# Coloured Petri Nets-based Approach for Manipulating RDF Data

Thi Hoa Hue Nguyen, Nhan Le Thanh

# Coloured Petri Nets-based Approach for Manipulating RDF Data

Thi-Hoa-Hue Nguyen and Nhan Le Thanh
WIMMICS - The I3S laboratory – CNRS, University of Nice Sophia Antipolis, Sophia Antipolis, France
Email: nguyenth@i3s.unice.fr, Nhan.LE-THANH@unice.fr

***Abstract*—This paper introduces a solution for controlling RDF data manipulation operations. We propose a formal approach to manage any modification, query or adaptation of the data to satisfy end-user/application criteria (e.g., RDF filtering). An overview of the RDF data manipulation framework with its main components is first presented. We then introduce the RDF-oriented Composition Definition Language (RDFCDL) including the syntax and its graphical representations defined based on CPNs. The language aims to support both expert and non-expert developers to create/compose RDF manipulation operations. Besides, an ontology for representing processes modelled with the RDFCDL language is developed in order to share and reuse the processes more easily. To the best of our knowledge, this is the first approach providing a means for end-users to create activity nodes over RDF data based upon system-defined functions and allowing manipulating processes to be stored in RDF file format.**

***Index Terms*—CPN, knowledge base, OWL DL ontology, RDF Data manipulation, RDF**

## I. Introduction

A knowledge base is an organized information repository. It provides a means for manipulating information, such as collecting, organizing, sharing, searching and using. According to [1], a knowledge base can be either machine-readable or intended for human use. Here we pay particular attention to the case of machine-readable knowledge bases designed for automated deductive reasoning. We consider ontologies expressed in RDF(S) or/and OWL that are a means of representing semantic knowledge.

The world is constantly full of changes, therefore it does generally not fit into a fixed, predetermined logic system. To explain this, especially to deal with the uncertainty inherent in the physical world, it is necessary to have different models of human reasoning. We focus on taking account domain experts in support of end-users to reason and make a decision in order to solve their problem.

On one hand, Coloured Petri Nets (CPNs) [2] have been developed to be a full-fledged language for design, specification, simulation, validation and implementation of large software systems. Moreover, CPNs are a well-proven language that is suitable for modelling of workflows or work processes [3]. However, the lack of semantic representation of CPN components can make business processes difficult to interoperate, share and reuse.

On the other hand, an ontology can provide machine-readable definitions of concepts, and therefore can represent semantically rich business process definitions. Consequently, Knowledge Engineering, Coloured Petri Nets and Semantic Web technologies can play an important role in this scenario to provide models and techniques to simplify control RDF data manipulation operations. We propose a formal framework which allows both novice and expert developers to process any modification, query or adaptation of the data to satisfy end-user/application criteria (e.g., RDF filtering).

In essence, existing techniques are applied to solve the problems related to each manipulation operation on a per-case basis or usually involving a high level of expertise in their respective fields. Our approach allows end-users to handle any RDF data-oriented operation in a simple and expressive way and thus overcomes the limitations mentioned above.

In this paper, we introduce the RDF data manipulation framework and particularly describe activity nodes. To achieve that, the rest of this paper is structured as follows: In Section 2, we introduce a few motivating scenarios. In Section 3, related work is discussed. We then present our approach in Section 4. Finally, Section 5 shows conclusions and ongoing works.

## II. Motivating Scenarios

Consider a construction project involving different stakeholders (e.g., experts in the construction industry, external regulatory bodies; company management; customers, etc.). Different manipulation/control scenarios are expected for distinct purposes of each stakeholder.

### A. Information Gathering

A customer would like to know the details of a ceiling light installed.

In order to achieve this, one technique would be required:

- **RDF data Filtering**: Filter RDF data based on concepts (e.g., $GraysonWoodPanel\_15$)

Manuscript received November 23, 2013; revised Januray 30, 2014.

### B. Related Information Gathering

An expert working in the company management is checking the lighting system in a construction project. The expert wishes to acquire all related technical information of the existing lighting components. Initially, high-level information (name the type of component, for example, $LedBulb$) is asked for and then the search engine has to seek all information involved in each component.

