Expert Systems with Applications 37 (2010) 6493–6509

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# OPENET: Ontology-based engine for high-level Petri nets

Juan C. Vidal, Manuel Lama [*], Alberto Bugarín

Dept. Electronics and Computer Science, University of Santiago de Compostela, 15782 Santiago de Compostela, Spain

ARTICLE INFO

*Keywords:*
Ontologies
High level Petri nets
High level Petri nets execution engine

ABSTRACT

In this paper, OPENET, an engine for the execution of high-level Petri nets (HLPNs) is presented. OPENET is based on an ontology that represents the knowledge of the ISO/IEC 15909-1 standard and, respectively, describes semantically and declaratively both the static structure and the dynamic behavior of HLPNs. Thus the ontology is composed of (i) a taxonomy that describes the main components of a net, capturing the vocabulary and semantics specified in the standard; and (ii) a set of axioms and rules that constrain how the instances of the taxonomy are created, restricting the range and domain of the relations, and the values of the attributes. These axioms guarantee that a HLPN is correctly constructed, and restrict how it should be executed; and (iii) a set of rules which contain the knowledge needed to execute HLPNs and thus infer new instances of the concepts that describe the dynamic model of the HLPN. The OPENET engine has been implemented in F-Logic with the FLORA-2 reasoner, and is being used in several domains: the execution of courses in E-learning, the modeling and execution of workflows in Industry, and the execution of web service choreographies.

## 1. Introduction

Market demands organizations to find solutions in order to increment their productivity and to reduce their costs. A way to achieve these objectives is the improvement of their business processes, and *workflows* (van der Aalst & van Hee, 2004) are increasingly being used to model such processes because they facilitate the communication, coordination and collaboration between the participants of the business process. Several formal techniques have been proposed to model both the static and dynamic behavior of workflows (Borger & Stark, 2003; Milner, 1999), but Petri nets (Jensen, 2003; van der Aalst, 1998) is the approach adopted for many of workflow management software vendors, because of its simple graphical representation and the soundness of its mathematical model (van der Aalst & van Hee, 2004).

In the last decades a number of Petri net tools and engines have been developed (Petri Net World, 2008). Note that we consider high level Petri nets (HLPNs) engines to the tools that moreover than designing HLPNs also support their execution and integration in another software modules. Therefore, these engines can be used for rapid prototyping and can be considered as a process/workflow engine. Much of these tools are oriented to design and simulate Petri nets (Baarir, Beccuti, & Franceschinis, 2008; Evangelista, 2005; Jensen, Kristensen, & Wells, 2007; Sklenar, 2002). However, they do not have an open interface and thus cannot be integrated as a part of an application. Therefore, the Petri nets are directly implemented in a common purpose programming language. This drawback is overcome by other engines that define a programming interface that facilitates the execution of Petri nets as a software module of an application (Basile, Carbone, & Chiacchio, 2002; Ceska, Janousek, & Vojnar, 2002; Kindler & Weber, 2001; Kummer et al., 2004; Pommereau, 2008; Zimmermann, Knoke, Huck, & Hommel, 2006). However, these Petri net engines usually present the following limitations:

- The extension and/or modification of the HLPN *execution semantics* requires the recoding of the engine. These engines implement the Petri net semantics in programming languages like Java, C++ or Phyton and therefore need to be recompiled whenever the semantics is modified. At the most, some of the engines support the management of new operators coded in that language (Basile et al., 2002; Ceska et al., 2002; Kummer et al., 2004; Pommereau, 2008; Zimmermann et al., 2006) or in a declarative language (Evangelista, 2005).
- The reuse of a given Petri net in different domains is difficult, since these engines define neither an explicit separation between algebras and signatures nor a mapping mechanism between them. That is, annotations of Petri nets (specified in signatures) are described with the concepts of a given domain (specified in algebras). Therefore, the adaptation of a Petri net between different domains must be done manually.
- The sharing of the execution of the Petri nets among agents is limited. These tools enable external agents to access to the intermediate results of the Petri net execution (steps, transition

* Corresponding author.
*E-mail addresses:* juan.vidal@usc.es (J.C. Vidal), manuel.lama@usc.es (M. Lama), alberto.bugarin.diz@usc.es (A. Bugarín).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.02.136

modes, firings, and so on) through its programming interface (Basile et al., 2002; Kindler & Weber, 2001; Kummer et al., 2004; Pommereau, 2008; Zimmermann et al., 2006). However, they do not offer these intermediate results through a declarative format, and thus the sharing among agents coded in different programming languages is not possible.

In order to solve these drawbacks ontologies are a good choice. Ontologies arise from the fact that a common vocabulary is not enough. In this sense, an XML format is only a step forward in ensuring that computers can communicate freely, but does not provide any semantics. As a formal and explicit specification of a shared conceptualization (Gómez-Pérez, Fernández-López, & Corcho, 2004), ontologies capture the shared knowledge of a given domain and represent this knowledge in a declarative and expressive language (*explicit*) that follows a logical paradigm (*formal*). Thus an ontology looks for the description of the concepts and relations between those concepts, but also for the axioms that constrain their meaning, and the rules that enable to infer new instances. Therefore, an engine based on an ontology that captures both the static and dynamic knowledge of HLPNs could deal with the drawbacks of current engines: on one hand, if new extensions and/or modifications to Petri nets are required, they could be declaratively codified in the ontology through new concepts, axioms or rules. In this case, no recoding in a programming language is necessary. On the other hand, the ontology is based on the Petri net standard. Since this standard explicitly separates signatures and algebras, ontologies may be used to map the net annotations with the domain concepts.

