The Knowledge Engineering Review, Vol. 31:3, 239–260.
doi:10.1017/S0269888916000072

A literature overview of knowledge sharing between Petri nets and ontologies

HAITAO CHENG[1] and ZONGMIN MA[1,2]

[1]*College of Computer Science & Engineering, Northeastern University, Shenyang 110819, China*
e-mail: haitaoneu@126.com;
[2]*College of Computer Science & Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China;*
e-mail: zongmin_ma@yahoo.com

Abstract

Knowledge representation is a subarea of artificial intelligence concerned with using formal symbols to represent a set of facts within a knowledge domain. Two popular knowledge representation languages, namely Petri net and ontology, are promising knowledge sharing and reusing methods in knowledge engineering. The combination of Petri net and ontology can facilitate achieving complementary advantages. Currently, many efforts have been done on knowledge sharing between Petri nets and ontologies. To investigate these issues and more importantly serve as identifying the direction of knowledge sharing between Petri nets and ontologies, in this paper we give a comprehensive literature overview of knowledge sharing between Petri net models and ontology models to satisfy the obvious need. In detail, we discuss the knowledge sharing from two aspects: the different knowledge representation approaches of ontology to represent and reason Petri net and issues of constructing Petri net from ontology. In addition, other important issues on applications and directions for future research are discussed in detail.

1 Introduction

Knowledge representation is a subarea of artificial intelligence (AI) concerned with using formal symbols to represent a set of facts within a knowledge domain so that AI systems can facilitate inferences from the existing knowledge to produce representation of new knowledge (Davis *et al*., 1993; Brachman & Levesque, 2004; Jozefowska, 2010). Currently, there are a wide variety of approaches to knowledge representation, such as frames (Lam & Srihari, 1991; Ranasinghe & Madurapperuma, 2003), rules (O'Leary, 2007; Kowalski & Burton, 2011), Petri nets (Deng & Chang, 1990; Lovrek, 1995; Yu *et al*., 1998; Ribaric & Hrkac, 2007; Jiang & Li, 2009; Ribaric & Hrkac, 2012), ontologies (Li & Wang, 2009; Zhang *et al*., 2010; Wang & Wu, 2012) and problem-solving methods (Puppe, 1998; Perez & Benjamins, 1999). However, there is no single knowledge representation that is best for all knowledge-based systems (Neches *et al*., 1991). Therefore, the possibility of sharing knowledge from different knowledge-based systems that different agents have, seems to be very promising.

So far, Petri nets and ontologies have been developed. Petri nets play an important role in Knowledge engineering. Petri nets cannot only represent knowledge, but also reason using knowledge. As a formal system model, Petri nets have a strong capability to model events and states in a distributed system. Besides, it is easy to capture sequential, concurrency and event-based control (Murata, 1989). In addition, Petri nets can describe static and dynamic characteristics, and they have a set of mathematical analysis methods to deal with the problems of logical reasoning. On the other hand, Ontologies are a formal and explicit specification of a shared conceptualization model (Studer *et al*., 1998), which can represent and reason with knowledge in a formal and explicit manner. Ontologies, as a formal knowledge representation method describing a conceptualization of some domain, can capture both the structure and semantics of an information environment.

Knowledge sharing is one of the key issues in knowledge management, which can utilize existing knowledge bases to construct a large and powerful knowledge-based system. The Advanced Research Projects Agency (ARPA) Knowledge Sharing Effort proposed a new way in which intelligent systems could be built by assembling reusable components (Neches *et al*., 1991). In this way, systems developers would not need to construct new knowledge bases from scratch, but they would be able to share and reuse the general knowledge and to concentrate on the specialized knowledge of their systems. Also, researchers define knowledge sharing as equivalent to knowledge transfer and sharing among members of organization that are analogous to knowledge systems (Behrang *et al*., 2010). Thus, in some cases, sharing knowledge involves translating from one representation approach to another. So far, many researchers have proposed a lot of techniques of knowledge sharing by combining two knowledge representation approaches, for instance, knowledge sharing between ontologies and problem-solving methods (Perez & Benjamins, 1999) facilitating building bigger and better systems and knowledge sharing between Petri nets and ontologies (Gasevic & Devedzic, 2007). In this paper, we focus on the integration of Petri nets and ontologies, which implements the functional interaction between them.

The aim of knowledge sharing between Petri nets and ontologies is to achieve the complementary advantage. To implement the knowledge sharing, the interactions, involving translating from Petri nets to ontologies and from ontologies to Petri nets, must be considered. From the point of mapping Petri nets to ontologies, the use of ontologies is to semantically describe Petri nets concepts and their relationships, including three major abstraction forms: *mapping a Petri net into a metamodel*, *mapping a Petri net into an ontology* and *mapping a Petri net into an ontology description language*. For example, Gasevic and Devedzic (2007) proposed a Petri net ontology with the aim of enabling an effective sharing of Petri nets. In the literature (Vidal & Lama, 2009; Vidal *et al*., 2010), the authors described HLPN (high-level Petri net) ontologies, which represent semantically and declaratively the static structure and the dynamic behavior of HLPNs. In Feng *et al*. (2008), a Hierarchical Object-Oriented Petri net (HOOPN) modeling method based on an ontology was presented. In Brockmans *et al*. (2006), an approach to represent Petri nets with Web ontology language (OWL) was proposed in order to enable semantic information exchange between the two semantic business models. From the point of mapping ontologies to Petri nets, the use of Petri nets within a power capability of describing dynamic behavior and analyzing knowledge, is to model ontologies. Similarly, this problem is considered as consisting of two parts, the translation from ontologies to Petri nets and from ontology description languages to Petri nets. For instance, Recker and Indulska (2007) established the representation mapping from the BWW ontology (Bunge, 1977; Wang & Weber, 1993) to a Petri net model. To solve the problems of semantics recognition, a formal approach of modeling OWL for Web Service (OWL-S) choreography by means of HLPNs was discussed by Vidal *et al*. (2007). Furthermore, a Petri net model of semantics service composition based on ontologies was presented by Yang and Zhou (2009).

To investigate these issues and more importantly serve as identifying the direction of the study of knowledge sharing between Petri nets and ontologies, the purpose of this paper is to provide a comprehensive literature overview of knowledge sharing between Petri net models and ontology models to satisfy the obvious need. To our best knowledge, there are no any published papers that review the literature of knowledge sharing between Petri nets and ontologies so far.

The remainder of this paper is organized as follows. In Section 2, we give the basic knowledge about Petri net and ontology theories that are the basis for knowledge sharing used in different approaches discussed later in this paper. In Section 3, we survey the different knowledge representation approaches of ontology to represent and reason about Petri nets. Issues of constructing Petri nets from ontologies are discussed in Section 4. Section 5 presents issues about the applications of the knowledge sharing between Petri nets and ontologies in detail, including applications in business process management (BPM), applications in Web service testing and Web service composition, and applications in some particular domains. Finally, the last section concludes the paper with the summary and possible future work.

2 Petri nets and ontologies

In order to implement knowledge sharing and reusing between Petri nets and ontologies, it is necessary to understand what they are. In the following, we first briefly present some basic notions from Petri nets.

Then, we give a brief introduction to ontologies that is a formal knowledge presentation method describing a conceptualization of some domain.

2.1 Petri nets

Petri nets have been around since 1962 (Petri, 1962) as a graphical and mathematical technique suitable for modeling and analyzing information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic. The Petri net model has a powerful capability to model events and states in a distributed system and to capture sequential, concurrency and event-based control. As a graphical technique, Petri nets can be used as a visual aid for communicating between different communities (for instance, between business analysts and systems different designers), while, as a mathematical technique, Petri nets provide a means for setting up state equations, algebraic equations and other models governing the behavior of systems (Murata, 1989).

A Petri net is a directed graph together with an initial state called *the initial marking*. The underlying graph of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called *places* and *transitions*, which are connected via *arcs*. Graphically, each place is represented by a circle, while transitions are marked as a rectangle. The flow relation of the Petri net is represented by arcs form places to transitions or from transitions to places. A marking assigns to each place a non-negative integer k, meaning that the place is marked with k *tokens*, which are represented as black dots. Places have a finite place capacity restricting the maximum number of tokens they can hold. A transition has a certain number of input and output places representing pre- and post-conditions for the firing of the transition. Sometimes, k tokens are put in a place to indicate that k data items or resources are available. In order to understand the Petri net, a formal definition of a Petri net is given as follows (Liu, 2006):

DEFINITION 1 *A Petri net is a five-tuple* $PN = (P, T, F, W, M_0)$, *where* $P = \{p_1, p_2, \cdots, p_m\}$ *is a finite set of places,* $T = \{t_1, t_2, \cdots, t_n\}$ *is a finite set of transitions,* $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation),* $W: F \to \{1, 2, 3, \cdots\}$ *is a weight function and* $M_0: P \to \{0, 1, 2, 3, \cdots\}$ *is the initial marking, and* $P \cap T = \Phi$ *and* $P \cup T \neq \Phi$.

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state or marking in a Petri net is changed according to the following firing rules (Recker & Indulska, 2007):

- a transition is enabled if each of its input places is marked with as many tokens as the weight of the arc leading to the transition described;
- an enabled transition may or may not fire; and
- a firing of an enabled transition removes the tokens from the input places in accordance to the arc weight and adds tokens to each output place in accordance to the weight of the arc leading from the transition to its output places.

Let us begin with an illustration to introduce the Petri nets. The producer–consumer problem is a very common example in computer science, which includes two processes: (i) a process of producer creates objects, which are put into the buffer; (ii) a process of consumer waits until an object has been put in the buffer, removes it, and consumes it. The Petri net shown in Figure 1 represents the well-known producer–consumer system, where the place buffer can be used to connect the producing process including place P1, P2, transition produce, put into buffer and the consuming process consisting of place P4, P5 and transition remove from buffer, consume. Each token represents an item, and the consumer can consume as long as the place buffer has items (tokens).