In order to achieve this, several techniques would be required:

- **RDF data Filtering**: Filter RDF data based on concepts.
- **RDF Structural Search**: Check RDF data if the component contains one or more elements based on the definition of giving concepts. All these elements (if any) matching to the nomenclature of the component are browsed. A result that contains all matching elements will be returned.

### C. Content Modification

The project is modified and some new components are added in. The components may be relevant to different domains. Domain experts wish to analyze and modify the structure and content of knowledge bases. Consequently, they can ensure the correctness of modified knowledge bases.

In order to achieve this, several techniques would be required:

- **RDF Structural Search**: Search for definitions of a component in the ontological knowledge bases.
- **RDF Content Search**: Search for the semantic content of RDF data stored.
- **RDF Structural Modification**: Modify the structure of an ontological knowledge base and ensure that the new components are defined correctly.
- **RDF Content Addition**: Add exactly new information about the new components to the ontological knowledge bases.

### D. Validation by Matching between Regulations and Ontological Knowledge Bases

An officer who works for the local government is checking the installations of an emergency lighting system. It is expected to comply with the emergency lighting system criteria established. The officer wishes to test the system with all of their rules and receive semantic annotations of the conformity checking results.

In order to achieve this, several techniques would be required:

- **Query Execution**: Execute user queries to check the conformance of ontological knowledge bases to the specified regulations.
- **Automated RDF annotation generation**: Generate an RDF file reporting the conformity checking results.

### E. Conformity checking Processes Based on Know-How Practices

Domain experts contribute to the development of knowledge bases in general and ontologies in particular. They define and validate all the concepts and relations of the knowledge bases, which are independent to conformity checking processes. Domain experts also formulate algorithms of effective checking. However, in some cases, their algorithms are not sufficient to cover the whole complexity of the real usage-driven conformity checking knowledge [4]. For example, in order to check the technical information of an entrance, the ground floor will be considered. This is because the entrance is normally defined as a door situated on the ground floor. But in fact, buildings located in the mountainous area in France, there are two entrance levels on the ground floor and on the second floor.

Therefore, it is necessary to take into account know-how practices for checking the conformance.

In order to achieve this, several techniques would be required:

- **RDF data Filtering**: Filter RDF data based on concepts.
- **RDF Structural and Content Search**: Search for definitions of a component and for the semantic content of RDF data stored.
- **Query Execution**: Execute user queries to check the conformance of ontological knowledge bases to the specified regulations.
- **Automated RDF file format generation**: Generate an RDF file reporting the conformity checking results.

Consequently, it is necessary to implement various techniques (i.e., content search, filtering, asking, etc.) together in order to provide a means for both non-expert and expert end-users to solve the mentioned requirements over RDF data. As far as we know, no framework has been created referring these matters simultaneously.

## III. Related Work

### A. Coloured Petri Nets and Semantic Business Process

CPNs are extended from Petri nets [5] with colour, time and expressions attached to arcs and transitions. They have a very well-defined semantics and have a graphical representation. Their notation that is similar to existing workflow languages can describe any type of workflow system. In addition, CPNs provide hierarchical descriptions. They offer interactive simulations where the CPN diagram can present the results directly. Consequently, there are many benefits to using CPNs as a modelling language.

Up to now, the combination of Petri Nets/high-level Petri Nets and ontologies has been studied in some research works [6], [7], [8], [9] to support (semi-) automatic system collaboration, provide machine-readable definitions of concepts and interpretable format. Unfortunately, existing approaches/techniques do not focus on modelling RDF data manipulation operations. We here propose an approach to manipulate RDF data

based upon CPNs and represent RDF-oriented compositions with ontologies.

### *B. Validation of Ontological Knowledge Bases*

Historically, most researches on compliance modelling of functions of a system and on modelling of regulations and conformity constraints have been carried out in parallel [4]. There is a rich history in the field of regulatory (performance concept) modelling in the research literature and existing standards documents [10], [11], [12], etc.

In [10], the regulatory requirements written in natural language were reformulated in a controlled and formal language of SBVR [13] (Semantics of Business Vocabulary and Business Rules) and then SPARQL [14]. They structured their control process based on expert practices.

In [11], the REFNet application was created for the purpose of developing an information infrastructure to determine the applicability of regulations in several conditions, based on a question and answer interface. Their compliance assistance system is based on an XML framework representing regulations and associated metadata.

