An Ontology-Enabled Approach for Modelling Business Processes

Thi Hoa Hue Nguyen, Nhan Le Thanh

An Ontology-enabled Approach for Modelling Business Processes

Thi-Hoa-Hue Nguyen and Nhan Le-Thanh

WIMMICS - The I3S Laboratory - CNRS - INRIA
University of Nice Sophia Antipolis
Sophia Antipolis, France
nguyenth@i3s.unice.fr,Nhan.LE-THANH@unice.fr

Abstract. Coloured Petri Nets (CPNs) have formal semantics and can describe any type of workflow system, behavioral and syntax wise simultaneously. They are widely studied and successfully applied in modelling of workflows and workflow systems. There is an inherent problem regarding business processes modelled with CPNs sharing and subsequently their reuse need to be considered. The Semantic Web technologies, such as ontologies, with their characteristics demonstrate that they can play an important role in this scenario. In this paper, we propose an ontological approach for representing business models in a meta-knowledge base. Firstly, the CPN ontology is defined to represent CPNs with OWL DL. Secondly, we introduce four basic types of manipulation operations on process models used to develop and modify business workflow patterns. To the best of our knowledge, representing business process definitions and business workflow patterns as knowledge based upon ontologies is a novel approach.

Keywords: Business Process, CPN, Manipulation Operation, OWL DL Ontology, Representing

1 Introduction

To date, software systems that automate business processes have been becoming more and more available and advanced. According to [1], process models, which are first designed during built-time phase on the basis of design requirements, are then automated by software systems during run-time. Therefore, grasping the requirements properly and then transforming them without losing any information into a semantically rich specification play an important role in supporting business process management. However, the existing practice of modelling business processes is mostly manual and is thus vulnerable to human error, resulting in a considerable number of failed projects. Consequently, it is desirable to develop an alternative approach, which ensures high quality and semantically rich business process definitions.

On one hand, Coloured Petri Nets (CPNs) [2] have been developed into a full-fledged language for the design, specification, simulation, validation and implementation of large software systems. CPNs are a well-proven language suitable

for modelling of workflows or work processes [3]. Although CPNs are widely studied and successfully applied in modelling of workflows and workflow systems, the lack of semantic representation of CPN components can make business processes difficult to interoperate, share and reuse.

On the other hand, an ontology with its components, which can provide machine-readable definitions of concepts, plays a pivotal role in representing semantically rich business process definitions. Once business process semantics are machine-processable, IT experts can easily develop their appropriate software systems from business process definitions.

Our objective, outlined in Fig. 1, is to represent Control flow-based Business Workflow Patterns (CBWPs) in a meta-knowledge base which intend to make them easy to be shared and reused among process-implementing software components. We focus on proposing an ontological model to represent Coloured Petri Nets (CPNs) with OWL DL. We first define a meta-knowledge base for CBWPs management. We then introduce manipulation operations on process models for the purpose of developing and modifying CBWPs. Our ongoing work is developing a graphical interface to design and simulate CBWPs. To the best of our knowledge, this is a novel approach for representing business process definitions and patterns as knowledge based upon ontologies.

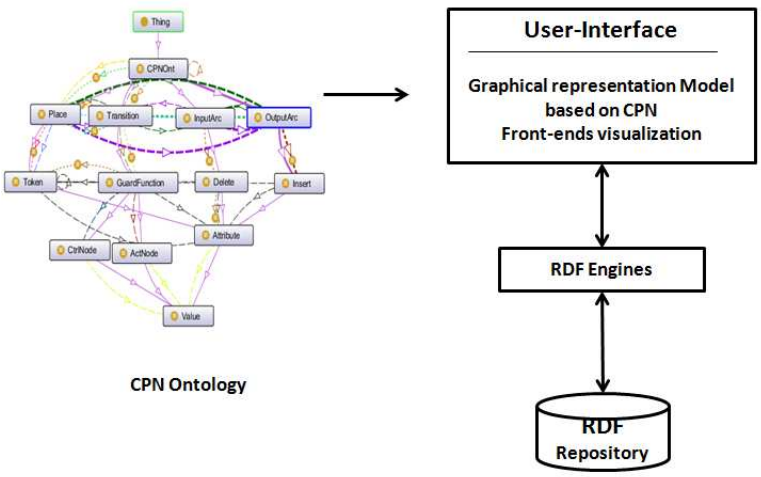

Fig. 1: An overall approach for representing Control flow-based Business Workflow Patterns (CBWPs) in knowledge base

The rest of this paper is structured as follows: In Section 2, we recall the main notions of CPNs and ontologies. In Section 3, we introduce a novel ontology for CPNs. We then elucidate the realization of our ontology. In Section 4, we introduce four basic types of manipulation operations to support the creation

and modification of CBWPs. We present related work in Section 5. Finally, Section 6 concludes the paper with an outlook on the future research.

