ActNode" defined for occurrence activity in activity nodes, "Delete" and "Insert" defined for all expressions in input arcs and output arcs, respectively; "Attribute" defined for all attributes of individuals); "Value" defined for all subsets of $I_1 \times I_2 \times \ldots \times I_n$ where $I_i$ is a set of individuals.

Properties between the concepts in the CPN Ontology are also specified in Fig 1. For example, the concept "CPNOnt" is defined with three properties "hasPlace", "hasTrans" and "hasArc". It can be glossed as 'The class CPNOnt is defined as the intersection of:' (i) any class having at least one property "hasPlace" whose value restricted to the class "Place" and; (ii) any class having at least one property "hasTransition" whose value is restricted to the class Transition and; (iii) any class having at least one property "hasArc" whose value is either restricted to the class "InputArc" or the class "OutputArc".

$$CPNOnt \equiv \geq 1 hasTrans.Transition \sqcap \geq 1 hasPlace.Place \sqcap \geq 1 hasArc.(InputArc \sqcup OutputArc)$$
$$Place \equiv connectsTrans.Transition \sqcap = \leq 1 hasMarking.Token$$
$$Transition \equiv connectsPlace.Place \sqcap 1 hasGuardFunction.GuardFunction$$
$$InputArc \equiv \geq 1 hasExpresion.Delete \sqcap \exists hasPlace.Place$$
$$OutputArc \equiv \geq 1 hasExpresion.Insert \sqcap \exists hasTrans.Transition$$
$$Delete \equiv \forall hasAttribute.Attribute$$
$$Insert \equiv \exists hasAttribute.Attribute$$
$$GuardFunction \equiv \geq 1 hasAttribute.Token \sqcap = 1 hasActivity.ActNode \sqcup = 1 hasControl.CtrlNode$$
$$Token \equiv \geq 1 hasAttribute.Attribute$$
$$Attribute \equiv \geq 1 valueAtt.Value$$
$$ActNode \equiv = 1 valueAtt.Value$$
$$CtrlNode \equiv \leq 1 valueAtt.Value$$
$$Value \equiv valueRef.Value$$

Fig. 1 : Coloured Petri Net ontology

## III. SEMANTIC CONSTRAIN FOR BUSSINESS PROCESS

As mentioned previously, our work aims at representing workflow processes modelled with CPNs is a knowledge base. Therefore, in this section, we focus on ensuring their quality by guaranteeing their semantic correctness.

### *A. Definition of Semantic Constraints*

By talking account domain experts in support of modellers at build time, a set of semantic constraints is specified, which then is used to develop a corresponding workflow. According to [13], there are two fundamental kinds of semantic constraints, including mutual exclusion constraints and dependency constraints. For interdependent tasks, e.g., the presence of task A indicates that task B must be included, however, task B can be executed while task A is absence. In fact, there may exist tasks that are coexistent. This refers to the coexistence constraints. Consequently, we propose three basic types: mutual exclusion constraints, dependency constraints and coexistence constraints.

**Definition 1** (Semantic Constraint): Let T be a set of tasks. A semantic constraint:
c = (constraintType, appliedTask, relatedTask, order, description,[Equivalence]) where:

- constraintType∈{mExclusion,dependency,coexistence };
- appliedTask ∈ T;
- relatedTask ∈ T;
- order ∈ {before, after, concurrence, notSpecified};
- description is an annotation of the constraint;
- Equivalence is a set of tasks which are equivalent to task appliedTask.

In Definition 1, the first parameter "constraintType" denotes the type of a semantic constraint. Each value of "constraintType" refers to the relationship between the executions of the source task denoted by the second parameter "appliedTask" and the target task denoted by the third parameter "relatedTask". Parameter "order" specifies the order between the source and target tasks in a process model. The first four parameters are very important when defining a semantic constraint. The fifth parameter, "description", is used for describing the constraint. "Equivalent3 is an optional parameter, which contains a set of tasks (if any) being equivalent to the source task.