In addition, "performance rules" are defined by regulatory bodies, informational or cultural standards, it would be clearly useful to check the conformity of all existing and/or potential systems to these recommended "performance rules". To date, various efforts have been made to check the conformance of a product according to defined rules [4], [15].

In [4], by taking into account expert knowledge in construction, the authors developed an ontology-based conformity checking model that allows (semi-) automating the conformity-checking process. In order to implement and validate that model, they proposed the C3R (Conformity Checking in Construction: Reasoning) conceptual framework and developed the C3R prototype respectively. The prototype is implemented with their ontology and methodology using the Semantic Web methods and tools used by professionals of the construction domain. Nonetheless, there is still significant room for improvement, for instance, allowing end-users to create/compose conformity checking processes.

### *C. Data Flow Visual Programming Languages*

Data flow Visual Programming Language (DFVPL) is known as a programming language based upon the data flow computing model [16]. Since data flow languages are programming paradigms and share some features of functional languages, DFVPLs are also primarily based upon functional composition. DFVPLs have essentially developed with the aim of allowing non-expert users, mostly scientists, to manipulate scientific data by means of visual compositions. According to [17], in a DFVPL, the distinction between language and environment is not clear. In addition, the distinction between the requirements, design, testing and coding of DFVPL-based software is blurred. This blurring thus makes DFVPLs easier for rapid prototyping. To date, various DFVPLs have been developed, e.g., LabView [18], Taverna [19], etc., which allow end-users to create their manipulation operations by providing a well-defined visual syntax. However, to the best of our knowledge, there is no existing approach adopting DFVPLs in RDF data manipulations.

## IV. OUR PROPOSAL

Our approach, called RDF data manipulation (RDFDM) framework, aims at enabling end-users to control RDF data manipulation operations. Ontologies expressed in RDF/S and/or OWL, which represent business processes, are subject to the RDFDM approach. As mentioned previously, some of our objectives of the RDFDM approach can be resolved by the existing approaches/techniques but not all of them. The RDFDM framework supports users to develop, modify, query or adapt RDF data in a provably rational and consistent manner. It can include simple operations as well as complex ones, such as RDF filtering, change operations, conformity checking of RDF data, etc.

The general idea of the approach is to define the RDFDM framework permitting activity nodes to be described and deployed in order to represent Control flow-based Business Workflow Patterns (CBWPs) in a knowledge base. There are three main components in the framework (Fig. 1).

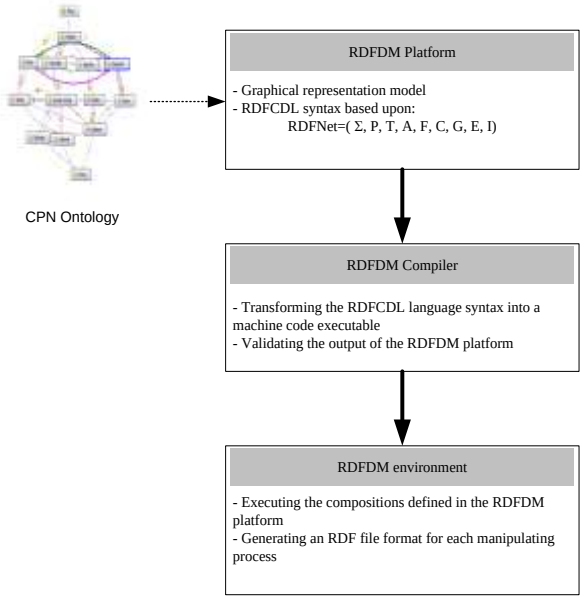


Figure 1. The components in the RDFDM framework

- The **RDFDM platform** is the most essential part and the main contribution of our work. We develop the RDFCDL (RDF-oriented Composition Definition Language) that allows end-users to create/compose their workflow patterns. A pattern represents a combination of procedures/functions used to perform a business process. Our platform relies on the graphical representations and allows textual descriptions. After defining, modifying or adapting a pattern, the output of the platform, which is an RDFDM document, is transmitted to the RDFDM compiler. Note that we build the CPN

Ontology for representing RDF data manipulation operations modelled with the RDFCDL language. This contributes to improve the semantic communication among process-implementing software components.

