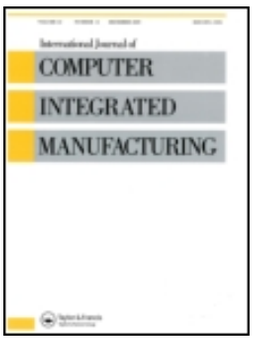

International Journal of Computer Integrated Manufacturing

ISSN: 0951-192X (Print) 1362-3052 (Online) Journal homepage: https://www.tandfonline.com/loi/tcim20

An ontology-based architecture for implementing semantic integration of supply chain management

Yan Ye , Dong Yang , Zhibin Jiang & Lixin Tong

To cite this article: Yan Ye , Dong Yang , Zhibin Jiang & Lixin Tong (2008) An ontology-based architecture for implementing semantic integration of supply chain management, International Journal of Computer Integrated Manufacturing, 21:1, 1-18, DOI: 10.1080/09511920601182225

To link to this article: https://doi.org/10.1080/09511920601182225

International Journal of Computer Integrated Manufacturing, Vol. 21, No. 1, January – February 2008, 1 – 18

Taylor & Francis
Taylor & Francis Group

An ontology-based architecture for implementing semantic integration of supply chain management

YAN YE†‡, DONG YANG*†, ZHIBIN JIANG† and LIXIN TONG†

†Department of Industrial Engineering and Management, School of Mechanical Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Road, Min Hang District, 200240, Shanghai, PR China
‡College of Mechanical Engineering, Zhejiang University of Technology, 310032, Hangzhou, Zhejiang, PR China

Efficient supply chain management (SCM) requires consistent exchange and sharing of information semantics, which is often hindered by semantic clashes among heterogeneous applications. This paper proposes an ontology-based architecture for addressing the problem of semantic integration. A supply chain ontology (SCO) is developed by using the skeletal method to capture concepts and relationships common to SCM. Moreover, it is formally defined in OWL DL, a web-based ontology language, to serve as an interlingua for an architecture enabling information integration of SCM. Syntactic transformation between XML Schema and OWL in the architecture is implemented by syntactic translators that are developed by using XSLT. Furthermore, a rule-based approach is presented to map semantically similar terminologies between SCO and application ontologies with the aim of enabling semantic interoperability among applications in supply chains. The mapping rules are represented by SWRL, a rule description language that is independent of the rule languages specific to inference engines. In addition, a case study of a printer supply chain is illustrated to demonstrate the proposed methodology. Finally, a prototype system is developed using the Java platform to implement the SCO-based approach to semantic integration of SCM.

Keywords: Supply chain management; Ontology; Semantic integration; Interoperability

1. Introduction

Supply chain management (SCM) has become an important operational strategy for enhancing organizational competitiveness and responding flexibly to changing markets. It facilitates a supply chain (SC), composed of geographically distributed enterprises in different business sectors, to efficiently transform raw materials into products and to deliver products and relevant services to customers at the right time and at the right place (Ulieru and Cobazru 2005). The purpose of SCM is to integrate key business processes from end users to original suppliers that provide value-added products, services and information for customers and other stakeholders (Lambert and Cooper 2000). A key factor for the successful implementation of such integration is effective information sharing and interoperability across collaborative supply chain partners, not only at the syntactic level but at the semantic level.

Extensive use of the internet and world wide web (WWW) has provided a web-based mechanism for supply chain operations and management to enable immediate communication among partners. Furthermore, most enterprise application integration standards have adopted extensible markup language (XML) technologies to support information integration of web-based SCM systems. XML standardizes the syntax of information exchange by defining markups and structures of documents using tags. However, XML documents fail to capture the semantics (meanings) of the data (Ray and Jones 2003). It is presumably reasonable that the XML-based standards can