Let us continue the example of a process of the Order Management activity. The process is determined as follows: After receiving an order, two tasks have to do in parallel are "authenticate client" and "check availability". If both of these tasks result "true", the order is accepted. An order confirmation is sent out. In contrast, an order refusal is sent out, etc. Some semantic constraints of the process are formed as follows:

*c1 = (dependency, authenticate client, receive request, before, receiving an order has to be performed before authenticating client, {authenticate purchaser});*
*c2 = (dependency, check availability, receive request, before, receiving an order has to be performed before checking availability);*
*c3 = (coexistence, authenticate client, check availability, concurrence, client authentication and checking availability are performed in parallel);*

*c4 = (dependency, evaluate results, authenticate client, before, evaluating the results obtained from the relevant departments);*
*c5 = (dependency, evaluate results, receive request, before, receiving an order has to be performed before evaluating results related to the order)*

### B. *Development of a Semantic conformance-oriented Ontology*

Our work aims at representing processes modelled with CPNs in a knowledge base. Therefore, to provide a representation of semantic constraints related to process elements, we develop an approach for constructing a new ontology. This ontology is oriented to semantic conformity checking in workflow processes. We focus on formalizing the concepts/relations corresponding to the knowledge that is required by model elements.

The following keystones to transform a set of semantic constraints into an OWL DL ontology:

- Each semantic constraint c is mapped to an instance of *owl :Class*.
- "appliedTask" and "relatedTask" are mapped into two instances of owl :Class. The *rdfs:subClassOf property* is used to state that these classes is a subclass of the constraint class.
- Each value of "constraitType" or order is defined as an instance of the built-in OWL class *owl:ObjectProperty*.
- Description is defined as an instance of the built-in OWL class *owl:Datatype* Property;
- Each value in the set Equivalence is mapped to an instance of *owl:Class*. The built-in property *owl: equivalentClass* is used to link every class description of these classes to the class description of "appliedTask".

In the next Section, we will discuss about the integration of a semantic conformance-oriented ontology (domain knowledge) and the CPN Ontology to create workflow processes.