- The **RDFDM compiler** is placed between the platform and the environment. It is responsible for transforming an output of the platform into a machine code executable and validating that output.
- The **RDFDM environment** defines an environment for executing the resulting patterns created on the platform. In addition, we define a component to transform all patterns deployed in the environment into the RDF file format for the purpose of reusing and sharing the business processes.

In the following subsections, we introduce an overview of the RDFCDL and the CPN Ontology and our prototype.

### *A. RDF-Oriented Composition Definition Language Overview*

RDFCDL allows end-users to create/compose RDF-oriented manipulation operations using ANodes. The main goals of the RDFCDL are:

- serving end-users having or not having programming skills;
- allowing end-users to describe not only simple compositions but also complex ones;
- permitting end-users to define new operations based upon ANodes;
- being suitable for processing RDF data and enabling its integration to different systems (e.g., web or desktop application, etc.).

There are three main parts in the RDFCDL as follows:

- **The Inputs** are ontologies expressed in RDF/S and/or OWL. They are a representation of a project, for example a construction project.
- **The ANodes** and **the compositions** which compose the RDFCDL core. They are defined as CPNs.
- **The Outputs** are stored in RDF file format. They are RDF annotations of manipulating processes.

The syntax and semantics of the RDFCDL core are based upon the grammar RDFNet (RDF-oriented Composition Grammar Net) defined using CPNs. Consequently, the RDFCDL inherits the computations and operational semantics from CPNs, e.g., the firing rule.

**Definition 1** (RDFNet): RDFNet represents the grammar of the RDFCDL in compliance with CPNs. It is defined as a 9-tuple:

$$\text{RDFNet} = (\Sigma, P, T, A, F, C, G, E, I)$$

where:

- $\sum$ is a finite set of non-empty types available in the RDFCDL, called colour sets:

$$\Sigma = \{\text{Char}, \text{String}, \text{Integer}, \text{Double}, \text{Boolean}, \text{Date}, \text{RDFNode}\}$$

where $Char$, $String$, $Integer$, $Double$, $Boolean$, $Date$ are standard types and $RDFNode$ is a super-type (see Definition 2).

- $P = P_{in} \cup P_{out}$ is a finite set of places. $P_{in}$ and $P_{out}$ Denote the input and output states of the function used in RDFCDL respectively. The number of tokens in place p: $\forall p \in P: [w(p) = 1]$
- T is a finite set of transitions. The behavior of the functions and operators in RDFCDL is represented by transitions.
- $A \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs connecting input places to transitions or transitions to output places. $\forall a \in A: a.p$ and $a.t$ stand for the place and transition linked by a, respectively.
- F is a set of operators/functions available in the libraries.
- $C: P \rightarrow \Sigma$ is a colour function. Each place has only one type from Σ: $\forall p \in P: [|C(p)| = 1]$.
- $G: T \rightarrow F$ is a guard function associating an operation to a transition.
- $E: A \rightarrow Expr$ is an arc expression function. It is defined from A into Expr such that:

$$\forall a \in A: [\text{Type}(E(a)) = C(a.p) \wedge \text{Type}(\text{Var}(E(a)))]$$

- $I: P \rightarrow Init$ is an initialization function associating initial values to places. It is defined from *P* into *Init* such that: $\forall p \in P: [Type(I(p)) = C(p)]$.

Since *F* is added to the initial CPN definition and has no effect on the CPN's functionality, in the rest of the definitions based upon CPNs it is bypassed.

We introduce the following definition of $RDFNode$ that designates an RDF (Resource Description Framework) [20] component.

**Definition 2 (**RDFNode): $RDFNode$ contains three sub-types that are *RDFNode:URI*, *RDFNode:Literal* and *RDFNode:Blank*, where:

- *RDFNode:URI* defines the RDF URI reference type.
- *RDFNode:Literal* defines the RDF literal type.
- *RDFNode:Blank* defines the RDF blank node type.

The ANodes and the compositions, which follow the grammar RDFNet, are defined based upon CPNs. Consequently, the inputs and outputs of ANodes are defined as places and drawn as ellipses. Note that in this study, a function can have an input and an output. Each place has a single colour defining its type. A transition, which is drawn as a rectangle, represents the operation of the function. It operates on the input and sends the result to the output. The input and output places are linked to transitions via directed arcs drawn as arrows. A directed arc connects a place with a transition or vice versa. Several sample functions are shown in Fig. 2.