*Corresponding author. Email: dongyang@sjtu.edu.cn

International Journal of Computer Integrated Manufacturing
ISSN 0951-192X print/ISSN 1362-3052 online © 2008 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/09511920601182225

well be applied in a relatively closed supply chain environment where several partners use the agreed vocabularies to describe their business data in terms of the same XML-based standard. Nevertheless, this assumption cannot hold for a web-based SCM system that is characterized by open-ended and ever-changing partners. In such a web-based open setting, business partners are dynamically discovered through the web and combine to form a supply chain in a relatively short period for a specific business objective, which is dissolved when the objective has been achieved. These partners normally do not use the same standard terms with the same semantics, which results in semantic mismatches between exchanged information affecting the realization of effective SCM, especially during the short period when the supply chain exists. Therefore, the problem of semantic integration, especially caused by inconsistent terms and semantic mismatches among different applications, is still an issue that needs to be solved for efficient and complete supply chain integration.

Enterprises dynamically participating in supply chains own heterogeneous applications and legacy systems, developed independently with different knowledge modelling schemata. These systems may use different terms and vocabularies to represent the same entities and concepts. For example, the concept *activity* may be represented by the term *operation* or *task*. Even when they use the same terminology, they often associate different semantics with the term to denote different concepts. Furthermore, the semantics of each term and vocabulary may not be adequately defined or unambiguously explained by the applications. Such inconsistent terms and semantic clashes impair the meaningful knowledge representation and semantic interoperability that are essential to supply chain integration in the web environment. Since the ontology defines precise semantics of shared terminologies describing domain knowledge (Studer *et al.* 1998), an ontology-based architecture is presented in this paper to address the problem of semantic integration of supply chains. Moreover, considering that information in web-based SCM systems is normally exchanged through the internet and WWW, this architecture is enhanced by using semantic web technologies (Antoniou and van Harmelen 2004) to meet the requirement of semantic interoperability of information on the web. More concretely, a supply chain ontology (SCO) is proposed to provide well-defined and formal representation of concepts and relationships common to SCM and therefore to act as an interlingua for the application integration architecture enabling syntactic and semantic interoperability across various applications of supply chain members. At the same time, the use of the semantic web technologies, especially OWL Web Ontology Language (OWL) (Horrocks *et al.* 2003) and Semantic Web Rule Language (SWRL) (Horrocks *et al.* 2004), provides sufficient expressive power and efficient reasoning support for the proposed architecture realizing the web-based supply chain integration.

The remainder of this paper is organized as follows. The next section describes current efforts related to ontologies and semantic integration of SC domains. The development of SCO is then explained in section 3. The ontology is captured by using Protégé (Gennari *et al.* 2003) and is encoded in the OWL language. Section 4 discusses the application integration framework for supply chain integration and presents methods of enabling semantic integration among supply chain applications through the use of the SCO model. In section 5, the architecture of a prototype system implementing the SCO-based integration framework is described. Finally, conclusions are given in section 6.

2. Ontology and semantic integration

The term ontology comes from philosophy, where it is a systematic account of existence of beings in the world. It has been adopted by the artificial intelligence (AI) community to describe the knowledge of a domain in a declarative formalism and in a machine-understandable way. An ontology is a formal, explicit specification of a shared conceptualization (Studer *et al.* 1998). Typically, the ontology is composed of terms representing domain entities, their semantics and formal axioms. It provides a shared and common understanding of a domain that can be communicated between people and heterogeneous applications, thus facilitating semantic interoperability, sharing and reuse of knowledge among information systems (Pinto and Martins 2004). Ontologies can be combined with the current supply chain network to form a new semantic supply chain network. In the current networks, systems in different enterprises have been able to exchange information rapidly through the internet, while in semantic supply chain networks, these systems will interoperate all the information, such as products, services, enterprises, business documents and processes, based on the semantics provided by ontologies, which helps to enable seamless information integration.