2 Foundations

In this section, the main notions of CPNs and ontologies are introduced. On this basis, we will present an ontology for business processes in the upcoming Section.

2.1 Coloured Petri Nets

Coloured Petri Nets (CPNs) are extended from Petri nets with colour, time and expressions attached to arcs and transitions. A CPN, which is a directed bipartite, consists of *places* (drawn as ellipses) and *transitions* (drawn as rectangles) connected by *directed arcs* (drawn as arrows). Each place holds a set of markers called *tokens*. The number of tokens in a place can vary over time. Each token can carry both a data value called its colour and a time-stamp. The type of tokens in a place is the same type as the type of the place.

Since transitions may consume and produce tokens, it is necessary to use *arc expressions* to determine the input-output relations. An incoming arc indicates that tokens may be removed by the transition from the corresponding place while an outgoing arc indicates that tokens may be added by the transition.

In addition, CPNs allow us to structure the descriptions in a hierarchical way. Large models may be divided into sub-models, also known as sub-processes or modules, in order to get a layered hierarchical description. As a result, sub-models can be reused. For more details on CPNs, please refer to [2].

The Workflow Management Coalition (WfMC) [4] identified four routing constructs including *sequential*, *parallel*, *conditional* and *iteration*. We use five building blocks, i.e., $Sequence$, $And-split$, $And-join$, $Xor-split$, $Xor-join$ to model these types of routing. Consequently, a routing construct might contain control nodes (transitions in the building blocks) and activity nodes (other transitions). An example of a business process modelled with CPNs is shown in Fig. 2. This process model comprises the activity nodes T_1, T_3, T_4, T_6, T_8, T_9 and T_{11} connected by four control nodes T_2, T_5, T_7, T_{10}.

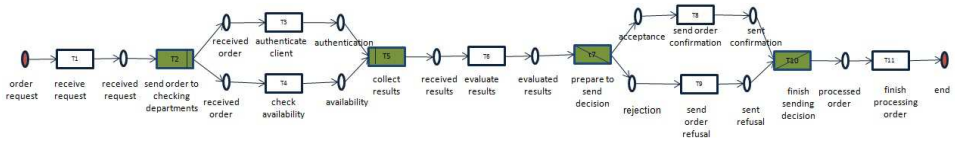

Fig. 2: An example of an order management process

2.2 Ontologies

Ontology definition languages such as RDFS[1], OIL[2] or OWL[3] can be used to define ontologies. Here, we focus on the Web Ontology Language (OWL), a W3C Recommendation, which is "a family of knowledge representation languages for authoring ontologies" [5]. OWL ontologies can be categorised into three sub-languages OWL Lite, OWL DL, and OWL Full.

OWL DL, which stands for OWL Description Logic, is equivalent to Description Logic $\mathcal{SHOIN}(\mathcal{D})$. OWL DL supports all OWL language constructs with restrictions (e.g., type separation), provides maximum expressiveness while always keeping computational completeness and decidability. Therefore, we choose OWL DL language for our work with the aim of taking advantage of off-the-shelf reasoning technologies. For more details on OWL DL, please refer to [6].

It is important to underline that an ontology consists of several different components including classes (concepts), individuals (instances), attributes, relations and axioms, etc. Therefore, by using OWL DL language to formulate an ontology, we get an OWL DL ontology that contains a set of OWL DL (class, individual, data range, object property, data type property) identifiers, and a set of axioms used to represent the ontology structure and the ontology instances.