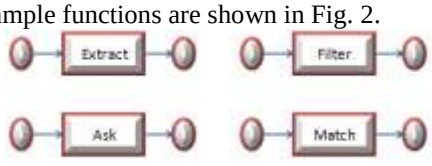


Figure 2. Some sample functions defined in RDFCDL

**Definition 3** (Activity node)**:** ANode is a function defined based upon CPNs. It describes an operation and is defined as:

$$\text{ANode} = (\Sigma, \text{P}, \text{T}, \text{A}, \text{C}, \text{G}, \text{E}, \text{I})$$

where:

- $\Sigma$ is a finite set of non-empty types available in the ANode, where: $\Sigma \subseteq RDFNet.\Sigma$.
- $\text{P} = \text{P}_{\text{in}} \cup \text{P}_{\text{out}}$ is a finite set of places defining the input and output states of the ANode.
- $P_{in}$ and $P_{out}$ are the set of input and output places respectively where:
$$P = P_{in} \cup P_{out}; P_{in} \cap P_{out} = \emptyset;$$
$$P_{in} = \{p_{in}\}; P_{out} = \{p_{out}\}$$
- $\text{T} = \{\text{t}\}$ is a finite set of transitions denoting the behavior of the ANode. Transition t contains the operation to be performed.
- $\text{A} \subseteq (\text{P} \times \{\text{t}\}) \cup (\{\text{t}\} \times \text{P})$ is a set of directed arcs connecting input places to transition $t$ or transition $t$ to output places.
- $\text{C}: \text{P} \to \Sigma$ is a colour function associating a type to each place. It is defined from $P$ to $\Sigma$.
- $\text{G}: \{\text{t}\} \to \text{F}$ is a guard function associating an operation to transition $t$. It is defined from $T = \{t\}$ into F where:
$$Type(G(t)) = Type(Var(G(t))) \wedge C(p_{out}) \subseteq \Sigma$$
- $\text{E}: \text{A} \to \text{Expr}$ is an arc expression function where $Expr$ is a set of expressions. It is defined from $A$ into $Expr$ where:
$$\forall \text{a} \in \text{A}: \text{E(a)} = \begin{Bmatrix} \text{M(a. p) if a. p} \in \text{P}_{\text{in}} \\ \text{G(a. t) otherwise} \end{Bmatrix}$$
$M(a.p)$ is the value of the token in $p$.
- $\text{I}: \text{P}_{\text{in}} \to \text{Init}$ is an initialization function associating initial values to input places.

A composition is defined by a mapping between the outputs and the inputs of ANodes. It is expressed by a combination of graphical functions via operators. A suitable operator is used for one link between some functions related to each other. Because a composition might be sequential, parallel or conditional, we use the operators including Sequence, And-split, And-join, Xor-split and Xor-join (Fig. 3) to create the compositions. These operators are defined based upon CPNs, which are compliant to RDFNet.

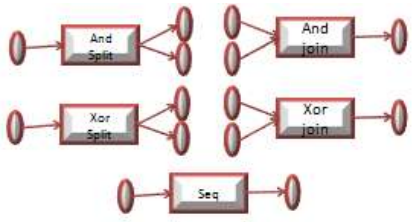


Figure 3. Operators are defined in RDFCDL

### B. CPN Ontology

We now define semantic metadata for processes of controlling RDF data-oriented operations modelled with the RDFCDL language. This facilitates it particularly easy to (semi-) automate the interoperability, share and reuse among process-implementing software components. We continue our work at [21], [22] to translate key components of the RDFCDL language into classes, properties and axioms of OWL DL ontologies.

**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), provides maximum expressiveness while always keeping computational completeness and decidability. Therefore, we choose the OWL DL language to represent RDFNet. For more details on OWL DL, please refer to [23].

Since RDFNet presents the grammar of the RDFCDL based upon CPNs, the RDFNet structure thus consists places, transitions and directed arcs. We start to construct their corresponding classes, for example, a place is translated into a class. The core concepts of our CPN Ontology are depicted in Fig. 4. The CPN Ontology is very close to the one proposed by A. Koschmider and A. Oberweis [8]. However there are some differences.