Several research efforts are focused on issues relevant to ontologies and information integration in SCM domains. The Enterprise Project (Uschold *et al.* 1998) has developed a semi-formal enterprise ontology to serve as a basis of an integrated framework for enterprise modelling. The ontology mainly covers a collection of terms and definitions relevant to a single enterprise and does not lay emphasis on information integration among interacting enterprises in supply chains. The TOVE project (Fox and Gruninger 1998) has created ontologies to provide shared terminologies for the enterprise and defined the semantics of each term using first-order logic. This project is dedicated to automatically deducing answers to many commonsense

questions about the enterprise. However, it does not cover the problem of semantic integration between heterogeneous systems in detail. The Standard for the Exchange of Product Data (STEP), developed by the International Organization for Standardization (ISO), aims to provide a basis of exchanging and sharing product information through the standardization of computer-interpretable representation of manufacturing product data (Owen 1993). It focuses only on the pure translation of terminologies from one system to another and therefore fails to map semantics related to product information in supply chains. The Process Specification Language (PSL) (Grüninger 2004) defines a neutral representation of discrete manufacturing process and develops a general ontology to facilitate information exchange among multiple process-related applications. It concentrates on semantic interoperability of manufacturing process information, which restricts itself to the manufacturing processes in SCM.

In addition, several studies are concerned with the development of standard business document frameworks for supply chain integration. The Universal Business Language (UBL), developed by the Organization for the Advancement of Structured Information Standards (OASIS), defines a generic XML interchange format of basic business documents that can be extended to meet the requirements of particular enterprises in supply chains (OASIS 2004). Specifically, current version 1.0 of UBL provides document types and business rules for supporting a typical order-to-invoice procurement process. The Open Application Group Integration Specification (OAGIS), developed by the Open Application Group, Inc. (OAGi), presents a common horizontal Business Object Document (BOD) standard based on the XML technologies to achieve interoperability between disparate systems, disparate companies and disparate supply chains (OAGi 2006).

Most researches described above are focused on particular aspects of SCM and do not consider the characteristics of the web environment where currently supply chains operate. Even though UBL and OAGIS are based on XML, they still have difficulty in completely supporting semantic integration of information in SCM because XML only imposes constraints on the structure of documents and thus represents documents at the syntactic level (Antoniou and van Harmelen 2004). As a result, these XML-based standards for supply chain integration fail to convey the semantics of documents. In contrast, OWL and SWRL used in the proposed SCO model and integration architecture have both well-defined syntax and formal semantics, which provides enough semantic expressiveness and reasoning capabilities for integrating information in SCM.

In addition to information integration considered by the above literature, business process integration is another important issue in the web-based SCM systems, which involves precise modelling, automatic discovery, composition, execution and monitoring of business processes within and between enterprises. Many researchers and organizations have made a meaningful study of business workflow integration. Shen *et al*. (2004) presented an integrated business process modelling approach combing IDEF0, IDEF3 and DFD for an e-business system. Vasiliu *et al*. (2004) implemented business negotiation between multiple machines on behalf of different partners as a semantic-enabled web service to automate the negotiation process. With the development and application of the semantic web and web services technologies, semantic web services have become a key technology for enabling the workflow integration in supply chains. A significant emerging standardization effort in this field is the Web Service Modelling Ontology (WSMO) (de Bruijn *et al*. 2005a, Feier and Domingue 2005, Roman *et al*. 2005) approach based on the Web Service Modelling Language (WSML) (de Bruijn *et al*. 2005b). WSMO is a meta-ontology for describing all relevant aspects of semantic web services in a unified manner that is formalized by WSML, a family of formal description languages consisting of five language variants based on different logical formalisms for offering different levels of expressiveness. The approach defines four main modelling elements, namely *Ontologies*, *Web services*, *Goals* and *Mediators*. Among these, *Ontologies* provide the terminology used by other WSMO elements. The *Web services* element defines the functional and behavioural aspects provided by web services while the *Goals* element specifies user desires that can be satisfied by executing web services. More importantly, WSMO provides the mediation mechanism through the *Mediators* element for dealing with interoperability problems between different WSMO elements and defines four different types of Mediators: *ooMediators*, *ggMediators*, *wgMediators* and *wwMediators*. Among them, *ooMediators* allow any WSMO element to import ontologies by resolving all the representation mismatches between ontologies. In addition, the Web Service Execution Environment (WSMX) is developed to provide a reference architecture and implementation for enabling the dynamic discovery, selection, mediation, invocation and interoperation of semantic web services based on the WSMO specification (Haller *et al*. 2005, Zaremba *et al*. 2005). In particular, the Data Mediator component in the WSMX architecture implements the functionality of the *ooMediators* by reconciling heterogeneous data expressed in terms of their own ontologies during the web services automation (Mocan and Cimpian 2005).