Petri nets have been proposed for a very wide variety of applications (Petri, 1962; Murata, 1989; Li & Lara-Rosano, 2000; Guo-Yan, 2006; Liu, 2006; Shen, 2006; Yen, 2006; Recker & Indulska, 2007; Araujo & Roque, 2009). In the following statements, we mainly introduce the application areas of Petri nets. Owing to the generality and permissiveness inherent, Petri nets can be informally applied to any area or

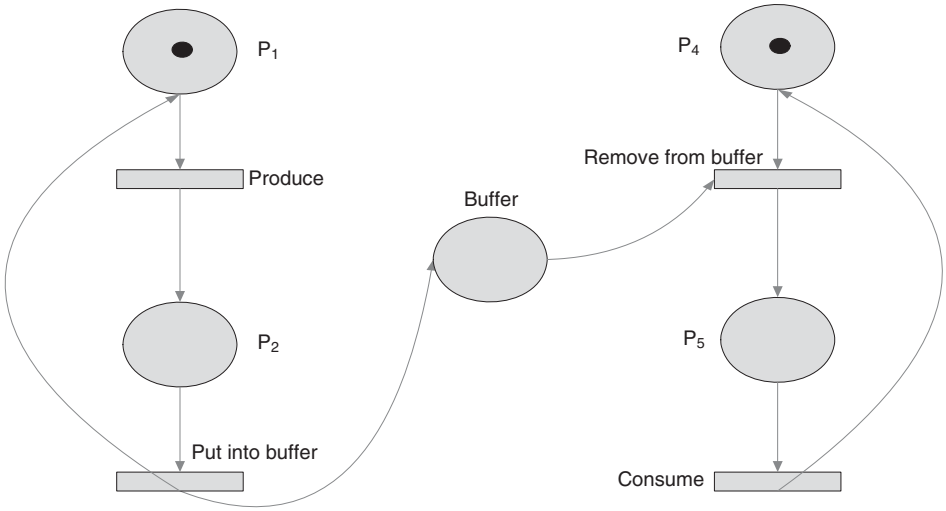

Figure 1 An illustration of Petri nets of a producer–consumer system

system that needs some means of representing parallel or concurrent activities. Methods of analysis for Petri net models can be mainly classified into the following three groups:

- the coverability tree;
- the incidence matrix; and
- reduction or decomposition techniques.

Thus, because of the analysis capability, Petri nets are used widely in some application areas, including distributed software systems, distributed database systems, concurrent and parallel programs, flexible manufacturing/industrial control systems, formal languages, logic inference and decision models. Additionally, performance evaluation and communication protocols are considered to be the successful application areas. Formal methods of protocol engineering, for instance, are based on the analysis of the Petri net model. Generally speaking, the formal methods of protocol engineering mainly provide a standardized guideline for the design state of the whole protocols, which include specification, verification, implementation and testing. For each state, it has a corresponding method and technique. The entire system can be described and analyzed by place/transition net model.

In general, Petri nets are divided into elementary nets, place/transition nets and HLPNs. In order to meet the needs of different applications, some modifications and extensions have been made to the Petri net model. According to ISO/IEC 15909-1 (2002), colored Petri nets (CPN), algebraic Petri nets and predicate/transition nets (PrT-nets), as three subclasses of HLPNs, are widely used in some applications. Additionally, Petri nets have several new variants, including time Petri nets, which are used for performance evaluation and scheduling problems of dynamic systems, and stochastic nets for discrete systems.

2.2 Ontologies

Being a formal knowledge representation method describing a conceptualization of some domain, ontologies can capture both the structure and semantics of information environment. Ontologies describe the nature of the world, and attempt to organize and describe what exists in reality in terms of the properties of, the structure of and the interactions between real-world things. Ontologies have a number of advantages, including adding semantics to data, maintaining knowledge, integrating information, as well as reusing or sharing components (Shanks *et al*., 2003). Ontologies have been used in diverse application domains such as education, business service, Semantic Web, communications, media, financial services and so on (Cardoso, 2007).

The first definition proposed by Neches, Fikes & Finin (1991) is that ‘an ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as rules for combining terms

and relations to define extension to the vocabulary'. The most widely cited definition in the literature is provided by Gruber (1993), who defined an ontology as 'an explicit specification of conceptualization'. Guarino and Giaretta defined an ontology as 'a logical theory which gives an explicit, partial account of conceptualization' (1995). Borst slightly altered Guarino's definition and thought that an ontology is 'a formal specification of a shared conceptualization' (1997). According to the study of the definition, Studer *et al.* (1998) defined an ontology as 'a formal and explicit specification of a shared conceptualization model'. And they interpreted this definition in four levels: (i) *conceptualization* refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon; (ii) *explicit* means that the type of concepts used, and the constraints on their use are explicitly defined; (iii) *formal* refers to the fact that the ontology should be machine-readable; and (vi) *shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.

In general, the knowledge in ontologies is formalized using six kinds of elements: classes, relations, functions, axioms, attributes and instances.

- *Classes* also called *concepts*. A concept is a collection of objects of the domain, representing the basic ideas in the real world. Typically, an ontology can be viewed as a taxonomic hierarchy where some concepts are explained by using others.
- *Relations* represent a type of interaction between concepts of the domain and determine the ontology structure. Formally, an n-ary relation R is defined as a subset of a product of n classes $C_1, \ldots, C_n$, that is

$$R \subseteq C_1 \times C_2 \times \ \ldots \ \times C_n$$

- *Functions* constitute a special case of relations in which the nth element of the relationship is unique for the $n-1$ preceding elements. Formally, a function F is defined as

$$F: C_1 \times C_2 \times \ \ldots \ \times C_{n-1} \rightarrow C_n$$

- *Instances* are used to represent particular elements of a given class.
- *Attributes* describe properties of the individuals of a class. They are usually called data types or concrete domains.
- *Axioms* are used to model sentences that are always true and express conditions to be verified by the elements of the ontologies in order to guarantee its correctness. In addition, axioms allow inferring new knowledge, which is not explicitly represented in the ontology.

Ontologies can be represented by extensible markup language (*XML*), resource description framework (*RDF*), RDF Schema (*RDF(S)*) and *OWL*. But, *OWL*, as a World Wide Web Consortium (W3C) recommendation standard, is widely used in the description of ontologies. OWL comprises three sublanguages of increasing expressive power: *OWL Lite*, *OWL* description logic (*DL*) and *OWL Full*. Compared with *XML*, *RDF* and *RDF(S)*, *OWL* has more mechanisms to express the semantics. OWL defines different properties such as *Object Properties* (link an individual to an individual), *Data Properties* (link an individual to an XML Schema data type value), *Domains and Ranges* (properties link individuals from one domain to individuals from another domain), *Data types and Restriction Types* (Quantifier Restrictions, has Value Restrictions and Cardinality Restrictions) to build an ontology.

Ontologies are becoming more and more popular nowadays. Popular applications of ontologies include knowledge management, natural language generation, enterprise modeling, knowledge-based systems, ontology-based brokers and interoperability between systems (Shanks *et al.*, 2003; Cardoso, 2007). Since their applications, ontology engineering has attracted an important attention and has been the objects of a lot of study in the field of AI (Staab & Studer, 2004).

3 Representation of and reasoning about Petri nets with ontologies

A lot of work has been done on the representation of and reasoning about Petri net models with ontologies to enable sharing knowledge of Petri nets. In this section, all the ways to represent Petri nets are introduced, including construction of a metamodel from a Petri net, construction of an ontology from a

Petri net and representation of a Petri net with an ontology description language. This work can be summarized in Table 1.

3.1 Construction of metamodel from Petri net

A metamodel is a model of the language that captures its essential properties and features (Clark *et al*., 2004). These include the language concepts it supports, its textual and/or graphical syntax, and its semantics (what the models and programs written in the language mean and how they behave). And it is widely believed that the idea of a metamodel is strongly related to the idea of an ontology, which can capture the shared domain knowledge and represent this knowledge. Moreover, a metamodel is considered as an ontology in the field software engineering (Bezivin, 1998).

In order to share the knowledge of a Petri net, much effort has been spent to construct metamodels from Petri nets. Breton and Bezivin (2001) proposed an explanatory and very exhaustive Petri net metamodel in the context of the Object Management Group's[1] model-driven architecture (MDA) initiative, where a set of concepts and relations of Petri net were defined, such as a series of assertions and the terminology. The Petri net metamodel is made up of three layers, which are listed in the following (Gasevic & Devedzic, 2006):

- The first layer mainly defines Petri net metamodel structure, including place, transition, arc, Petri nets itself and mutual relationships between places. It is important to note that the whole Petri net metamodel utilizes Object Constraint Language (OCL) to explain the source and destination of each arc.
- The second layer is the Petri net situation metamodel, which defines a particular situation of Petri nets, especially making full use of the marking and token concepts.
- The purpose of Petri nets is to describe the process. In order to present the execution of Petri net, the third layer defines Petri net execution metamodel. In fact, a sequence of particular situation is provided by using a new Petri net concept called *move*.

With regard to the execution semantics of Petri net metamodel, the action semantics for unified modeling language (UML) was defined by Object Management Group (1998) in order to specify action semantics, but some constraints in OCL were not defined because the constraints alone in OCL do not specify the whole execution semantics.

Based on the extension of the metamodel, the UML profiles were defined by Kogut *et al*. (2002), where the authors supplemented some concepts such as tagged values and constraints on the basis of the UML metamodel. In order to represent Petri net models, Hansen (2001) proposed a CPN UML profiles. In more detail, the author extended the previous UML metamodel by adding one important concept called stereotypes, which had a major role in Petri net nodes, transition, places and arcs. Further, Gasevic and Devedzic (2006) used a UML profile to represent the development of UML-based Petri net ontology, in which some constraints of concepts were the same as previous Petri net metamodels, using the OCL language to specify the semantics.