In this paper, an ontology of HLPNs that captures the knowledge of the ISO/IEC 15909-1 (ISO/IEC, 2002) standard is presented. The ontology represents semantically and declaratively the static structure and the dynamic behavior of HLPNs, and is composed of (i) a taxonomy that describes the main components of a net, capturing the vocabulary and semantics specified in the standard; (ii) a set of axioms that constrain how the instances of the taxonomy are created, restricting the range and domain of the relations, and the values of the attributes. These axioms ensure that a HLPN is correctly constructed, and restricts how it should be executed; and (iii) a set of rules that enable both the management of the HLPN static model and its execution. Based on this ontology the OPENET HLPN engine has been developed using the FLORA-2 (Yang, Kifer, & Zhao, 2003) environment. This engine has an *open and extensible* architecture with a set of modules that allow the definition, validation, and execution of HLPNs in domains modeled through ontologies expressed in FLORA-2 (dialect of F-Logic (Kifer, Lausen, & Wu, 1995)). This engine has been used for the execution of units of learning in Education, for process management in Industry, and for the orchestration of semantic web services (Payne & Lassila, 2004).

The paper is structured as follows: Section 2 describes the HLPNs ontology. Based on this ontology, Section 3 details the architecture of the HLPNs engine. Section 4 presents the main advantages of our approach and compares the ontology described in this paper with other proposals. Finally, Section 5 evaluates the ontology and the system performance, and Section 6 presents the conclusions of this work.

## 2. High-level Petri nets ontology

The ontology describes HLPNs as a piece of knowledge that may be reused and shared among the Petri nets community. It specifies the concepts and relations that characterize this class of nets and represents them through a logic formalism so they are machine processable. The ontology is based on the shared knowledge defined in the standard ISO/IEC 15909-1 (ISO/IEC, 2002) and translates its mathematical specification into a *taxonomy*, a set of *axioms* and *rules* which formally constrain the semantics of that taxonomy. The ontology axiomatizes all the aspects of HLPN definition and execution models and defines its underlying execution rules. Note that some of the axioms, such as those that check the data integrity constraints, and some of the rules have not been included in this paper for the sake of brevity.

This ontology is not limited to the representation of the graph structure of Petri nets (the elements that represent invariants) (Breton & Bézivin, 2001). As the purpose of this work is the creation of an ontology-based HLPNs engine, this ontology also describes the dynamic model of HLPNs, and thus makes its underlying execution rules explicit. In this sense, we must emphasize that this ontology supports the entire ISO/IEC 15909-1 standard. Even features not usually supported in most of HLPN models have been included (for instance, the separation between signatures and algebras).

Multisets, which are very representative of this type of nets, are also captured in this ontology. The standard ISO/IEC 15909-1 refers to a multiset of terms that may annotate an arc, a multiset of tokens that may be located in a place, and a multiset of modes that may occur in a firing. However, ontology languages do not have a direct support for multisets representation. In this work we did not define a concept for modeling multisets but we used lists for the definition of multisets/bags. For example, the $1'3 + 2'4$ multiset is represented by the $[3,4,4]$ list.

The HLPNs ontology has been implemented in the FLORA-2 language, and a Java bridge to FLORA-2 is available to facilitate its use in external applications.[1] To illustrate this ontology, the taxonomy is represented graphically through semantic networks, axioms are described in first-order logic and rules in the FLORA-2 language that implements the OPENET reasoner. Firstly, a semantic network is a formalism typically used for representing ontologies since it facilitates the graphical view of its concepts (classes) and roles (relations). However, this representation has the issue that the cardinality of relations cannot be displayed. The cardinality of these relations has not been included in this paper since we considered (i) that the role names are sufficiently expressive and (ii) in interest of brevity, since cardinality is specified through axioms and so two axioms are needed to specify the minimum and maximum cardinality of the relation, respectively. Secondly, axioms are described in first-order logic to facilitate understanding. Thus, concepts and relations are described though unary and binary predicates: the name of each predicate represents the class or the relation, respectively. The variables of the sentences are universally or existentially quantified. For example, $sourceNode(A,N)$ represents the *sourceNode* relation between the arc $A$ and the node $N$. Finally, rules are described in the FLORA-2 language to show how the rules are implemented in the OPENET engine.

### 2.1. Astronomy case study

To illustrate the description of the HLPN ontology a real-world example is used. The example is a case study of astronomy that belongs to the field of e-learning and has been developed to test a units of learning engine (Sánchez, Lama, Amorim, Vidal, & Novegil, 2008) built on the top of OPENET. This case is modeled as a unit of learning composed of several activities that students must perform to classify the planets according to their distance to the sun. To achieve this objective the unit of learning has been designed following a collaborative learning approach: the teacher proposes a game to two teams of students (*Team A* and *Team B*) where each team has a *fraction* of the knowledge needed to solve the problem,

[1] Both developments can be downloaded from the OPENET website at http://www.gsi.dec.usc.es/hlpno/downloads.html.