WSMO, WSML and WSMX aim to facilitate the automation of semantic web services for workflow integration. Like OWL, WSML-based WSMO provides meta-ontology constructs, such as *concepts*, *relations* and *axioms*, for describing domain knowledge and can therefore act as a

general ontology language for the construction of domain ontologies. However, different from OWL, WSMO puts much emphasis on modelling web services applications. In addition, WSMO and WSMX provide the general implementation mechanism of ontology mediation for the discovery, selection and composition of web services. In this research, the emphasis is on the information integration for efficient SCM, and especially on the syntactic and semantic mediation between business information including documents. In addition, OWL (Horrocks *et al*. 2003) has become a widely used ontology language for the semantic web proposed by the World Wide Web Consortium (W3C) organization and a growing number of automated tools are available to formally process OWL documents. Furthermore, SWRL (Horrocks *et al*. 2004), a semantic web rule language closely associated with OWL, is independent of the rule languages internal to inference engines, which is advantageous to the free configuration of specific engines for performing the reasoning based on the SWRL rules. Therefore, based on these characteristics and advantages, OWL and SWRL are chosen for the proposed SCO-based supply chain integration approach. The presented SCO model is specially geared toward the supply chain domain, which is different from WSMO. Although the implementation architecture in the approach contains similar modules to the WSMX model, the WSMX model only offers the high-level modular structure of the implementation system. By comparison, more specific software tools, such as XSLT-based translators and SWRL2JESS, are constructed in the proposed architecture for syntactic and semantic transformations between applications in the real-world supply chain setting. In addition, compared to the B2B integration technology architecture proposed by Bussler (2003), this research particularly emphasizes the implementation details of the semantic integration framework.

3. Supply chain ontology (SCO) model

The SCO model developed in this work used the skeletal methodology presented by Uschold and King (1995). The methodology includes the following four stages: identify the purpose of the ontology; build the ontology; evaluate it; and document it. The building activity is further decomposed into three activities of capturing the ontology, coding it and integrating other ontologies inside the current one. This section describes main development activities and gives a global view of SCO.

3.1. Identifying the purpose of SCO

The purpose of SCO is to serve as an interlingua to enable information integration across interacting applications in supply chains. Hence, the model should be able to represent concepts and relationships common to all the applications in supply chains. In other words, it cannot limit itself to the representation of information specific to certain application in a specific enterprise. In general, different types of enterprises in supply chains exchange and share relevant information, manufacture products, and delivery products and services to customers. Therefore, SCO needs to capture general information, such as business objects and organizations, business activities and business documents. In addition, as different enterprises dynamically participate in the supply chains of specific domains and interact with each other, SCO should be extensible to support the representation of additional information and semantics of specific application domains.

3.2. Capturing SCO

Ontology capture includes the identification of terms referring to key concepts and relationships in the domain of interest and the production of their unambiguous text definitions. These modelling activities are supported by Protégé (Gennari *et al*. 2003), a platform for constructing domain knowledge models and knowledge-based applications. Protégé provides a suite of tools to support the creation, visualization and manipulation of ontologies in various representation formats. Among these tools, the Protégé-OWL plug-in (Knublauch *et al*. 2004) is used to capture the classes and their relations of SCO in OWL for the semantic supply chain networks and the OntoViz plug-in (Sintek 2006) is adopted to visualize them with the help of Graphviz graph drawing software.