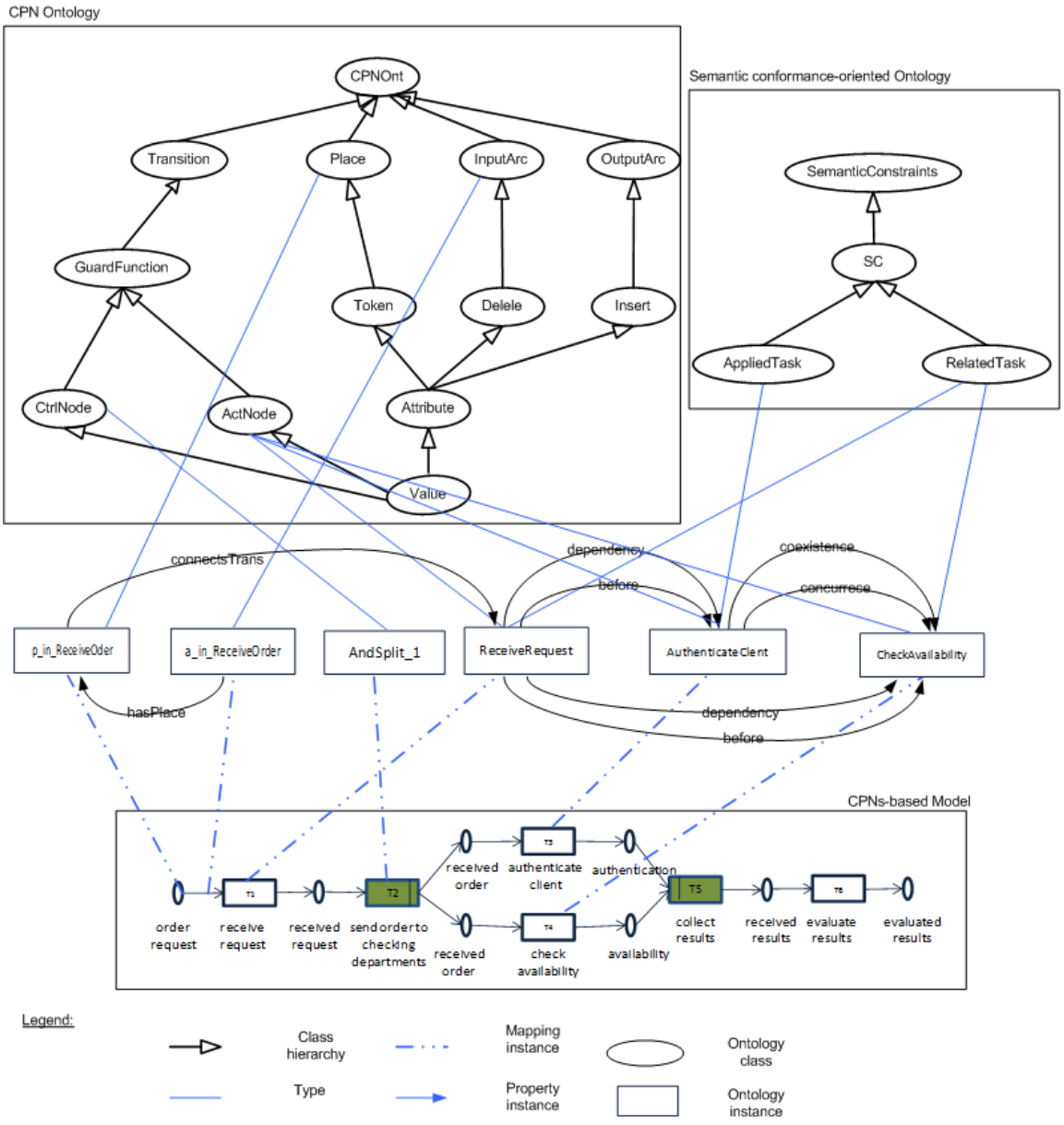


Fig. 2 : An example of ontology mapping

## IV. CREATION OF CORRESPONDENCES BETWEEN ONTOLOGIES

We rely on ontology mapping techniques for matching semantics between ontologies, i.e., the CPN Ontology and Domain Ontology (a semantic conformance-oriented ontology). In our case, the articulation of two ontologies is used not only for creating semantically workflow processes, but also for verifying their correctness.

We now define our use of the term “mapping": Consider two ontologies, $O_1$ and $O_2$. Mapping of one ontology with another is defined as bringing ontologies into a mutual agreement in order to make them consistent and coherent. It means that for a concept or a relation in ontology $O_1$, we try to find the same intended meaning in ontology $O_2$ ; For an instance in ontology $O_1$, we find the same instance in ontology $O_2$.

**Definition 2** (Mapping related to the “before” property) Give an instance, $I_C$, of a semantic constraint in which the order between the instance of class “appliedTask”, named $task_a$, and the instance of class “relatedTask”, named $task_b$, is indicated by the object property before. The type of instance $I_C$ is either dependency or coexistence. A set of correspondences is determined as follows:

- Each instance of class “appliedTask” or “relatedTask” is mapped into an instance of class Transition (expresses activity node).
- There exists a firing sequence $t_1t_2 \dots t_n$ , where $t_1$; $t_n$ are the instances of class T ransition corresponding to instances $task_a$ and $t_b$ respectively, $t_a = t_1$, $t_b = t_n$ $n \geq 2$.

**Definition 3 (**Mapping related to the “concurrence” property**)**

Give an instance, $I_C$, of a semantic constraint in which the order between the instance of class “appliedTask”, named $task_a$, and the instance of class “relatedTask”, named $task_b$, is indicated by the object property concurrence. The type of instance $I_C$ is coexistence. A set of correspondences is determined as follows:

- Each instance of class “appliedTask” or “relatedTask” is mapped into an instance of class “Transition” (expresses activity node).
- Two instances of class transitions which correspond to instance $task_a$ and instance $task_b$ can be enabled at the same time.

It is important to note that object property “before” is the symmetrical property of object property “after”. Consequently, we do not define a mapping related to the “after” property.

By continuing the process schema for the Order Management Activity in Section 4, Figure 1 shows the mapping of some instances between two ontologies, CPN Ontology and Semantic Conformance-oriented Ontology.

We have introduced the formal definition of semantic constraints and illustrated how to model a workflow process with CPNs based on specified semantic constraints. Note that concrete workflow processes are represented in RDF syntax. Moreover, to develop or modify a workflow process, manipulation operations [20] (e.g., inserting a new element) are required. Therefore, it is necessary to verify workflow processes at the design time before using it.