In order to help different systems interacting with the CPN Ontology to understand what is represented by each element in the ontology, in the next step we describe the meaning of its main elements.

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 in places; **GuardFunction** defined for all transition expressions; **CtrlNode** defined for occurrence condition in operators; **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 $I_1 \times I_2 \times ... \times I_n$ where $I_i$ is a set of individuals.

Properties between the concepts in the CPN Ontology are also shown in Fig. 4. We define two properties *hasMarking* and *connectsTrans*, for example, for the concept *Place* as illustrated in Fig. 5.

### C. Prototype: The RDFCD Platform

For the purpose of validating our approach, we have implemented a Java based prototype named RDFDM. A screen shot of this prototype is shown in Fig. 6. We set up the prototype with the RDFNet defined based upon CPN mentioned previously. We now illustrate how it works by considering the scenario D presented in the Motivating scenarios Section.

$$CPNOnt \equiv \geq 1 hasTrans.Transition \sqcap \geq 1 hasPlace.Place \sqcap \geq 1 hasArc.(InputArc \sqcup OutputArc)$$
$$Place \equiv connectsTrans.Transition \sqcap = 1 hasMarking.Token$$
$$Transition \equiv connectsPlace.Place \sqcap 1 hasGuardFunction.GuardFunction$$
$$InputArc \equiv \geq 1 hasExpresion.Delete \sqcap \exists hasPlace.Place$$
$$OutputArc \equiv \geq 1 hasExpresion.Insert$$

$$\sqcap \exists\, hasTrans.Transition$$
$$Delete \equiv \forall\, hasAttribute.Attribute$$
$$Insert \equiv \exists\, hasAttribute.Attribute$$
$$GuardFunction \equiv \geq 1 hasAttribute.Attribute \sqcap = 1 hasActivity.ActNode \sqcup = 1 hasControl.CtrlNode$$
$$Token \equiv \geq 1 hasAttribute.Attribute$$
$$Attribute \equiv \geq 1 valueAtt.Value$$
$$ActNode \equiv = 1 valueAtt.Value$$
$$CtrlNode \equiv \leq 1 valueAtt.Value$$
$$Value \equiv valueRef.Value$$

Figure 4. CPN ontology.

```
< owl:ObjectProperty rdf:about =
"#hasMarking" >
  < rdfs:domain rdf:resource = "#Place"/>
 < rdfs:range rdf:resource = "#Token"/>
</owl:ObjectProperty >
< owl:ObjectProperty rdf:about =
#connectsTrans" >
  < rdfs:domain >
 < owl:Class >
  < owl:unionOf rdf:parseType =
"Collection" >
   < owl:Class rdf:about = "#InputArc"/>
   < owl:Class rdf:about = "#Place"/>
  </owl:unionOf >
 </owl:Class >
</rdfs:domain >
< rdfs:range >
 < owl:Class >
  < owl:unionOf rdf:parseType =
"Collection" >
   < owl:Class rdf:about =
"#OutputArc"/>
   < owl:Class rdf:about = "#Transition"/>
  </owl:unionOf >
 </owl:Class >
</rdfs:range >
</owl:ObjectProperty >
```

Figure 5. The definitions of two properties *hasMarking* and *connectsTrans*

In this scenario, an officer wants to check the conformance of the installations of an emergency lighting system in a construction project to the regulations defined. For example, the system has to ensure that the luminaires are located at mandatory "points of emphasis":

*"Specific locations where a luminaire must be provided are: (1) At each exit door; (2) All safety exit signs; (3) Outside and near each final exit; (4) Near stairs so that each tread receives direct light; (5) At each change of direction; (6) Near each first aid post; (7) Near any other change of floor level; (8) At each intersection of corridors; (9) Near each piece of fire fighting equipment and call point"* [24].

To create the corresponding composition, we first have to define the input and output content description structures. The inputs are expressed in ontologies representing the construction project. The output is generated as RDF annotations of the manipulating process. In the next subsection, we introduce our RDF file format.

As we can see, in Fig. 6, ANodes along with composition operators are provided graphically. In the case of checking the conformity of locating luminaires, we visually compose two required ANodes and one operator. This work has done without the aid of expert programmers.