3 An Ontology for Business Processes modelled with Coloured Petri Nets

3.1 Representation of Coloured Petri Net with OWL DL Ontology

In this subsection, we define semantic metadata for business processes modelled with CPNs. The main purpose is to facilitate business process models easy to be shared and reused among process-implementing software components. We continue our work at [7], [8] to develop the CPN ontology. Each element of CPNs is concisely translated into a corresponding OWL concept. Fig. 3 depicts the core concepts of our CPN ontology. In the next step, we will describe the main constructs of the ontology modelled with OWL DL.

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 arcs 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.

[1] http://www.w3.org/2001/sw/wiki/RDFS
[2] http://www.w3.org/TR/daml+oil-reference
[3] http://www.w3.org/2004/OWL/

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

Fig. 3: CPN ontology expressed in a description logic

Properties between the concepts in the CPN ontology are also indicated in Fig. 3. For example, a class *Place* has two properties $hasMarking$ and $connectsTrans$. Consequently, the concept *Place* can be glossed as 'The class *Place* is defined as the intersection of: (i) any class having at least one property $connectsTrans$ whose value is equal to the class $Transition$ and; (ii) any class having one property $hasMarking$ whose value is restricted to the class $Token$'.

3.2 Realization

We rely on OWL DL and use Protégé[4], an OWL editor, to create our CPN ontology. We here describe some axioms created for the CPN ontology.

It is necessary to note that two OWL classes, $owl : Thing$ and $owl : Nothing$, are particularly predefined. The class extension of $owl : Thing$ is employed to denote the set of all individuals. The class extension of $owl : Nothing$ is the empty set. As a result, each user-defined class is absolutely a subclass of $owl : Thing$.

- The class *Place* comprises two properties $connectsTrans$ and $hasMarking$. The class axiom is created as follows:
  $Class(Place\ complete\ restriction(connectsTrans\ hasValue(Transition))$
  $\quad restriction(hasMarking\ allValuesFrom(Token)$
  $\quad qualifiedCardinality(1)))$;
- The class *Place* is a sub-concept of the class $CPNOnt$. The class axiom is created as follows:
  $SubClassOf(Place\ CPNOnt)$;
- The classes *Place* and $Transition$ are mutual disjoint. The class axiom is created as follows:
  $DisjointClasses(Place\ Transition)$;

[4] http://protege.stanford.edu/

- The domain of the property *connectionsTrans* is a union of the class *Place* with the class *InputArc*. The range of this property is a union of the class *Transition* with the class *OutputArc*. In addition, the properties *connects−Trans* and *connectsPlace* are inverse properties. We create the property axiom for *connecsTrans* as follows:
  $ObjectProperty(connectsTrans\ domain(unionOf(Place\ InputArc))$
  $range(unionOf(Transition\ OutputArc))\ inverseOf(connectsPlace))$;
- The class *Attribute* contains at least one $Value$. We create the class axiom for *Attribute* as follows:
  $EquivalentClasses(Attribute\ restriction(valueAtt\ allValuesFrom(Value)$
  $minQualifiedCardinality(1)))$;

After presenting some axioms created for *Classes* and *Properties* of our ontology, we now introduce the modelling of *Individuals* being the third OWL element. Individuals or instances are generated based on the modeller and depend on the modelling objective. For example, Fig. 4 shows the mapping of the transition *receive request* named T_1, which is depicted in Fig. 2, to the classes and properties of the CPN ontology.

```
<Transition rdf:ID="T_1">
 <hasGuardFunction rdf:resource="#Guard_Request"/>
 <connectsPlace rdf:resource="#Received"/>
</Transition>
```

Fig. 4: Mapping Individuals to Classes and Properties of the CPN ontology

We have been introducing the CPN ontology represented in OWL DL. In order to develop or modify CBWPs (CPN models), in the next Section, we will present manipulation operations on their elements. We also introduce the corresponding manipulation statements written in the SPARQL language[5] to store concrete CBWPs (CPN models) in RDF[6].

4 Manipulation Operations on Basic Business Process Models

For the purpose of modelling basis business processes with CPNs, the following basic types of manipulation operations on its elements are required:

1. Inserting new elements (i.e., places, transitions or arcs, etc.) into a process model.
2. Deleting existing elements from a process model.

[5] http://www.w3.org/TR/sparql11-query/
[6] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

3. Updating existing elements for adapting to a process model.
4. Editing the order of existing elements in a process model.

More complex operations can be developed based upon these basic operations. For example, merging two separate CBWPs, which represent two process models into one, can be considered as inserting all places, transitions and arcs from one pattern to the other. Besides, one new arc is also inserted in order to link these CBWPs.