In addition, a Petri net metamodel, namely Petri Net Markup Language (PNML for short) (Billington *et al*., 2003; Weber & Kindler, 2003) is described in the ISO/IEC 15909 standard (ISO/IEC 15909-1, 2002; ISO/IEC 15909-2, 2005). This standard defines an open Petri net metamodel and the PNML aims at facilitating the interchange of any kind of Petri nets between different tools or applications. Moreover, the standard focuses on the representation issues of the Petri net components (such as place, transitions, arc, terms, etc.) and graphical components (colors, labels, etc.). More important, the Petri net metamodel uses OCL to represent constraints between concepts of Petri nets.

Various approaches to construct metamodel from Petri nets are listed in Table 2 with respect to the issues of the expressive power and the reasoning ability of metamodel.

3.2 Construction of ontology from Petri net

A metamodel gives the useful representation of Petri net concepts, but it cannot show how a Petri net model can be used by the Semantic Web directly (Gasevic & Devedzic, 2006). In order that an ontology

[1] `http://www.omg.org/`

Table 1 The representation and reasoning of Petri nets with ontologies

	Source: Petri nets	Target: ontologies	Corresponding references	Mainly discussed issues
Construction of metamodel from Petri nets	High-level Petri net (HLPN)	Metamodel	Breton and Bezivin (2001)	Petri net metamodel, Petri net definition metamodel, Petri net situation metamodel and Petri net execution metamodel
	Petri net	UML metamodel	Gasevic and Devedzic (2006)	UML-based Petri net model
	Colored Petri net	UML profile	Hansen (2001)	Petri net UML profile
	Petri net	PNML	Billington *et al*. (2003) and Weber and Kindler (2003)	PNML that is based on Petri net metamodel
Construction of ontologies from Petri nets	Static Petri net	Ontology	Gasevic and Devedzic (2006, 2007)	Petri net ontology
	HLPN	Ontology	Lu *et al*. (2007), Vidal *et al*. (2006, 2010) and Vidal and Lama (2009)	HLPN ontology
	Object-oriented Petri net	Ontology	Feng *et al*. (2008)	Hierarchical Object-Oriented Petri net ontology
	Basic Petri net	Ontology	Takai-Igarashi (2005), Soffer *et al*. (2008) and Thomas *et al*. (2011)	Ontological semantics for Petri net
Representation of Petri nets with ontologies description language	Static and dynamic HLPN	FOL	Tan (2010), Vidal and Lama (2009) and Vidal *et al*. (2010)	The description of static and dynamic HLPN with FOL and the reasoning of FOL
	HLPN	DL	Lu *et al*. (2007)	The description of HLPN with DL and the reasoning of DL
	Petri net	OWL DL	Zhang *et al*. (2011)	The OWL DL definition of Petri net
	Petri net	OWL	Brockmans *et al*. (2006), Gasevic and Devedzic (2006, 2007), Wang *et al*. (2007) and Ma and Xu (2009a)	The OWL definition of Petri net ontology and the OWL reasoning
	Fuzzy Petri net	Fuzzy OWL	Zhang *et al*. (2012)	The representation of fuzzy Petri net with fuzzy OWL
	HLPN	F-Logic	Vidal *et al*. (2006)	The description of HLPN with F-Logic and the reasoning of F-Logic

PNML = Petri Net Markup Language; FOL = first-order logic; DL = description logic; OWL = Web ontology language; F-Logic = Frame-Logic.

Table 2 Construction of metamodel from Petri net

Corresponding reference	Petri nets	The expressive power of metamodel	The definition of constraints	Reasoning ability
Breton and Bezivin (2001)	High-level Petri net	metamodel	OCL	No
Gasevic and Devedzic (2006)	Petri net	UML metamodel	OCL	No
Hansen (2001)	Colored Petri net	UML profile	OCL	No
Billington *et al*. (2003) and Weber and Kindler (2003)	Petri net	PNML	OCL	No

OCL = Object Constraint Language.

can enable sharing knowledge represented by Petri nets on the Semantic Web (Koschmider and Oberweis, 2005; Gasevic & Devedzic, 2006, 2007), new research on how to construct ontology from Petri net has emerged and some researchers have tried to construct a Petri net ontology to share Petri nets on the Semantic Web. Gasevic and Devedzic (2006, 2007) proposed a Petri net ontology with the aim of enabling an effective sharing of Petri nets, solving the drawbacks of the XML-based syntactic approaches. It is interesting to note that this ontology consists of two submodels: one is about core Petri net concepts, and another is Petri net properties. Moreover, two kinds of structuring mechanisms, called *pages* and *modules*, are introduced in the core Petri net concepts in order to make Petri net models easy to maintain. Specially, the Petri net ontology utilizes OCL to constrain some concepts such as *PlaceReference* and *TransitionReference*. Compared with the above Petri net ontology, Koschmider and Oberweis (2005) presented a novel ontology for Petri nets to facilitate the semantic interconnectivity of business processes. Many constructs of core Petri net concepts presented in Koschmider and Oberweis (2005) were different from those by Gasevic and Devedzic (2006, 2007), which used the OWL elements *Classes*, together with the taxonomic construct *SubClassesOf* and *Properties* to describe Petri net elements. In this ontology, the classes of *fromPlace* and *toPlace* replace the *arc* and derive subclasses of *delete* and *insert*, respectively. Similarly, the class of *place* has three subclasses: *number*, *indistinguishable* that can distinguish arcs between two types of arcs and *individualDataItem*. Especially, according to the *hasMarking* property values in class *place*, the ontology can describe different kinds of Petri nets. For instance, if place contains several tokens and a capacity limit, the Petri net belongs to place/transition net and if place has one or zero tokens, the Petri net is a condition/event net. Moreover, the class of *transition* is the class *logicalConcept* with the properties *hasOperation* and *hasAttribute*.

However, the Petri net ontology, as mentioned above, can only describe static Petri net models and not fully share all knowledge that Petri nets represent. There are a few new proposals that have emerged relating to ontology construction from Petri net which can describe the static structure and dynamic behavior of Petri net. A HLPN ontology was presented by Vidal *et al*. (2006), where the static structure and the dynamic behavior of a HLPN were described semantically and declaratively. In more detail, the static model of a HLPN ontology is only a graph, which describes some concepts such as nodes, arcs, transition, algebra, variable and signature and relationships between those concepts. It should be noted that algebra mainly defines the arc explanations and transition guard in order to describe other concepts at a syntactic level. In addition, in order to describe the dynamic behavior of a HLPN, the authors defined a HLPN ontology dynamic model, where *HLPNExecution* and *Firing* concepts were mainly defined to represent dynamic processes. For example, *HLPNExecution* shows an initial status, current status and the status changes between them. And the *Firing* concept is responsible for the status changes from initial status to current status, and directly contacts with the transition fired. When a transition fires, it must remove tokens from the input places and add new ones to output places. In addition, this ontology is composed of two taxonomies:

- a taxonomy that describes the main components of a HLPN, capturing the vocabulary and semantics based on the ISO/IEC 15909-1 (2002) standard, including static model and dynamic model;
- a set of axioms that constrain the creation of the instances of the taxonomy, restricting the range and domain of the relations, and the values of attributes in order to ensure that a HLPN is correctly constructed, avoid the definition of incorrect HLPN, and define how a dynamic model should be executed.

Similarly, based on the International Standard ISO/IEC 15909-1, HLPN ontologies (Vidal & Lama, 2009; Vidal *et al*., 2010) are described, which axiomatize all the aspects of HLPN definition and execution models, including static model, situation model and execution model. In contrast with the early studies of Vidal *et al*. (2006), the HLPN ontologies defined by Vidal and Lama (2009) and Vidal *et al*. (2010) give a set of rules that enable both the management of the HLPN static model and its execution and represent the semantics of the execution model.

In Lu *et al*., (2007), a HLPN ontology is divided into Structure-ontology and Algebra-ontology from a modeling perspective for reuse and easy analysis. The Structure-ontology called the static model of a Petri net ontology mainly represents some concepts, relationships between concepts and axioms, where concepts are comprised of place, transition and arc. The Algebra-ontology called the dynamic model of a Petri net ontology mainly contains several important concepts:

- sort, representing the resource of Petri net, like the token described in Gasevic and Devedzic (2006);
- variables, which are some annotations assigned by arcs;
- operators, including $>$, $<$, $=$ and so on for using the condition judgment;
- terms, describing the conditions, which are composed of variables and operators.

The values of those concepts change along with the development of a HLPN. It should be noted that HLPN ontologies and common ontologies can be described by DLs. To connect these two kinds of ontologies, some link properties described by e-connection have been defined in the literature. In (Lu *et al*., 2007) DLs are applied to indicate the static and dynamic models of Petri nets, representing the semantics of Petri nets and to share knowledge between Petri nets and ontologies. HLPN ontologies enable the reuse of the Petri net because the net graph can be described independently by the algebra that annotates it.

In addition, other efforts in this direction were presented by Takai-Igarashi (2005) and Feng *et al*. (2008). In more detail, Takai-Igarashi (2005) constructed a Cell Signaling Networks Ontology (CSNO) to prevent the Petri net graphs, which represents biological pathways, from being too complicated and unmanageable as well as from being inconsistent. Especially, a HOOPN modeling method based on ontology was presented by Feng *et al*. (2008), in which this method was composed by object identification place that can identify a class, internal object net that can depict the behaviors of a class, and data dictionary that can declare the attributes of a class. The main advantage is to extend Petri nets with object-oriented concepts and provide excellent concepts to model real-world problems. Particularly, compared with Petri net ontologies with graphical representation, as mentioned above, the formal definition of HOOPN can describe accurately and semantically the knowledge that Petri nets represent.