The top-level basic classes in the SCO model include *Supply_Chain*, *SC_Structure*, *Party*, *Role*, *Purpose*, *Activity*, *Resource*, *Transfer_Object*, *Performance*, and *Performance_Metric*, as shown in figure 1 displayed using the OntoViz plug-in. In figure 1, boxes represent classes and arrows express relations between the corresponding classes. In particular, the arrow labelled with the relation name *isa* represents the subclass relationship, that is, the class at the tail of the arrow is a subclass of the class at the head of the arrow. An asterisk (*) placed after a relation name labelling an arrow denotes the multiplicity of the relation, that is, the relation occurs between an instance of the class corresponding to the tail of the arrow and multiple instances of the class corresponding to the head of the arrow. Each class is further classified into a class hierarchy to describe main characteristics of SCM systems in detail. Although the purpose of this study is to develop a neutral representation language SCO enabling semantic integration of SCM systems, this section only deals with the descriptions of basic classes and relations for the consideration of the length and scope of the paper.

A supply chain is a dynamic network that is composed of some parties sharing the common purposes and contains different types of activities performed by these parties.

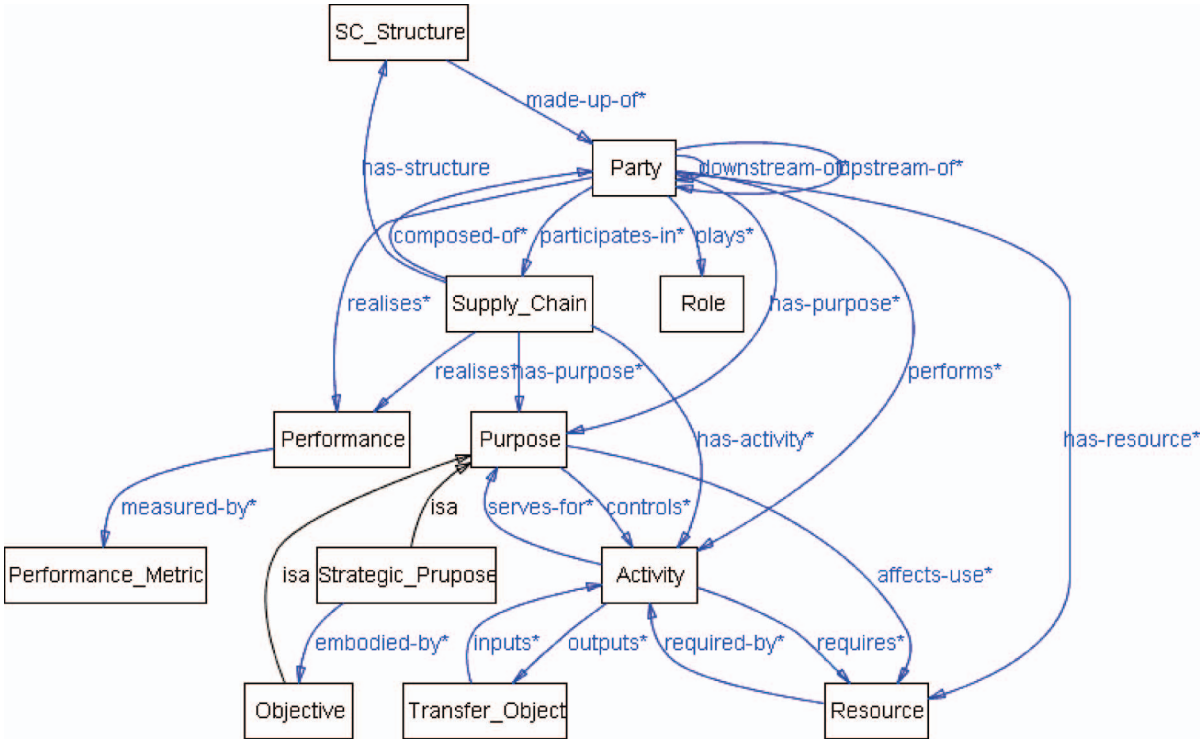

Figure 1. Top-level basic classes and relations in the SCO model.