We next define the basic manipulation statements by the corresponding pseudocodes. With the aim of storing CBWPs in RDF, we introduce the SPARQL statements being suitable to the manipulation statements.

- Inserting new elements into a process model.
  $INSERT\ ELEMENT\ \{e_1, e_2, \ldots, e_n\}\ INTO\ PROCESS\ wf$
  $[WHERE\ cond_1, cond_2, \ldots, cond_m];\ (n \geq 1, m \geq 1)$
  This statement means that elements $e_1, e_2, \ldots, e_n$, each of which has been created, are inserted into a process model named wf. Conditions $cond_1, cond_2$, $\ldots$, $cond_m$ in the $WHERE$ clause (if any) specify how to insert these new elements in the process model wf.
  The INSERT DATA statement or the INSERT WHERE statement in the SPARQL query language fits for inserting new elements into the RDF file format. As an example, in Fig. 5, a new place, which contains a token and is connected to a transition, is inserted into a process model[7].

```
INSERT DATA{
 myWF:NameOfPlace a CPNOntology:Place;
   CPNOntology:hasMarking myWF:NameOfToken.
 myWF:NameOfWF CPNOntology:hasPlace myWF:NameOfPlace.}
```

Fig. 5: An example of the INSERT DATA statement

- Deleting existing elements from a process model.
  $DELETE\ ELEMENT\ \{e_1, e_2, \ldots, e_n\}\ FROM\ PROCESS\ wf;\ (n \geq 1)$
  This statement means that existing elements $e_1, e_2, \ldots, e_n$ are completely deleted from a process model named wf.
  The DELETE DATA statement or the DELETE WHERE statement in the SPARQL query language fits for deleting existing elements from the RDF file format.
- Updating existing elements for adapting to a process model.
  $UPDATE\ ELEMENT\ \{e_1, e_2, \ldots, e_n\}\ ON\ PROCESS\ wf$

[7] Two prefixes are assumed as:
$PREFIX\ CPNOntology: <http://myOnt.tutorial.org/2013\#>$
$PREFIX\ myWF: <http://myOnt.tutorial.org/2013-instances\#>$

$[WHERE\ cond_1, cond_2, \ldots, cond_m];\ (n \geq 1, m \geq 1)$

This statement means that elements $e_1, e_2, \ldots, e_n$ in a process model named wf, each of which has been created, are updated. Conditions $cond_1, cond_2, \ldots, cond_m$ in the $WHERE$ clause (if any) specify how to update these elements in the process model wf.

In this case, some statements in the SPARQL query language can be used, such as the INSERT DATA statement, the INSERT WHERE statement or the DELETE INSERT WHERE statement.

- Editing the order of existing elements in a process model.

  $MODIFY\ PROCESS\ wf$

  $WHERE\ cond_1, cond_2, \ldots, cond_n$

  $REPLACE\ condR_1, condR_2, \ldots, condR_m;\ (n \geq 1, m \geq 1)$

  This statement is used to edit ordering relationships in a process model. No element inserted, deleted or updated in the model.

  The DELETE INSERT DATA statement is used to edit the order of existing elements in the RDF file format.

5 Related work

As of today, many workflow modelling languages have been proposed, some of which have become widely accepted, used and replacing others. Most of them are based upon textual programming languages or graphical notations. For example, the Business Process Execution Language (BPEL) [9] is a standard way of orchestrating Web service execution in a business domain; the Business Process Model and Notation (BPMN) [10] is the de-facto standard for business process modelling; the Yet Another Workflow Language (YAWL) [11] is a domain specific language based on rigorous analysis of workflow patterns [12]; Petri Nets [13] and its extensions are well applied process modelling and analysis, etc. However, they use different concepts and different terminologies. This makes them difficult to inter-operate, share or reuse different workflow systems. As a result, it is a need for the semantics of concepts and their relationships.

We know that the ontology-based approach for modelling business process is not a new idea. There are some works made efforts to build business workflow ontologies, such as [14], [15], [16], [17], [18], [19], [20], [21], etc., to support (semi-)automatic system collaboration, provide machine-readable definitions of concepts and interpretable format. In [17], the authors defined an ontology for Petri Nets based business process description. Their ontology aims to facilitate the semantic interconnectivity of semantic business processes allowing information exchange. Our CPN ontology is very close to the one proposed by [17], however there are some differences. Our work focuses on representing process models in a meta-knowledge base, which is defined based upon the CPN ontology, in order to share and reuse them.