In order to verify the effectiveness of process models, many researchers have proposed methods which model process models with Petri nets and evaluate process models by mapping Petri nets into ontologies for checking completed process models for structure and behavioral properties. For example, Soffer *et al*. (2008) demonstrated the use of ontological semantics for Petri nets based on process models. Moreover, the formal definition of Generic Process Models (GPM for short) which provides a process specification semantics based on ontological concepts, is given, and some definitions of Petri nets and their properties are also provided. To verify the completed models, a translation from Petri nets to GPM is established by assigning the semantics to places, transitions and arcs. Besides, Thomas *et al*. (2011) proposed an approach of the use of Petri nets to model the accurate behavior of the game player in serious games and the mapping from Petri nets to ontologies to analyze and diagnose learner's mistakes.

Various approaches that constructing ontology from Petri net are listed in Table 3, which includes corresponding reference, source, target, semantics constraints, mainly models and formal definition. It should be noted that, by the formal definition of Petri net model and ontologies, we can find out the relationships between Petri nets and ontologies.

3.3 Representation of Petri net with ontology description language

In order to reuse and share knowledge of Petri nets, there are other approaches proposed in the literature, which use ontology description languages to represent Petri net concepts and their relationships. Here we summarize these approaches in Table 4.

Table 3 Construction of ontology from Petri net

Corresponding reference	Source: Petri net	Target: ontology	Semantic constraints	Mainly models	Formal definition
Gasevic and Devedzic (2006, 2007) and Koschmider and Oberweis (2005)	Static Petri net	Ontology	Yes	Petri net static model and property hierarchy model	No
Vidal and Lama (2009) and Vidal *et al*. (2010)	High-level Petri net (HLPN)	Ontology	Yes	HLPN static model, HLPN situation model and HLPN execution model	No
Feng *et al*. (2008)	Object-oriented Petri net	Ontology	Yes	Object identification place, internal object net and data dictionary	Yes
Takai-Igarashi (2005)	Petri net	Ontology	No		No
Soffer *et al*. (2008) and Thomas *et al*. (2011)	Basic Petri net	Ontology	Yes		Yes
Lu *et al*. (2007)	Static HLPN	Ontology	Yes	Structure-ontology, Algebra-ontology and link properties	No
Vidal *et al*. (2006)	Static and dynamic HLPN	Ontologies	Yes	HLPN static model, HLPN dynamic model and the description of the HLPN ontology axioms	No

Table 4 Representation of Petri nets with ontologies description language

Corresponding reference	Source: Petri nets	Target: ontologies	Reasoning	Reasoner
Vidal *et al*. (2006)	High-level Petri net	F-Logic	Yes	FLORA-2
Lu *et al*. (2007)	High-level Petri net	DL	Yes	
Tan (2010), Vidal and Lama (2009) and Vidal *et al*. (2010)	Static and dynamic high-level Petri net	FOL	Yes	FLORA-2
Zhang *et al*. (2011)	Petri net	OWL DL	No	
Brockmans *et al*. (2006), Gasevic and Devedzic (2006, 2007), Ma and Xu (2009a) and Wang *et al*. (2007)	Petri net	OWL	Yes	
Zhang *et al*. (2012)	Fuzzy Petri net	Fuzzy OWL	Yes	DeLoean

F-Logic = Frame-Logic; DL = description logic; FOL = first-order logic; OWL = Web ontology language; FLORA-2 = F-Logic tRAnslator .

F-Logic (Frame-Logic) is considered as an ontology language as well as a language for building applications that use ontologies. As an ontology modeling language for knowledge representation, F-Logic accounts in a clean and declarative fashion for most of the structural aspects of object-oriented and frame-based languages (Angele *et al*., 2009). *Furthermore*, F-Logic has a model-theoretic semantics and a complete and sound resolution-based proof theory (Kiefer *et al*., 1995). Vidal *et al*. (2006) proposed a formal approach of mapping HLPN mathematical specification into an F-Logic language. It is important to note that the F-Logic language is a machine-readable formulation. On one hand, the concepts and relations in the HLPN are represented by means of F-Logic, and on the other hand, a set of axioms that guarantee that HLPNs are correctly created and its execution are formalized in F-Logic. Through a general reasoner, called F-Logic tRAnslator (FLORA-2), it is easy to check whether the axioms are verified or not.

In addition to F-Logic, first-order logic (FOL for short) can also represent and reason Petri net models. FOL, also called first-order predicate calculus, determines a domain of discourse that specifies the range of the quantifiers. Vidal and Lama (2009) and Vidal *et al*. (2010) proposed two approaches for representing static and dynamic HLPN models with FOL, where the purpose of FOL is to constrain the axioms of HLPN. Based on the FOL, *FLORA-2* (Yang *et al*., 2003) reasoner is established to provide the sentences and rules to create, modify and delete the instances of the HLPN ontology and a set of queries for reasoning with the Petri net. Besides, Tan (2010) gave a formal approach for describing Petri nets called situation calculus ontologies of petri nets (SCOPE) with FOL, which makes the full use of the strong

correspondence between the situation in situation calculus and the marking in Petri nets. Moreover, a set of FOL axioms can support concrete reasoning in Petri nets.

Although FOL is a knowledge representation and ontology language, DL (a subset of FOL) is more popular and accepted because of its powerful expressive ability. DLs are a logical reconstruction of the so-called frame-based knowledge representation languages, with the aim of providing a simple well-established Tarski-style declarative semantics to capture the meaning of the most popular features of structured representations of knowledge (Baader *et al*., 2002).

Nowadays, *DL*s have gained even more popularity due to their applications in the context of the Semantic Web (Baader *et al*., 2002; Angele *et al*., 2009). As the logical foundation of the Semantic Web, DLs have good syntax rules and reasoning ability. Hence, a new method was proposed by Lu *et al*. (2007) in order to translate an information model represented by a Petri net into a knowledge base represented by DL, where the reasoning of the Petri net can be done through the reasoning mechanism of DL. In more detail, Lu *et al*. (2007) defined a DL knowledge base to capture the characteristics of HLPN, where the base consists of an ABox that describes concepts of HLPN, and a TBox that presents axioms that constrain the semantics of concepts in the structure of HLPN. It is important to note that the purpose of the knowledge base is to formalize and analyze the HLPN. In addition, the reasoning of HLPN is implemented through the reasoning of DL.

As an extension of RDF or RDFS, OWL is formed based on DARPA Agent Markup Language (DAML) + ontology inference layer or ontology interchange language (OIL). So far, OWL (Dean *et al*., 2004), a W3C recommendation, has been an ontology description language standard and is the fundamental support for the Semantic Web. Moreover, the main advantage of OWL is to provide more primitives to support richer semantic representation and reasoning. Therefore, in order to implement the representation of and reasoning about Petri nets, several approaches based on OWL have been proposed as will be introduced in the following.

In order to enable semantic information exchange between two semantic business models, Brockmans *et al*. (2006) proposed an approach for representing Petri nets with OWL. Moreover, Gasevic and Devedzic (2006, 2007) gave the OWL definition of the concepts for the core Petri net ontology and the ontologies of upgraded Petri nets (Strbac, 2002). Further, the OWL ontologies attached to the Petri net model were defined by Wang *et al*. (2007), where test data of Petri nets can be generated by OWL reasoning. Besides, Ma and Xu (2009a, 2009b) discussed a novel approach for describing Petri nets with PNML in combination with OWL. Specifically, a Petri net structure is described with PNML, and place elements have their semantic specification with OWL. In addition, the ontology reasoner can smoothly achieve the automatic sharing composition operation of Petri nets. Here, the Petri net is applied to model and analyze dynamic behavior relationships on the Semantic Web.

OWL DL, as the sublanguage of OWL, is equivalent to DL *SHOIN* (*D*) (Hustadt *et al*., 2004), which provides a good balance of expressive and reasoning power. For example, Zhang *et al*. (2011) gave the formal definition of Petri nets and OWL DL ontologies, and represented Petri nets with OWL DL ontologies by mapping some key features of Petri nets into classes, properties and axioms of OWL DL ontologies.

As information imprecision and uncertainty exist in many real-world applications such as manufacturing and information systems, the Petri net model is extended to a fuzzy Petri net (FPN) model to represent uncertainties. In order to share and reuse the FPNs, Zhang *et al*. (2012) proposed a fuzzy OWL ontology approach for representing FPNs, where they map the key features of FPNs into fuzzy classes, fuzzy properties, fuzzy individuals and fuzzy axioms of fuzzy OWL ontologies. By means of the existing fuzzy ontology reasoner *DeLoean* (Bobillo & Delgado, 2007), reasoning on FPNs is implemented.

4 Petri net representation of ontologies

Modeling ontologies with Petri nets is a crucial approach to realize knowledge sharing between Petri nets and ontologies. The aim of constructing a Petri net model from an ontology is to adopt the power capabilities of describing dynamic behavior and analyzing knowledge of Petri nets to model ontology. Much work has been carried out toward Petri net representation of ontologies. By summing up these

Table 5 Petri net representation of ontologies

	Ontology or ontology description language	Corresponding references	Mainly discussed issues
Mapping ontology into Petri net	Task ontology	Liu (2006)	Task ontology, Petri net model and intermediate model of mapping
	BWW ontology	Recker and Indulska (2007)	Translating rules
	WSMO	Yang and Zhou (2009)	The formal definition of ontology, service and semantic service composition, and the formal definition of Petri net of service composition, optimization algorithm
Mapping ontology description language into Petri net	OWL-S	Brogi *et al*. (2007), Vidal *et al*. (2007) and Wang *et al*. (2007)	The Petri net model of OWL-S process, verification, implementation and reasoning
	DAML-S	Moldt and Ortmann (2004) and Narayanan and McIlraith (2002)	The Petri net semantics of DAML-S implementation
	DDL	Ma and Xu (2009)	Constructing Petri net models for DDL and incidence matrix
	RDF	Yim *et al*. (2011)	The semantics of RDF model in CPN and the mapping algorithm
	OWL + SWRL	Zhang *et al*. (2006)	The Petri net model for OWL DL ontology and SWRL
	FOL	Murata and Zhang (1988), Peterka and Murata (1989), Chuang *et al*. (1993) and Liu (1994)	The Petri net model for logic program, its syntax, its translation procedure, its semantics and inference

BWW = Bunge, 1977, Wang & Weber, 1993; WSMO = Web Service Modeling Ontology; OWL-S = Web ontology language for Web Service; DAML-S = DARPA Agent Markup Language-Service; DDL = dynamic description logic; FOL = first-order logic; RDF = Resource Description Framework; SWRL = Semantic Web Rule Language; CPN = colored Petri net.

approaches, we give an overview of modeling ontologies with Petri nets composed of two aspects: one corresponding to map ontologies into Petri nets and the other corresponding to map ontology description languages into Petri nets. Petri net representations of ontologies can be summarized in Table 5, which contains corresponding references and the main issues discussed in these papers.