## V. CUSTOMIZING OF WORKFLOW

In this section, we introduce a set of changed operations to help the users modify the workflow to be corresponded to their requirements. Each customized workflow we consider like a workflow instance which must respect the set of predefined semantic constraints and set of predefined syntactic constraints.

In the Semantic conformance-oriented Ontology, we have a set of terminologies and constraints in many domains. But when an user download a template to modify it, maybe they want to modify the set of terminologies to be corresponded to their system, so that is why we provide operation change “Mapping terminology”.

TABLE I. SET OF CHANGED OPERATIONS

| **Changed Operations** | | |
|---|---|---|
| **Operation** | **Template** | **Instance** |
| Mapping terminologies | Terminology templates | User’s terminologies |
| Split | A Node | Set of nodes |
| Merge | Set of nodes | A Node |
| Insert/Remove | N node | N+1 node/ N-1 node |

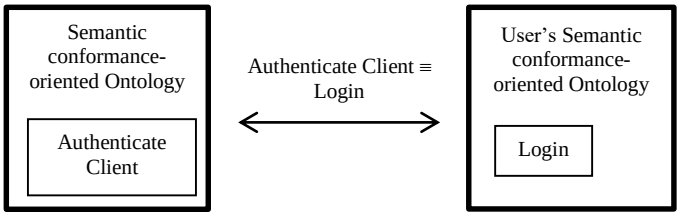


Fig. 3. Mapping of terminologies

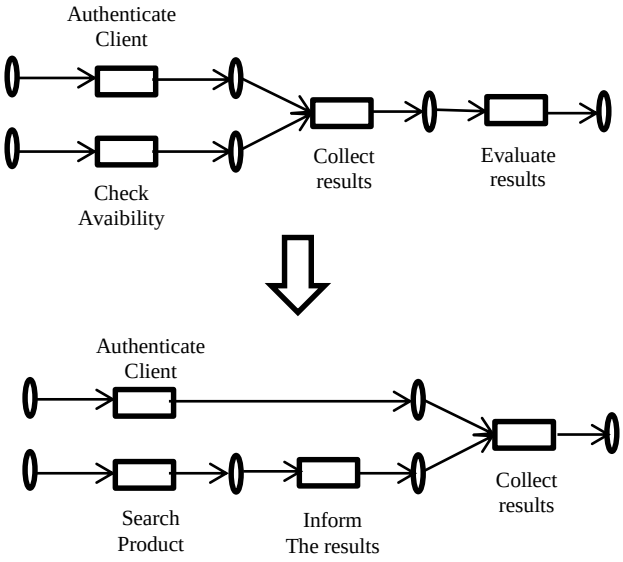


Fig. 4. Splited operation

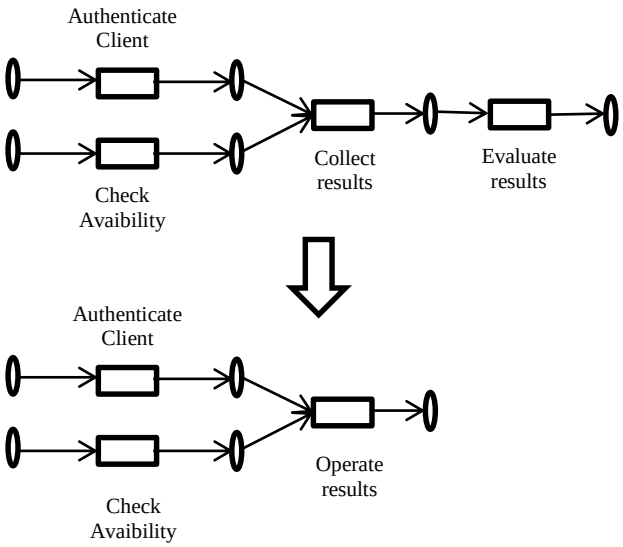


Fig. 5. Merged operation

Let us continue the previous example. There are someones who download this workflow to reuse it, but they want to use the terminology "Login" instead of the terminology "Authenticate Client". we have a function to help him to create a Semantic conformance-oriented Ontology which contain his own set of terminologies. The mapping between two Semantic conformance-oriented ontology is created also. "Split" operation allows users to split a node into many nodes, the set of splited nodes can be executed sequentially or in parallel.