6 Conclusion

In this paper, we have introduced an ontological approach for modelling business processes. We defined the CPN ontology to represent CPNs with OWL DL. Firstly, each element of CPNs was translated into a corresponding OWL concept. Secondly, some of the axioms created for *Classes* and *Properties* in the CPN ontology were presented. The third OWL element, *Individuals* was also considered. As a result, the combination of CPNs and ontologies provides not only semantically rich business process definitions but also machine-processable ones.

In order to model business processes, four basic types of manipulation operations on the elements of process models are required. Therefore, we introduced the basic manipulation statements written in the pseudo code. The SPARQL statements, which correspond to those statements, were indicated to store CBWPs in the RDF file format. In spite of lacking the graphics, layout and GUI descriptions, the RDF files afterwards could be sent to other process-implementing systems to share and reuse them.

For validating CBWPs, we are planning to develop a run-time environment, which relies on the CORESE [22] semantic engine answering SPAQRL queries asked against an RDF knowledge base.

Acknowledgments. This research is conducted within the UCN@Sophia Labex.

References

1. OMG: Workflow management facility specification, v1.2. http://www.workflowpatterns.com/documentation/documents/00-05-02.pdf (2000)
2. Kristensen, L.M., Christensen, S., Jensen, K.: The practitioner's guide to coloured petri nets. STTT **2**(2) (1998) 98–132
3. Jørgensen, J.B., Lassen, K.B., van der Aalst, W.M.P.: From task descriptions via colored petri nets towards an implementation of a new electronic patient record workflow system. STTT **10**(1) (2008) 15–28
4. WFMC: Workflow management coalition terminology and glossary (wfmc-tc-1011), document number wfmc-tc-1011. Technical report (1999)
5. : Web ontology language. http://en.wikipedia.org/wiki/Web_Ontology_Language
6. W3C: Owl web ontology language reference. http://www.w3.org/TR/owl-ref/ (2004) W3C Recommendation.
7. Nguyen, T.H.H., Le-Thanh, N.: Representation of coloured workflow nets with owl dl ontoloty. In: Second International workshop "Rencontres scientifiques UNS-UD" (RUNSUD2013). (2013) 29–41
8. Nguyen, T.H.H., Le-Thanh, N.: Representation of rdf-oriented composition with owl dl ontology. In: Web Intelligence/IAT Workshops. (2013) 147–150
9. Andrews, T., Curbera, F., Dholakia, H., et al.: Business process execution language for web services version 1.1. http://msdn.microsoft.com/en-us/library/ee251594(v=bts.10).aspx (May 2003)

10. OMG: Business process model and notation, v2.0. http://www.bpmn.org/
11. : Yawl: Yet another workflow language. http://www.yawlfoundation.org/
12. : The workflow patterns home page. http://www.workflowpatterns.com/
13. : Petri net. http://en.wikipedia.org/wiki/Petri_net
14. Gasevic, D., Devedzic, V.: Reusing petri nets through the semantic web. In: ESWS. (2004) 284–298
15. Gasevic, D., Devedzic, V.: Interoperable petri net models via ontology. Int. J. Web Eng. Technol. **3**(4) (2007) 374–396
16. Hepp, M., Roman, D.: An ontology framework for semantic business process management. In: Wirtschaftsinformatik (1). (2007) 423–440
17. Koschmider, A., Oberweis, A.: Ontology based business process description. In: EMOI-INTEROP, Springer (2005) 321–333
18. Salimifard, K., Wright, M.: Petri net-based modelling of workflow systems: An overview. European Journal of Operational Research **134**(3) (2001) 664–676
19. Sebastian, A., Noy, N.F., Tudorache, T., Musen, M.A.: A generic ontology for collaborative ontology-development workflows. In: EKAW. (2008) 318–328
20. 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)
21. Zhang, F., Ma, Z.M., Ribaric, S.: Representation of petri net with owl dl ontology. In: FSKD. (2011) 1396–1400
22. Corby, O., et al.: Corese/kgram. https://wimmics.inria.fr/corese