4.1 Mapping ontologies into Petri nets

Mapping ontologies into Petri nets means the use of Petri nets to model knowledge that ontologies represent to share knowledge between them. On one hand, Petri nets can dynamically describe ontology knowledge, and on the other, they provide some techniques to analyze the ontology models. To enhance knowledge reusability and sharability over different applications, Liu (2006) proposed a framework, called TTIPP (Task analysis, Task ontology, IDEF0, Petri net, PNML), in which Petri nets are used to completely model a task ontology (Guarino, 1998) and formally specify the terminology associated with a problem type. By means of an intermediate model, namely the IDEF0 model (Feldmann, 1998), the task ontology is converted into a Petri net model. Compared with the IDEF0 model, the greatest advantage of the Petri net model is the use of its time-based function, including cumbersomeness, ambiguity in activity specification and its static nature.

As a popular process modeling technique, the use of Petri nets can provide an abundance of analysis techniques to evaluate process modeling languages. However, an analysis of the strengths and weaknesses of Petri nets for modeling business processes cannot be conducted. An approach to map ontologies into Petri nets is proposed, which can evaluate the representational capability of Petri nets. The representation mapping from the BWW ontology (Bunge, 1977; Wang & Weber, 1993) to a Petri net model is built by Recker and Indulska (2007) in order to establish a comprehensive understanding of the ontological completeness and clarity that Petri nets exhibit. In more detail, the mapping is implemented by converting BWW real-world concepts such as *thing*, *class*, *state law*, *lawful state space*, *event, transformation*, *stability condition*, *acts-on*, *unstable state*, *internal event* and *well-defined event* into Petri net concepts.

Based on the representation mapping, the evaluation shows that there are a number of representational limitations of Petri nets including construct deficit, construct redundancy and construct overload.

Recognition of semantics and information extraction are two important issues in the process of service composition. In order to solve these issues, a Petri net model of ontology-based semantics service composition is proposed by Yang and Zhou (2009). In their paper, the formal definitions of ontology, service, semantic service composition and Petri net of service composition are given. By designing grammar detection and semantics authentication, a Petri net model of service composition is implemented to improve search speed of service composition. In particular, a greedy genetic algorithm is applied to optimize Petri net models.

4.2 Mapping ontology description languages into Petri nets

Information on the Semantic Web is described with ontology description languages, such as OWL, RDF and OWL-S. Since Petri nets have a powerful capability to model events and states in information processing systems and to formally analyze the processes (Nielsen & Sassone, 1988), the aim of mapping ontology description languages into Petri nets is to use the formal analysis of Petri nets to supplement the functions of ontologies. Nowadays, OWL-S and DAML-S (DAML-Service) are used to specify the process of Web service application. Thus, the representation of the OWL-S and DAML-S with Petri nets has been the focus of a great deal of research.

A formal approach for modeling OWL-S service choreography with HLPNs is discussed by Vidal *et al.* (2007), in which each OWL-S is transformed into a Petri net. As consequence of this approach, the Petri net can be used to check the correctness of the OWL-S choreography model by verifying some Petri net properties and can interact with services. Moreover, Wang *et al.* (2007) proposed a transformation from OWL-S processes to Petri net models:

i. one perform action is mapped to a transition;
ii. inputs and preconditions of the perform action are mapped to the places holding tokens pointing to the transition;
iii. outputs and effects of the perform action are mapped to output arcs and places of the transition.

Based on the Petri net model of OWL-S process and ontology reasoning, a test case (test process and test data) generation approach is presented. In particular, the correctness of a Petri nets is guaranteed by defining the marking reachability and construct inclusion, and a prototype system TCGen4WS is implemented. Besides, Brogi *et al.* (2007) developed a tool – named OWLS2PNML – that translates OWL-S descriptions into Petri nets described by a PNML (Weber & Kindler, 2003) file. The main contribution of OWLS2PNML is to make it possible to analyze and verify OWL-S and reason with heterogeneous services.

In order to automatically describe, simulate and verify Web services, Narayanan and McIlraith (2002) proposed Petri net models for describing the operational semantics of DAML-S, a logic language describing Web services. In addition, they defined the semantics of atomic processes by situation calculus axioms, map the situation calculus axioms into the corresponding Petri net structure, and describe the net structures for various control constructs defining composite processes. Specifically, this approach extends the basic Petri net formalism with typed arcs, hierarchical control, durative transitions, parameterization, typed (individual) tokens and stochasticity. Moreover, a tool that maps a DAML-S description of a Web service into a Petri net is implemented to support the various verification and performance analysis. Moldt and Ortmann (2004) implemented the Petri net semantics of DAML-S by means of a tool called DaGen, in which a DAML-S description is automatically translated into a HLPN. As a plug-in for reference net workshop (RENEW) (Kummer *et al.*, 2003), DaGen provides a Petri net simulator that allows for different views, including control flow, data and execution.

DL only expresses and reasons with static knowledge, and does not meet the demand of processing dynamic knowledge. Faced with the deficiency, many researchers proposed dynamic DL (DDL), which extends the traditional DL with dynamic aspects and combines representation of and reasoning with static knowledge and dynamic knowledge (Shi *et al.*, 2005). Since DDL does not describe and analyze

relationships among actions, especially concurrent relationships, and Petri nets have powerful capabilities to describe these relationships, the use of Petri nets to model DDL can achieve knowledge sharing between Petri nets and DDL. Ma and Xu (2009a, 2009b) proposed a method for modeling and analyzing DDL action relationships with Petri net models, which supplements the function of description and analysis of DDL action theory. In more detail, the transformation from DDL actions to the Petri net model includes four steps:

- considering each atomic action and its condition descriptions as a *transition* and *places* of Petri nets, respectively;
- obtaining the Petri net incidence matrix corresponding to the DDL description;
- using DL reasoning technology to find equivalent condition descriptions of atomic actions;
- executing sharing synthesis operation of Petri nets among actions with their Petri nets incidence matrixes and then getting the Petri net model of DDL actions.

Basically, in the mapping of DDL actions to Petri net model, the actions and the relationships among actions are transformed to an equivalent Petri net model.

In addition, Yim *et al*. (2011) proposed a method for translating the RDF model into a CPN, which represents the semantics of the RDF model in the CPN by mapping the classes, resources, literals, statements and properties of the RDF model onto CPN places and representing the relationships between those classes and properties as token transitions in CPN. Based on the proposed method, an application can draw an inference with a CPN model and answer RDF queries, verify the correctness of transformation by CPN tools. In particular, a prototype database system for demonstrating the method is also implemented.

Logic programming defined by FOL, is also applied to knowledge representation, and PrT-nets can provide a uniform mathematical formalization of knowledge presentation and control implementation for inference. The combination of logic programming and PrT-nets was described by Murata and Zhang (1988), where a PrT-net is proposed to represent and analyze a logic program. The syntax, transformation procedure and semantics are also discussed. Through the transformation, a Horn clause program can be automatically translated into a PrT-net and various inference problems in the program are also converted to problems in the corresponding Petri net model. Similarly, Chuang *et al*. (1993) proposed a method for modeling Horn clauses and first-order predicate logic with Petri nets, where the logical inference was implemented by computing T-invariants. In particular, Liu *et al*. (1994) addressed the topic of modeling the semantics of logic program with PrT-nets in order to realize the feature of non-monotonic inference.

Petri nets not only have the powerful capability to model logic programs defined by FOL, but can also describe OWL DL ontologies and semantic web rule language (SWRL) rules to realize knowledge sharing between ontologies and Petri nets. For example, a novel approach for reasoning with SWRL rules and OWL ontologies based on PrT-nets was presented by Zhang *et al*. (2006). This work was implemented by transforming OWL DL and SWRL rules into PrT-nets, which requires four steps:

- mapping OWL axioms such as constants, class and property to PrT-net model consisting of a transition between two places with the free variables universally quantified at the arc;
- mapping OWL class constructors including *conjunction*, *disjunction*, *universal restriction*, *existential restriction*, *complementof* and *cardinality restrictions* to PrT-net models;
- translating SWRL rules to PrT-net models;
- computing T-invariants to realize rule inference.

Based on the discussions above, the approaches that map ontology description languages to Petri nets are summarized in Table 6, which contains corresponding references, sources (ontology description languages), targets (Petri nets), verification, implementation and inference. In the next section, we elaborate the applications of knowledge sharing between Petri nets and ontologies.