In the Fig. 4, the task "Check availability" is splited into two tasks, "Search product" and "Inform the results". Before do this operation, two terminologies above must be inserted into the user's Semantic conformance-oriented Ontology. The relations between the new terminologies and the other one must be generated depending on the old terminologies's relations.

We also provide the "Merged operation" to allow the users to merge two tasks into one task. Continuing the previous example, in Fig 4, two tasks "Collect results" and "Evaluate results" are merged into one task "Operate results". In this case, to respect the semantic constraints, the new terminology "Operate results" must be inserted into the user's Semantic conformance-oriented Ontology and have the same relations with other terminologies which have a relation with two terminologies "Collect results" and "Evaluate results".

"Insert/Delete" operation help users to add a node into a workflow or remove a node. When the user insert a node into a workflow, must add a terminology into his Semantic conformance-oriented Ontology before using it to create a node in the workflow.

## VI. SEMANTIC VERIFICATION ISSUE

We here pay attention to the research question relating to semantic verification: Is the behavior of the individual activities satisfied and conformed with the control flow? To answer this question, we address the following semantic verification issues:

- Are there activities whose occurrences are mutual exclusion, but that may be executed in parallel or in sequence?
- Are there activities whose executions are interdependent, but that may be carried out in choice or in parallel?
- Are there activities whose occurrences are coexistent, but that may be executed in a choice?
- Are there any couples of activities whose order executions are defined as one before the other, but that may be executed in the opposite?
- Are there any couples of activities whose order executions are defined as one after the other, but that may be executed in the opposite order?

Because concrete workflows are stored in RDF syntax, we rely on the CORESE [25] semantic search engine for answering SPARQL queries asked against an RDF knowledge base. We initiate SPARQL queries to verify whether workflow processes contain semantic errors or not. SELECT query form is chosen for this work. After a SELECT keyword, the variables are listed that contain the return values. And in the WHERE clause, one or more graph patterns can be specified to describe the desired result.

The following query relating to the third verification issue is used to query if the model contains `any pairs of activities whose occurrences are coexistence but that may be executed in choice'. The properties "h:coexistence" and "h:concurrence" defined in the first ontology indicates the semantic constraint between activities ?t1 and ?t2. On the other hand, the other properties defined in the second ontology which represent these activities restricted to the control flow perspective. By applying this query to the workflow example depicted in Figure 1, the result is empty.

The sample query does not only demonstrate that the SPARQL query language is able to check the semantic correctness of workflow processes, but also the usage of terminological background knowledge provided by the semantic conformance-oriented ontology and CPN Ontology.

Moreover, by representing CPNs-based business processes with OWL DL ontology we can also verify the soundness of models. This means that we can check syntactic errors (for example, deadlocks, infinite cycles and missing synchronization, etc.) by the SPARQL query language.

```
SELECT ?t1 ?t2 WHERE
{
    ?t1 rdf:type h:Transition
    ?t2 rdf:type h:Transition
    ?t3 rdf:type h:Xor-split
    ?t4 rdf:type h:Xor-join
    ?t1 h:coexistence ?t2
    ?t2 h:concurrence ?t1
    ?t3 h:connectsPlace/h:connectsTrans ?t1
    ?t3 h:connectsPlace/h:connectsTrans ?t2
    ?t1 h:connectsPlace/h:connectsTrans ?t4
    ?t2 h:connectsPlace/h:connectsTrans ?t4
    FILTER (?t1!=?t2)
}
```

## VII. CONCLUSION

This paper presents an approach to define and customize a workflow. First, we propose a formal method represents the semantic constraint which is used to ensure the semantic correctness of a workflow. To integrate the domain knowledge used for annotating the process elements, we develop a semantic conformance-oriented ontology. This ontology is then matched with the CPN Ontology (a representation of CPNs with OWL DL). We use the mapping two ontologies to verify the semantic correctness of a workflow. Second, we provide some changed operations which allow the users to customize a workflow to be corresponded to their requirements.