The *SC_Structure* class refers to a set of supply chain structures that represent the upstream and downstream links and the supplier-buyer relationships between parties in supply chains (Huang *et al*. 2003). The class can be specialized into five subclasses: *Dyadic_Structure*, *Serial_Structure*, *Divergent_Structure*, *Convergent_Structure* and *Network_Structure*. The dyadic structure is made up of two enterprises, e.g. buyer and vendor. The serial structure results from cascading several dyadic structures. A typical serial supply chain usually contains retailers, distributors, manufacturers and suppliers. Both the divergent and convergent structures are modified serial structures. In a divergent structure, one supplier distributes products to several downstream enterprises. In a convergent structure, several components and materials provided by upstream enterprises are assembled by a manufacturer. A network structure, a combination of the convergent and divergent structures, represents a complex supply chain.

A party is a legal entity participating in supply chains. It is an important concept because resources, activities, capabilities, purposes and performances of supply chains depend on each individual party. The Enterprise Ontology (Uschold *et al*. 1998) defines the party as the union of *Person*, *Corporation* and *Partnership* and focuses on the single enterprise environment. In contrast, a supply chain is an integrated multi-enterprise environment and thus the explicit representation of business capabilities of the single enterprise and the entire supply chain is necessary. Therefore, the SCO model defines the supply chain as the aggregation of the *Party* objects. Each party is a downstream entity or an upstream entity of other parties in the chain and plays different roles, such as *Supplier*, *Manufacturer*, *Retailer*, *Forwarder*, *Vendor* and *Customer*, in terms of its capacity and purpose.

The *Purpose* class represents the knowledge that can direct SCM and enterprises operations and affect decisions related to the dynamic configuration of supply chain structures, the use of resources and the execution of activities. The class is specialized into the subclasses *Strategy* and *Objective*. Generally, a strategy is embodied by several objectives to more concrete guide activities and resources. For example, knowledge about the objective of assuring the delivery date of each order may influence the choice of machines for a specific batch of products.

An activity is something done over a particular time interval (Uschold *et al*. 1998) that requires certain resources to satisfy given purposes. According to PSL, activities include primitive activities, atomic activities and complex activities. A primitive activity has no proper sub-activities. An atomic activity is concurrent superposition of a set of primitive activities and a complex activity is an arbitrary combination of activities that are sub-activities of the activity. It should be noted that the concept of process is not explicitly defined in SCO because a business process is

virtually an arrangement of activities that are related through hierarchical or logic connection relationships. Business processes, in this sense, can normally correspond to complex activities.

The *Resource* class is an important part of the capabilities of parties and supply chains, representing a support mechanism for the execution of activities. The class has wide scope and various types. At the higher level of abstraction, it may consist of the subclasses *Production_Resource*, *Storage_Resource*, *Transportation_Resource* and *Human_Resource*. More concretely, resources can refer to a certain type of machines, materials and employees.

The *Transfer_Object* class describes a set of business objects that flow through activities in supply chains. These objects are inputted into activities and are exported through transformation behaviour of the activities. The class can be classified into three subclasses: *Material_Object*, *Capital_Object* and *Information_Object*. Here, the class *Material_Object* further contains its subclasses: *Material*, *Product* and *Service*. In the Enterprise Ontology, the concept of service is implicitly represented by the *Product* class, while the *Service* class is explicitly defined in SCO, due to the fact that services have played an increasingly important role in the environment of information economy. Moreover, different services can be combined with the same type of products to achieve different performances. For example, 24-hour after service for printers may produce higher customer satisfaction than 8-hour after service. The class hierarchy of the class *Transfer_Object* is shown in figure 2.