5 Applications

With the combination of Petri net modeling and analysis techniques and ontology knowledge representation and reasoning techniques, many problems have been resolved. Sharing knowledge between

Table 6 Mapping ontology description language into Petri net

Corresponding reference	Source: ontology description language	Target: Petri net	Verification	Implementation	Inference
Narayanan and McIlraith (2002)	DAML-S	Basic Petri net	Yes	Yes	Yes
Moldt and Ortmann (2004)	DAML-S	HLPN	No	Yes	No
Vidal *et al*. (2007)	OWL-S	HLPN	Yes	No	No
Brogi *et al*. (2007) and Wang *et al*. (2007)	OWL-S	Basic Petri net	Yes	Yes	Yes
Ma and Xu (2009)	DDL	Basic Petri net	No	No	No
Yim *et al*. (2011)	RDF	CPN	Yes	Yes	Yes
Zhang *et al*. (2006)	OWL + SWRL	PrT-net	No	No	Yes
Murata and Zhang (1988), Peterka and Murata (1989), Chuang *et al*. (1993) and Liu (1994)	FOL	PrT-net	No	No	Yes

DAML-S = DARPA Agent Markup Language-Service; HLPN = high-level Petri net; OWL-S = Web ontology language for Web Service; DDL = dynamic description logic; RDF = Resource Description Framework; CPN = colored Petri net; PrT-net = predicate/transition net; FOL = first-order logic.

Table 7 Applications of knowledge sharing between Petri net and ontology

		Applications in Web service		Applications in some particular domains		
	Applications in business process management	Web service testing	Web service composition	Biological networks	IHCS	Serious game
Mapping Petri net model into ontology	√			√		
Mapping Petri net into ontology description language	√					
Mapping ontology into Petri net	√		√		√	√
Mapping ontology description into Petri net		√	√			

IHCS = Incident or Hazard Command System.

Petri nets and ontologies has been applied in BPM, Web services, biological networks, serious games and so forth. Here we summarize some applications of knowledge sharing between Petri nets and ontologies in Table 7.

5.1 Applications in business process management

In BPM, building business process capabilities is seen as a major challenge for senior executives. Due to the rapid growth of electronic market activities, it is needed to establish interconnectivity of business processes in order to reduce communication efforts. An approach for representing Petri nets with OWL was described by Brockmans *et al*. (2006), where the aim of these approaches is to enable semantic information exchange. Also a background ontology is modeled in UML for improving the semantic alignment of business process, and the alignment of Petri nets by making use of existing technology for ontology alignment was described for solving ambiguity issues of Petri nets. Similarly, Koschmider and Oberweis (2005) addressed semantic representation of distributed business processes, in which semantic metadata for business processes modeled with Petri nets is defined by mapping structured data in Petri nets into the ontology elements described in OWL. In addition, the process data contained in places was reasoned about using OWL DL.

In addition, Recker and Indulska (2007) proposed an evaluation of business process modeling by mapping BWW constructs based on an ontology into Petri net constructs, which facilities a deeper understanding of the capabilities and weaknesses of process modeling technique and ultimately leads to an increased quality of process models. The mapping from Petri net constructs into GPM interpretation based on ontological constructs was proposed by Soffer *et al*. (2008), which, in turn, can provide modeling rules for developing process modeling, and avoid process modeling problems by the verification of the process models.

5.2 Applications in Web service

Web services in the Semantic Web (Berners-Lee *et al*., 2001), called Semantic Web services, are described through ontologies (Gomez-Perez *et al*., 2003), which formally represent service features by using a semantic markup language that follows a logical paradigm (such as a DL). This description will allow agents to understand what the service is and facilitate its discovery, composition and invocation.

One of the most often used ontologies to specify Semantic Web services is OWL-S (Martin, 2004), which uses the OWL (Dean *et al*., 2004) language capabilities to represent the property and function of services and enables agents to make inferences about it. OWL-S is defined as a W3C standard to provide a computer-interpretable description of the services, service access and service composition.

Currently, Petri nets are widely applied in the field of Web services because Petri nets have the ability to model Web services and execute and analyze the process of services. For example, Petri nets can solve the problem of semantics recognition and information extraction in the process of service composition. Some researchers utilize Petri nets to model Web services, which are mainly applied in Web service testing and Web service composition. In the following, we briefly introduce these applications.

5.2.1 Web service testing

Web service testing is an emerging area, where most researchers focus on applying Petri net models to Web service testing processes. Yi and Kochut (2004) applied a CPN model, extending the Petri net model, to model and verify service composition. Narayanan and McIlraith (2002) proposed using the DAML-S ontology to describe service semantics. Specially, they translated the DAML-S semantic specification into a Petri net specification and tested Web services by simulating their execution under different input conditions.

Wang *et al*. (2007) addressed automatic test case generation in the context of Web service testing. This research took OWL-S as the input to the test generator, and transformed OWL-S into a Petri net model to provide a formal representation of the structure and behavior of the service under test. Test cases were generated from two aspects:

i. test step generation by traversing various Petri nets execution paths;
ii. test data generation by reasoning over the Petri net ontology.

Vidal *et al*. (2007) presented an approach for utilizing HLPNs to formally describe Semantic Web service choreography described in the OWL-S language. In this paper, a Petri net was created to model OWL-S choreography which was based on the process concept of Web services. And the purpose of the approach is to represent the state of process, including inputs or outputs, execution conditions and the state of service after that execution. Also, the Petri net can be used to check the correctness of the OWL-S choreography models and to interact with the service by executing the information described in OWL-S models. In addition, the mapping from OWL-S composite processes into Petri nets was discussed by Brogi *et al*. (2007), where a tool, named OWLS2PNML, was also implemented to handle the sharing of the input or output data among different processes and verify the OWL-S services.

5.2.2 Web service composition

As Web services increasingly mature, more and more stable and easy-to-use Web services are shared on the Semantic Web. But the individual Web services can provide limited functionality. In order to make full

use of Web services shared on the Web, it is necessary to combine Web services to accelerate the development of systems and to meet the needs of users. Currently, Petri nets have a strong capability to model events and states in a distributed system and to capture sequential, concurrent and event-based control. Thus, some researchers suggest using Petri nets to model Web service composition.

A key application for knowledge sharing between Petri nets and ontologies is Web service composition (Narayanan & McIlraith, 2002; Moldt & Ortmann, 2004; Yang & Zhou, 2009). Narayanan and McIlraith (2002) proposed an approach for composing Web services to perform some desired task. This includes two phases: exploiting the DAML-S ontology to provide semantic markup of the content and capabilities of Web services, and constructing an execution semantics for DAML-S using Petri nets that are an extension of the basic Petri net, including typed arcs, hierarchical control, durative transitions, parameterization, typed tokens and stochasticity. In addition, Moldt and Ortmann (2004) implemented a tool for automatic translation from DAML-S to HLPNs in order to compose some tasks of Web services. In addition, aiming at solving the problems of semantics recognition and information extraction in the process of service composition and information extraction in the process of service composition, Yang and Zhou (2009) proposed a Petri net model of semantics service composition based on Ontology, where the Web Service Modeling Ontology was applied to describe semantic service composition.

5.3 Applications in some particular domains

While there is a lot of research on sharing knowledge between Petri nets and ontologies, little research exists that incorporates Petri nets and ontologies in emerging applications. Several such applications are presented as follows.

In biological networks, Petri nets are widely used because they offer a concise graphical representation of biological concepts and the quantitative analysis of dynamical behaviors of biological entities. But some inconsistencies appear in existing Petri net modeling of cellular networks. Thus, it is necessary to use other methods to semantically describe Petri net models to develop an ontology and then share Petri net domain knowledge. In order to standardize Petri net representations, in this area, the CSNOs was proposed by Takai-Igarashi (2005), which gave the definition of the process of signaling. Based on the ontology, a Petri net can represent a signaling network without ambiguity by using a denotation, consisting of names of reactant, external states of reactant and components of a complex.

In the Incident or Hazard Command System's knowledge management, a large amount of poor, incomplete and inconsistent knowledge exists in the entire system, resulting in reducing the system performance. In order to reduce the brittle nature of traditional knowledge-based system and enhance the knowledge reusability and sharability, Liu (2006) proposed a methodology, called TTIPP, which built up a task ontology to represent task analysis and then systemically analyzed the process with IDEF0 and Petri nets.

A serious game, designed for the purpose of solving some problems, is actually a simulation of real-world events or processes, used in business training, education, health care, public policy and so on. For the purpose of evaluating the effectiveness of serious games, an approach for tracking a player's actions was described by Thomas *et al*. (2011), which utilized Petri nets to model and evaluate serious games. In addition, a domain and game action ontology, as a complement to Petri nets, was constructed to specify the actions of the learner, giving satisfactory results.

6 Summaries and future research directions

The knowledge sharing of Petri nets and ontologies has been an important topic of knowledge engineering. On one hand, ontology techniques can provide representation of and reasoning with knowledge for some domains, and on the other hand, Petri nets have a strong capability to model and analyze the dynamic behavior. Research has been conducted into various approaches to share knowledge between Petri nets and ontologies. Some of these techniques describe Petri net concepts and their relationships with ontology-based knowledge representations such as metamodel, ontology and ontology description language, and others concentrate on constructing Petri net models from ontologies. Petri nets and ontologies can be

viewed as complementary knowledge representation approaches used to configure new knowledge systems from existing knowledge.

Various knowledge sharing approaches between Petri nets and ontologies have been proposed and some major issues related to these approaches have been investigated. In this paper, we elaborate on recent research achievements in knowledge sharing between Petri nets and ontologies, and we provide a comprehensive literature overview of them, which can contribute to investigating knowledge sharing and reuse and, more importantly, serving as identifying the direction of the study of such knowledge sharing. The important issues on knowledge sharing between Petri nets and ontologies, which make full use of their respective advantages to represent knowledge-based systems and to realize interactive functions, are discussed in detail. In particular, we discuss the interactions between Petri nets and ontologies from two aspects, involving translating from Petri net to ontology and from ontology to Petri net. On one hand, we use metamodels, ontologies and ontology description languages such as *FOL*, *DL*, *OWL DL* and *OWL* to represent and reason about Petri nets. On the other hand, constructing Petri net models from *DDL*, *RDF*, *OWL-S*, *DMAL-S, FOL,* is discussed. In addition, the major applications of knowledge sharing between Petri nets and ontologies, covering BPM, Web services and some particular domains, is presented. In addition, we make some analyses in our whole review.