Figure 6. An illustration of scenario D of RDFDM

### D. RDF File Format

After modelling the processes of RDF data manipulations with the RDFCDL language, the models are mapped to the CPN Ontology and exported automatically to OWL syntax.

```
<rdfnet:Place rdf:ID="Place_ID01">
 <rdfnet:hasMarking>
  <rdfnet:Token rdf:ID="Token_ID04">
   <rdfnet:hasValue>
    <rdfnet:Value rdf:ID="Value_ID05">
     <rdfnet:valueRef>
 PREFIX lsOnt: <http://www.semanticweb.org
 /myOntology/ont#>
 SELECT ?x WHERE
 {?x rdf:type lsOnt:EmergencyExit}
     </rdfnet:valueRef>
    </rdfnet:Value>
   </rdfnet:hasValue>
  </rdfnet:Token>
 </rdfnet:hasMarking>
 <rdfnet:connectsTrans
       rdf:resource="#Trans_ID03"/>
</rdfnet:Place>
```

Figure 7. Representing semantic process of RDF data manipulations in RDF syntax (excerpt)

In spite of lacking the graphics, layout and GUI descriptions, our file format afterwards could be sent to other RDF data-oriented process-implementing systems to reuse and share it. Fig. 7 depicts an excerpted RDF serialization of the scenario D.

## V. CONCLUSIONS

In this paper, we have introduced an overview of the RDFDM framework with three main components supporting users to control manipulation operations. We presented an overview of the RDFCDL language allowing end-users to create/compose RDF manipulation

operations. We defined RDFNet based on CPNs being the grammar of the RDFCDL.

We developed the CPN Ontology for representing CPNs with OWL, which aims to share and reuse the processes of manipulating RDF data not only in the Semantic Web, but also in workflow systems.

A prototype named RDFDM have been implemented to valid our approach. Our ongoing works focus on refining our prototype. To execute the components of the RDFCDL language, we are planning to develop a run-time environment, which relies on the CORESE [25] semantic engine that answers SPAQRL queries asked against an RDF/OWL knowledge base.

## ACKNOWLEDGMENT

This research is conducted within the UCN@Sophia Labex.

## REFERENCES

[1] Knowledge Base. (November 2013) [Online]. Available: http://comp.utm.my/lab/our-service/knowledge-base/page/2/

[2] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use* in EATCS monographs on Theoretical Computer Science, vol. 1, Berlin:Springer-Verlag, 1997.

[3] J. B. Jørgensen, K. B. Lassen, and W. M. P. van der Aalst, "From task descriptions via colored Petri nets towards an implementation of a new electronic patient record workflow system," *International Journal on Software Tools for Technology Transfer*, vol. 10, issue 1, pp. 15-28, January 2008.

[4] A. Yurchyshyna, "An ontology based approach for modelling the process of conformity checking in construction," Ph.D. Dissertation, I3S laboratory, University of Nice Sophia Antipolis, Sophia Antipolis, France, 2009.

[5] Petri net. (November 2013). [Online]. Available: http://en.wikipedia.org/wiki/Petri_net

[6] D. V. Gašević and V. B. Devedžić, "Reusing Petri Nets Through the Semantic Web," in *The Semantic Web: Research and Applications*, Christoph Bussler, John Davies, Dieter Fensel *et al.* , Eds. Berlin:Springer-Verlag, 2004, pp. 284-298.

[7] D. V. Gašević and V. B. Devedžić, "Interoperable Petri net models via ontology," *International Journal of Web Engineering and Technology*, vol. 3, no. 4, pp. 374-396, 2007.

[8] A. Koschmider and A. Oberweis, "Ontology based business process description," in *Proc. of the CAiSE´05 WORKSHOPS*, 2005, pp. 321-333.

[9] F. Zhang, Z. M. Ma, and S. Ribaric, "Representation of Petri net with OWL DL ontology," in *Proc. Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2011, pp. 1396-1400 .

[10] K. R. Bouzidi, B. Fies, C. Faron-Zucker, A. Zarli, and N. Le-Thanh, "Semantic web approach to ease regulation compliance checking in construction industry," *Future Internet*, vol. 4, no. 3, pp. 830-851, 2012.

[11] S. Kerrigan and K. H. Law, "Logic-based regulation compliance-assistance," in *Pro. of 9th International Conference on Artificial Intelligence and Law*, Edinburgh, Scotland, 2003, pp. 126-135.