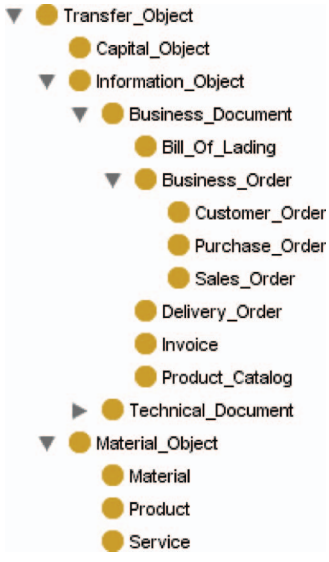

Figure 2. The class hierarchy of the *Transfer_Object* class.

The *Performance* class represents achievements that entities obtain through activities. It contains two levels: supply chain performance and party performance. The former depends on the latter. Moreover, both can be measured by several performance indexes. The Supply Chain Operations Reference (SCOR) model developed by the Supply Chain Council (2005) provides a basis for describing the knowledge of supply chain performance in SCO. The model contains five performance attributes of supply chains that are represented by the classes *SC_Reliability*, *SC_Responsiveness*, *SC_Flexibility*, *SC_Costs* and *SC_Assets*. Each attribute is measured by the corresponding metric classes for operational levels. For example, the *SC_Responsivenes* attribute is associated with the top-level metric class *Order_Fulfillment_Cycle_Time*.

3.3. Coding SCO

Ontology coding means the explicit representation of the conceptualization captured above in some formal language. Typically, the languages representing ontologies can be grouped into two major categories (Corcho *et al.* 2003). The first category consists of several ontology languages that are derived from research efforts of knowledge representation paradigms in the AI community and are not designed for use in the web, such as KIF, Ontolingua, OCML, FLogic and Loom. The second category is usually called the web-based ontology language, including RDFS, SHOE, OIL, DAML, DAML + OIL and OWL. The first five languages are all predecessors of OWL (Horrocks *et al.* 2003). In this study, OWL is used to encode the developed SCO based on the following facts. First, OWL is a web-oriented ontology language that exploits the characteristics of the web and thereby, compared with the first language category, is more suitable for web-based SCM systems because the systems operate on the internet and WWW. Secondly, it is quite a sophisticated language that, on the one hand, shares useful language constructs and design features with its predecessors whilst maintaining maximum compatibility with them and, on the other hand, adds desirable features through appropriate extensions to satisfy a large number of sometimes conflicting requirements, such as the trade-off of expressive power and inference support. Finally, OWL adopts a description logic (DL) (Baader *et al.* 2003) style model theory to formalize the meaning of the language, which allows ontologies, and information using vocabulary defined by ontologies, to be shared and exchanged with consistent and precise semantics. Another advantage of the feature is that automated reasoning techniques can be used to compute classification hierarchy and to check the consistency of ontologies (Horrocks *et al.* 2003).

3.3.1. Overview of the OWL language. OWL (Horrocks *et al*. 2003) is an ontology language for the semantic web developed by the W3C organization and has three increasingly expressive sublanguages—OWL Lite, OWL DL and OWL Full. Furthermore, OWL has both a frame-like abstract syntax and a normative RDF/XML exchange syntax. The former is an easy-to-read syntax that abstracts from and is less general than the latter for facilitating access to and evaluation of ontologies. Generally, OWL ontologies are stored as web documents written in RDF/XML. In addition, OWL is provided with formal model–theoretic semantics. Specifically, the semantics of OWL DL is very similar to that of DL, and even some definitions using OWL DL language constructs have the same semantics as the corresponding DL axioms, as shown in table 1. The semantics is given by means of the interpretation $\mathtt{I} = (\Delta^{\mathtt{I}}, \bullet^{\mathtt{I}})$, where $\Delta^{\mathtt{I}}$ is a non-empty set representing the domain of individuals in a model and $\bullet^{\mathtt{I}}$ is an interpretation function, which maps the class name C and the property name P to a subset $C^{\mathtt{I}}$ of $\Delta^{\mathtt{I}}$ and a binary relation $P^{\mathtt{I}}$ over $\Delta^{\mathtt{I}}$, respectively, and is extended to class definitions and property axioms in table 1.