After surveying most of the methodologies of knowledge sharing between Petri nets and ontologies, it has been widely approved that sharing and reuse play a crucial role in knowledge-based systems. In order to achieve efficient and fair knowledge sharing and reuse, some methods, solving the interactions between Petri nets and ontologies, are proposed. However, the full potential of integrating Petri nets and ontologies has not been exhaustively explored. Therefore, there are several interesting directions of research that we are currently exploring. The following issues are a good starting point to find the appropriate techniques in many applications:

- *The more expressive Petri net models* may be needed to enhance the applicability of Petri net formalisms to other domain. For instance, extending Petri nets, including CPNs, time Petri nets and HLPNs, can represent more complex knowledge and share more rich knowledge. After extending Petri nets, the representation and reasoning of Petri nets with ontologies is still an important issue. On the other hand, applying the modeling and analyzing techniques of Petri nets to represent ontologies will also attract much attention.
- *The implementation of more expressive ontology reasoners or the enhancement of the reasoning ability of Petri nets with ontologies will be investigated in depth.* As shown in this paper, there have been several existing reasoning techniques such as *DL* (Lu *et al*., 2007), *SHOIN* (Hustadt *et al*., 2004), *SWRL* (Zhang *et al*., 2006) to reason about information that is contained in Petri nets, and some reasoning tasks may be done automatically by means of the existing ontology reasoners *DeLoean* (Bobillo & Delgado, 2007), and *FLORA-2* (Yang *et al*., 2003). But, most of them suffer some limitations and are not deep enough.
- *The implementation of Petri net execution in the context of Semantic Web service.* In order to support efficient integration and analysis of Web service applications, a Petri net model for Semantic Web service composition will attract much attention. Although there has been much research on applying Petri net models to compose Web services, integrating more expressive Petri nets into basic Petri net models is needed. Moreover, little research focuses on verifying the correctness of Web service composition by using analysis techniques of Petri nets. Thus, investigating the verification of Web service composition is very important issue.
- *The problem of inference in Petri net models* will become a hot research topic. As mentioned in two papers (Chuang *et al*., 1993; Liu, 1994), there has been some research on logical inference of predicate logic in Petri net models, but integrating more expressive *OWL DL* or *DL* into predicate logic is needed. It is necessary to transform *OWL DL* inference into Petri net models and use existing Petri net analysis methods such as T-invariants to deal with ontology inference.
- In order to handle imprecise and uncertain information that exists in real-world domains, *fuzzy extensions to Petri nets and ontologies and knowledge sharing between FPNs and fuzzy ontologies* will be an interesting topic in future research. As mentioned in Zhang *et al*. (2012), to represent more

complex knowledge, Petri nets and ontologies can be extended by using fuzzy set theory, and the use of fuzzy ontology approaches to represent FPNs is proposed. Furthermore, constructing FPN models from fuzzy ontologies will attract much attention. Thus, we can make full use of FPN analysis techniques to solve the interactive problems of FPNs and fuzzy ontologies.

- *The implementation of an automated tool achieving the interaction between Petri nets and ontologies.* This tool would be used to map Petri nets to ontologies or *OWL DL* format, or, in turn, ontologies to Petri nets. As shown in this paper, the interactions between Petri nets and ontologies have been investigated regarding the theoretical side, but there is no tool to implement these approaches. Thus, developing a tool to automatically interact with Petri nets and ontologies will attract much attention.

Acknowledgment

The work was supported in part by the National Natural Science Foundation of China (61370075, 61572118 and 61073139).

References

Angele, J., Kifer, M. & Lausen, G. 2009. *Ontologies in F-Logic*. International Handbooks on Information Systems. Springer, 45–70.

Araujo, M. & Roque, L. 2009. Modeling games with Petri nets. In *Proceedings of the 2009 Digital Games Research Association Conference*, London UK, September 2009.

Baader, F. D., Calvanese, D., McGuinness, D. L., Nardi, D. & Patel-Schneider, P. F. 2002. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Behrang, Z. J., Pornpit, W. & Farookh, K. H. 2010. Ontologies based approach in knowledge sharing measurement. *Journal of Universal Computer Science* **16**(6), 956–982.

Berners-Lee, T., Hendler, J. & Lassila, O. 2001. The Semantic Web. *Scientific American* **284**(5), 34–43.

Bezivin, J. 1998. Who's Afraid of Ontologies?. In *OOPSLA'98 Workshop: Model Engineering, Method and Tools Integration with CDIF*.

Billington, J., Christensen, S. & vanHee, K. 2003. The Petri nets Markup Language Concepts, Technology, and Tools. In *Proceedings of the 24th International Conference on Applications and Theory of Petri nets*, 483–505. Springer-Verlag.

Bobillo, F. & Delgado, M. 2007. DeLoean. `http://webdiis.unizar.es/~fbobillo/delorean.php`.

Borst, W. 1997. *Construction of Engineering Ontologies*. PhD thesis, Institute for Telematica and Information Technology, University of Twente.

Brachman, R. J. & Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers.

Breton, E. & Bezivin, J. 2001. Towards an understanding of model executability. In *Proceedings of the International Conference on Formal Ontologies in Information Systems*, 70–80.

Brockmans, S., Ehrig, M., Oberweis, A. & Sduder, R. 2006. Semantic alignment of business processes. In *Proceedings of the Eighth International Conference on Enterprise Information System*, 191–196.

Brogi, A., Corfini, S. & Iardella, S. 2007. From OWL-S descriptions to Petri nets. In *Proceedings of ICSOC 2007 Workshops*, 427–438.

Bunge, M. A. 1977. *Treatise on Basic Philosophy Volume 3: Ontologies I-the Furniture of the World*. Kluwer Academic Publishers.

Cardoso, J. 2007. The semantic web version: where are we? *IEEE Intelligent System* **22**(5), 84–88.

Chuang, L., Chaudhury, A., Whinston, A. B. & Marinescu, D. C. 1993. Logical inference of Horn clauses in Petri net models. *Journal of IEEE Transactions on Knowledge and Data Engineering* **5**(3), 416–425.

Clark, T., Sammut, P. & Willans, J. 2004. Application Modeling: A Foundation for Language Driven Development. `http://albini.xactium.com`.

Davis, R., Shrobe, H. & Szolovits, P. 1993. What is a knowledge representation? *AI Magazine* **14**, 17–33.

Dean, M., Schreiber, G., Bechhofer, S., van Harmelen, J., Helena, J. & Horrocks, I. 2004. OWL web ontologies language reference. *W3C recommendation*. `http://www.w3.org/TR/owl-ref`.

Deng, Y. & Chang, S. K. 1990. A G-net model for knowledge representation and reasoning. *IEEE Transaction on Knowledge and Data Engineering* **2**(3), 295–310.

Feng, X. N., Wang, Z. & Yin, G. S. 2008. Hierarchical Object-Oriented Petri net modeling method based on ontologies. In *Proceedings of 2008 International Conference on Internet Computing in Science and Engineering*, 553–556.

Feldmann, C. G. 1998. *The Practical Guide to Business Process Reengineering Using IDEF0*. Dorset House Publishing.

Gomez-Perez, A., Fernandez-Lopez, M. & Corcho, O. 2003. *Ontological Engineering. Advanced Information and Knowledge Processing*. Springer-Verlag.

Gasevic, D. & Devedzic, V. 2007. Interoperable Petri net models via ontologies. *International Journal of Web Engineering and Technology* **3**(4), 376–396.

Gasevic, D. & Devedzic, V. 2006. Petri net ontology. *Knowledge-Based Systems* **19**, 220–234.

Guarino, N. 1998. Formal ontologies and information systems. In *Proceedings of Formal Ontologies in Information System*, 3–15.

Guarino, N. & Giaretta, P. 1995. *Ontologies and knowledge bases: towards a terminological clarification. Towards Very Large Knowledge Bases Knowledge*. IOS Press, 25–32.

Guo-Yan, H. 2006. Analysis of artificial intelligence based Petri nets approach to intelligent integration of design. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, 1691–1695.

Grube, T. R. 1993. A translation approach to portable ontologies. *Knowledge Acquisition* **5**(2), 199–220.

Hansen, K. M. 2001. Towards a colored Petri nets profile for the unified modeling language – issues, definition, and implementation. *Internal Center for Object Technology Report COT/2-52*, University of Arhus. `http://www.daimi.au.dk/~marius/writings/cpn.pdf`.

Hustadt, U., Motik, B. & Sattler, U. 2004. Reducing SHIQ description logic to disjunctive datalog programs. In *Proceedings of the 9th International Conference on Knowledge Representation and Reasoning*, 152–162.

ISO/IEC 15909-1 2002. Software and systems engineering – high-level Petri nets. Part 1: concepts, definitions and graphical notation, Final Draft of the International Standard ISO/IEC 15909-1. ISO.

ISO/IEC 15909-2 2005. Software and systems engineering – high-level Petri nets. Part 2: transfer format, Working Draft of the International Standard ISO/IEC 15909-2. ISO.

Jozefowska, J. 2010. Knowledge representation for automated reasoning. In *Proceedings of KES-AMSTA 2010*, 6–11.

Jiang, L. X. & Li, H. W. 2009. An improved fault Petri nets for knowledge representation. In *Proceedings of the 2009 International Conference on Computational Intelligence and Software Engineering*, 1–4.

Kiefer, M., Lausen, G. & Wu, J. 1995. Logical foundations of object-oriented and frame-based languages. *Journal of ACM* **42**, 741–843.

Kogut, P., Cranefield, S. & Hart, L. 2002. UML for ontologies development. *The Knowledge Engineering Review* **17**(1), 61–64.

Koschmider, A. & Oberweis, A. 2005. Ontologies based business process description. In *Proceedings of the Open Interop Workshop on Enterprise Modeling and Ontologies for Interoperability*.

Kowalski, R. & Burton, A. 2011. WUENIC – a case study in rule-based knowledge representation and reasoning. In *JSAI-isAI Workshop*, 112–125.