[12] A. Nazarenko, A. Guisse, F. Levy, N. Omrane, and S. Szulman, "Integrating written policies in business rule management systems," *Rule-Based Reasoning, Programming, and Applications Lecture Notes in Computer Science* , vol. 6826, pp. 99-113, 2011.

[13] SBVR (Semantics of Business Vocabulary and Business Rules). (November 2013). [Online]. Available: http://www.w3.org/TR/rdf-sparql-query

[14] SPARQL 1.1 Query Language.(November 2013).) [Online]. Available: http://www.w3.org/TR/sparql11-query/

[15] C. Eastman, J. M. Lee, Y. S. Jeong, and J. K. Lee, "Automatic rule-based checking of building designs," *Automation in Construction*, vol. 18, issue 8, pp. 1011-1033, 2009.

[16] S. Abe, "Plumber - A higher order data flow visual programming language in Lisp," presented at the International LISP conference 2012, Kyoto, Japan, October 21-24, 2012.

[17] W. M. Johnston, *et al.*, "Advances in dataflow programming languages," *ACM Comput. Surv.*, vol. 36, pp. 1-34, 2004.

[18] LabVIEW (Laboratory Virtual Instrument Engineering Workbench. (October 2013) [Online]. Available: http://www.ni.com/labview/f/

[19] Taverna Workflow Management System. (October 2013). [Online]. Available: http://www.taverna.org.uk/

[20] RDF (Resource Description Framework: Concepts and Abstract Syntax). (October 2013). [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[21] T. H. H. Nguyen and N. Le-Thanh, "Representation of coloured workflow nets with OWL DL ontology," in *Proc. Second International workshop ``Rencontres scientifiques UNS-UD'' (RUNSUD'2013)*, Vietnam, 2013, pp. 29-41.

[22] T. H. H. Nguyen and N. Le-Thanh, "Representation of RDF-Oriented Composition with OWL DL Ontology," *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Atlanta, USA, 2013, vol. 3, pp.147-150.

[23] OWL Web Ontology Language Overview. (November 2013). [Online]. Available: http://www.w3.org/TR/owl-features/

[24] Emergency Lighting Design Guide. [Online]. Available: http://www.iar.unicamp.br/lab/luz/ld/Seguran%E7a/emergency_lighting_design_guide.pdf (accessed on 18 November 2013)

[25] Corese/KGRAM. (November 2013) [Online]. Available: http://wimmics.inria.fr/coresewakka.php?wiki=Corese

**Thi-Hoa-Hue Nguyen** was born in 1982 in Bac Giang, Vietnam. She received the Bachelor in Applied Mathematics and Informatics from the College of Science, Vietnam National University, Hanoi in 2003 and the Master of Engineering in Computer Science from the University of Danang in 2008. She is currently completing her PhD thesis in the WIMMICS team, I3S laboratory, UMR 7271 CNRS-INRIA, France. Her work and research interests fall mainly the areas of RDF data manipulation/control, machine-readable knowledge bases, business process modelled with Coloured Petri Nets and visual languages.
Ms. Nguyen holds a teaching position in the Computer Science Faculty, Vietnam-Korea Friendship Information Technology College where she mainly teaches Object-Oriented Database and Semantic Web (e.g., RDF/S, SPARQL) on courses.

**Nhan Le-Thanh** was born in 1952 in Thanh Hoa, Vietnam. He received the Bachelor degree in Mathematics from the Hanoi University, Vietnam in 1973, the Engineer degree in Mathematics and Computer Science from the National Polytechnic Institute of Toulouse, France in 1982, and the Ph.D. degree in computer science from the University of Nice Sophia Antipolis (UNS), France in 1986. His research interests are in the problems of semantic transformation in distributed systems, in particular the following three issues: the integration of heterogeneous data sources in the Semantic Web, the decomposition of an ontology described in Description Logic (DL) in a distributed system of ontologies described on distributed description logic (DDL) and checking the consistency and organization of exchange in a distributed scalable federation system type.
Since September 1987, Prof. Le-Thanh has worked in the Department of Computer Sciences, UNS where he became an Associate Professor in 1988 and a Professor in 1992. He is a permanent member of the WIMMICS team, I3S laboratory, UMR 7271 CNRS-INRIA.