With the future work, we will try to resolve following problems:

- Verifying the semantic correctness and syntactic correctness of the workflow when an user makes a change in a workflow.
- Explaining in more detail about the conditions to merge a set of nodes into a node and split a node into a set of nodes, insert and delete a node in the workflow.
- Considering version control when an user makes a change on the workflow, the system can be rollback to the previous step.
- Working with other more complex constraints, not only dependency, coexistence and exclusion.
- Enriching the set of terminologies.

## REFERENCES


[1] Microsoft : Window workflow foundation. Version 4.5. Available from http://msdn.microsoft.com/en-us/vstudio/jj684582.aspx

[2] Oracle : Oracle workflow for user. Version 2.6. Available from http://docs.oracle.com/cd/B14117_01/workflow.101/b10285/ugov.htm

[3] Barros, A.P., ter Hofstede, A.H.M., Proper, H.A.: Essential principles for workflow modelling effectiveness. In: PACIS. (1997) 15

[4] Koschmider, A., Oberweis, A.: Ontology based business process description. In: EMOI-INTEROP, Springer (2005) 321-333

[5] Fellmann, M., Thomas, O., Busch, B.: A query-driven approach for checking the semantic correctness of ontology-based process representations. In: BIS. (2011) 62-73

[6] W3C: Sparql 1.1 query language. http://www.w3.org/TR/sparql11-query/ (March 2013) W3C Recommendation

[7] van der Aalst W.M.P.: Verification of workflow nets. In: ICATPN. (1997) 407-426

[8] Verbeek, H., Basten, T., van der Aalst, W.: Diagnosing workflow processes using woflan. The computer journal **44** (1999) 246-279

[9] Bi, H.H., Zhao, J.L.: Applying propositional logic to workflow verification. Information Technology and Management **5**(3-4) (2004) 293-318

[10] Wainer, J.: Logic representation of processes in work activity coordination. In: Proceedings of the 2000 ACM Symposium on Applied Computing - Volume 1. SAC'00, New York, NY, USA, ACM (2000) 203-209

[11] Sadiq, W., Maria, Orlowska, E.: Analyzing process models using graph reduction techniques. Information Systems **25** (2000) 117-134

[12] Lu, S., Bernstein, A.J., Lewis, P.M.: Automatic workflow verification and generation. Theor. Comput. Sci. **353**(1-3) (2006) 71-92

[13] Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. Data Knowl. Eng. **64**(1) (2008)3-23

[14] Kumar, A., Yao, W., Chu, C.H., Li, Z.: Ensuring compliance with semantic constraints in process adaptation with rule-based event processing. In: RuleML. (2010) 50-65

[15] Ly, L.T., Rinderle-Ma, S., G • oser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems - requirements, challenges, solutions. Information Systems Frontiers **14**(2) (2012) 195-219

[16] Thomas, O., Fellmann, M.: Semantic process modeling - design and implementation of an ontology-based representation of business processes. Business & Information Systems Engineering **1**(6) (2009) 438-451

[17] Weber,I.,Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distributed and Parallel Databases **27**(3) (2010) 271-343

[18] Gasevic, D., Devedzic, V.: Interoperable petri net models via ontology. Int. J. Web Eng. Technol. **3**(4) (2007) 374-396

[19] Sebastian, A., Tudorache, T., Noy, N.F., Musen, M.A.: Customizable workflow support for collaborative ontology development. In: 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE) at ISWC 2008. (2008)

[20] Nguyen, T.H.H., Le-Thanh, N.: An ontology-enabled approach for modelling business processes. In: Beyond Databases, Architectures and Structures. Volume 424 of Communications in Computer and Information Science. Springer International Publishing (2014) 139-147

[21] Kristensen, L.M., Christensen, S., Jensen, K.: The practitioner's guide to coloured petri nets. STTT 2(2) (1998) 98-132

[22] Ellis, C.A., Nutt, G.J.: Modeling and enactment of work ow systems. In: Application and Theory of Petri Nets. (1993) 1-16

[23] W3C: Owl web ontology language reference. http://www.w3.org/TR/owl-ref/ (2004) W3C Recommendation.

[24] van der Aalst, W.M.P.: The application of petri nets to work ow management.Journal of Circuits, Systems, and Computers 8(1) (1998) 21-66

[25] Corby, O., et al.: Corese/kgram. https://wimmics.inria.fr/corese