Kummer, O., Wienberg, F. & Duvigneau, M. 2003. Renew – the reference net workshop. `http://renew.de/`.

Lam, S. W. & Srihari, S. N. 1991. Frame-based knowledge representation for multi-domain document layout analysis. In *Proceedings of the 1991 International Conference on Systems, Man, and Cybernetics. 'Decision Aiding for Complex Systems'* **1** (3), 1859–1864.

Li, X. & Lara-Rosano, F. 2000. Adaptive fuzzy Petri nets for dynamic knowledge representation and inference. *Expert Systems With Applications* **19**, 235–241.

Li, S. & Wang, S. Q. 2009. Geometry knowledge acquisition and representation on ontologies. In *Proceedings of the 2009 International Conference on Computational Intelligence and Software Engineering*, 1–4.

Liu, L. W. 1994. High-level Petri net model of logic program with negation. *IEEE Trans on Knowledge and Data Engineering* **6**(3), 382–395.

Liu, W. Y. 2006. Using IDEF0/Petri nets for ontologies-based task knowledge analysis: the case of emergency response for debris-flow. *Proceeding of the 39th Hawaii International Conference on System Sciences* **4**, 76c.

Lovrek, I. 1995. Petri nets based knowledge representation for intelligent networks. In *Proceedings of the 1995 IEEE International Symposium on Intelligent Control*, 602–607.

Lu, S. F., Sun, C. F. & Ma, X. J. 2007. Using e-connection and description logic for formalizing and analyzing high-level Petri nets. In *Proceedings of the 9th International Symposium on Symbolic and Numeri Algorithms for Scientific Computing*, 514–517.

Ma, B. X. & Xu, Y. L. 2009a. Integrating PNML with OWL for Petri nets. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology*, 228–230.

Ma, B. X. & Xu, Y. L. 2009b. Constructing Petri net model for dynamic description logic actions. *Proceedings of the 1st International Workshop on Education Technology and Computer Science* **3**, 753–756.

Martin, D. 2004. *OWL-S: Semantic Markup for Web Service. World Wide Web Consortium (W3C).* `http://www.w3.org/Submission/OWL-S/`.

Moldt, D. & Ortmann, J. 2004. DaGen: a tool for automatic translation from DAML-S to high-level Petri nets. In *Proceedings of the 7th International Conference of FASE 2004*, 209–213.

Murata, T. 1989. Petri nets: properties, analysis and applications. *Proceedings of the IEEE* **77**, 541–580.

Murata, T. & Zhang, D. 1988. A prediate-transition net model for parallel interpretation of logic program. *IEEE Transcations Software Engineer* **14**, 481–497.

Narayanan, S. & McIlraith, S. 2002. Simulation, verification and automated composition of web service. In *Proceedings of WWW'02*. ACM, 77–88.

Neches, R., Fikes, R. E. & Finin, T. 1991. Enabling technology for knowledge sharing. *AI Magazine* **12**(3), 36–56.

Nielsen, M. & Sassone, V. 1988. Petri nets and other models of concurrency. In *Lectures on Petri nets I: Basic Models*, Lecture Notes in Computer Science, **4546**, 587–642. Springer.

O'Leary, D. E. 2007. Knowledge representation of rules: a note. *Intelligent System in Accounting Finance and Management* **15**, 73–84.

Perez, A. G. & Benjamins, V. R. 1999. Overview of knowledge sharing and reuse components: ontologies and problem-solving methods. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods*.

Peterka, G. & Murata, T. 1989. Proof procedure and answer extraction in Petri net model of logic programs. *IEEE Transactions on Software Engineering* **15**(2), 209–217.

Petri, C. A. 1962. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress 62: Information Processing*, Popplewell, C. M. (ed.), North-Holland, 386–390.

Puppe, F. 1998. Knowledge reuse among diagnostic problem-solving methods in the shell-kit D3. *International Journal of Human-Computer Studies* **49**(4), 627–649.

Ranasinghe, R. A. C. & Madurapperuma, A. P. 2003. Enhanced frame-based knowledge representation for an intelligent environment. In *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration, and Engineering*, 25–30.

Recker, J. & Indulska, M. 2007. An ontology-based evaluation of process modeling with Petri nets. *Journal of Interoperability in Business Information System* **2**, 45–64.

Ribaric, S. & Hrkac, T. 2007. A knowledge representation and reasoning based on Petri nets with spatio-temporal tokens. In *Proceedings of International Conference on Computer as a Tool*, 793–800.

Ribaric, S. & Hrkac, T. 2012. A fuzzy spatio-temporal knowledge representation and reasoning based on high-level Petri nets. *Information Systems* **37**, 238–256.

Shanks, G., Tansley, E. & Weber, R. 2003. Using ontologies to validate conceptual models. *Communications of ACM* **46**(10), 85–89.

Soffer, P., Kaner, M. & Wand, Y. 2008. Assigning ontologies-based semantics to process models: the case of Petri nets. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering*, Lecture Notes in Computer Science **5074**, 16–31.

Staab, S. & Studer, R. 2004. *Handbook on Ontologies. International Handbooks on Information Systems*. Springer.

Studer, R., Benjamins, V. R. & Fensel, D. 1998. Knowledge and engineering: principles and methods. *Data and Knowledge Engineering* **25**, 161–197.

Strbac, P. 2002. *An Approach to Modeling Communication Protocol by Using Upgraded Petri Nets*. PhD dissertation, Military Academy.

Shen, V. R. L. 2006. Knowledge representation using high-level fuzzy Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics – Part A* **36**(6), 1220–1227.

Shi, Z. Z., Dong, M. K., Jiang, Y. C. & Zhang, H. J. 2005. A logical foundation for the semantic web. *Science in China Series F-Information Sciences* **48**(2), 161–178.

Takai-Igarashi, T. 2005. Ontologies based standardization of Petri net modeling for signaling pathways. *Silico Biology* **5**, 0047.

Tan, X. 2010. SCOPE: a Situation Calculus Ontologies of Petri nets. In *Proceedings of the Sixth International Conference on Formal Ontologies in Information Systems*, 227–240.

Thomas, P., Yessad, A. & Labat, J. M. 2011. Petri nets and ontologies: tools for the 'learning player' assessment in serious games. In *Proceedings of the 11th IEEE International Conference on Advanced Learning Technologies*, 415–419.

Vidal, J. C., Lama, M. & Bugarin, A. 2006. A high-level Petri net ontology compatible with PNML. *Petri Nets Newsletter* **71**, 11–23.

Vidal, J. C., Lama, M. & Bugarin, A. 2007. Petri nets semantics for OWL-S service choreography. In *Proceedings of the International Workshop on Formal Aspects of Business Processes and Web Service*, 7–20.

Vidal, J. C. & Lama, M. 2009. OPENET LD: an ontology-based Petri nets engine to execute IMS LD units of learning. In *Proceedings of the Ninth IEEE International Conference on Advanced Learning Technologies*, 499–503.

Vidal, J. C., Lama, M. & Bugarin, A. 2010. OPENET: ontologies-based engine for high-level Petri nets. *International Journal on Expert Systems with Applications* **1**(37), 6493–6509.

Wang, X. L. & Wu, X. L. 2012. A novel knowledge representation method based on ontologies for natural disaster decision-making. In *Proceedings of the 2012 IEEE International Conference on Computer Science and Automation Engineering*, 241–245.

Wang, Y. & Weber, R. 1993. On the ontological expressiveness of information systems analysis and design grammars. *Journal of Information Systems* **3**(4), 217–237.

Wang, Y., Bai, X., Li, J. & Huang, R. 2007. Ontologies-based test case generation for testing web services. In *Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems*, 43–50.

Weber, M. & Kindler, E. 2003. The Petri nets markup language. In *Advances in Petri nets. Petri nets technology for communication based systems*. Lecture Notes in Computer Science **2472**, 124–144. Springer-Verlag.

Yang, G., Kifer, M. & Zhao, C. 2003. FLORA-2: a rule-based knowledge representation and inference infrastructure for the semantic web. In *On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Lecture Notes in Computer Science **2888**, 671–688. Springer.

Yang, X. P. & Zhou, X. B. 2009. Ontologies-oriented Petri net model of semantics service composition. In *Proceedings of the 2nd International Workshop on Knowledge Discovery and Data Mining*, 932–936.

Yen, H. C. 2006. Introduction to Petri nets theory. *Recent Advances in Formal Languages and Applications* **25**, 343–373.

Yi, X. & Kochut, K. J. 2004. A CP-nets-based design and verification framework for web services composition. In *Proceedings of the IEEE International Conference on Web Services*, 756–760.

Yim, J., Joo, J. & Lee, G. 2011. Petri nets representation of ontologies for indoor location-based services. *GDC 2011*, CCIS **261**, 423–430. Springer-Verlag.

Yu, S. K., Hsu, W. & Pung, H. K. 1998. KPN: a Petri net model for general knowledge representation and reasoning. In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*, 184–189.

Zhang, F., Ma, Z. M. & Ribaric, S. 2011. Representation of Petri nets with OWL DL ontologies. In *Proceedings of the 8th International Conference on Fuzzy Systems and Knowledge Discovery* **3**, 1396–1400.

Zhang, F., Ma, Z. M. & Yan, L. 2012. A fuzzy ontologies approach for representing fuzzy Petri nets. In *Proceedings of WCCI 2012 IEEE International Conference on Computational Intelligence*, 10–15.

Zhang, G., Meng, F., Jiang, C. & Pang, J. 2006. Using Petri nets to reason with rule and owl. In *Proceeding of the 6th IEEE International Conference on Computer and Information Technology*.

Zhang, X., Chen, X. Y., Guo, S. Y. & Zhan, Z. Q. 2010. Ontologies-based ITSM knowledge representation research. In *Proceedings of the 2010 International Conference on Advanced Intelligence and Awareness Internet*, 230–235.