3.3.2. Coding SCO in OWL DL. The key classes and relations of SCO in figure 1 can be described by class axioms and property axioms in OWL DL. For instance, the *Party* class is formally defined in OWL DL RDF/XML syntax as follows.

```
<owl:Class rdf:ID="Party">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:ID="Person"/>
        <owl:Class rdf:ID="Corporation"/>
        <owl:Class rdf:ID="Partnership"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="participates-
        in"/>
      </owl:onProperty>
      <owl:allValuesFrom>
        <owl:Class rdf:ID="Supply_Chain"/>
      </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  ......
</owl:Class>
```

Here, *Thing* is a built-in class predefined in OWL and represents a super-class of all OWL classes. Therefore, the class *Party* and other classes in the SCO model are the subclasses of the class. Moreover, these definitions in RDF/XML exchange syntax can be written in OWL DL abstract syntax shown in table 1, expressing the same semantics. The rewriting results are:

SubClassOf (*Party* owl:Thing)
SubClassOf (*Party* unionOf (*Person Corporation Partnership*))
SubClassOf (*Party* restriction (*participates-in* allValuesFrom (*Supply_Chain*)))

In addition, as OWL DL corresponds to DL, the above definitions can be expressed by the following equivalent axioms in DL:

Axiom 1. The *Party* class is a subclass of the class *Thing* and includes things that are *Person*, *Corporation*, or *Partnership*. Moreover, all values of the property *participates-in* of the *Party* class are members of the class *Supply_Chain*.

Table 1. OWL DL abstract syntax and semantics.

OWL DL Abstract Syntax	DL Syntax	Semantics
unionOf (C_1 C_2)	$C_1 \sqcup C_2$	${C_1}^{\mathtt{I}} \cup {C_2}^{\mathtt{I}}$
intersectionOf (C_1 C_2)	$C_1 \sqcap C_2$	${C_1}^{\mathtt{I}} \cap {C_2}^{\mathtt{I}}$
complementOf (C)	$\neg C$	$\Delta^{\mathtt{I}} \setminus C^{\mathtt{I}}$
restriction (P allValuesFrom (C))	$\forall P.C$	$\{x \mid \forall y.(x, y) \in P^{\mathtt{I}} \rightarrow y \in C^{\mathtt{I}}\}$
restriction (P someValuesFrom (C))	$\exists P.C$	$\{x \mid \forall y.(x, y) \in P^{\mathtt{I}} \wedge y \in C^{\mathtt{I}}\}$
restriction (P minCardinality (n))	$\geq n\ P$	$\{x \mid \lvert\{y \mid (x, y) \in P^{\mathtt{I}}\}\rvert \geq n\}$
restriction (P maxCaidinality (n))	$\leq n\ P$	$\{x \mid \lvert\{y \mid (x, y) \in P^{\mathtt{I}}\}\rvert \leq n\}$
SubClassOf (C_1 C_2)	$C_1 \sqsubseteq C_2$	${C_1}^{\mathtt{I}} \subseteq {C_2}^{\mathtt{I}}$
DisjointClasses (C_1 C_2)	$C_1 \sqcap C_2 \equiv \bot$	${C_1}^{\mathtt{I}} \cap {C_2}^{\mathtt{I}} = \{\}$
EquivalentClasses (C_1 C_2)	$C_1 \equiv C_2$	${C_1}^{\mathtt{I}} = {C_2}^{\mathtt{I}}$
ObjectProperty (P super (P_1)	$P \sqsubseteq P_1$	$P^{\mathtt{I}} \subseteq {P_1}^{\mathtt{I}}$
domain (C_1)	$\geq 1\ P \sqsubseteq C_1$	$P^{\mathtt{I}} \subseteq {C_1}^{\mathtt{I}} \times \Delta^{\mathtt{I}}$
range (C_2)	$\top \sqsubseteq \forall P.C_2$	$P^{\mathtt{I}} \subseteq \Delta^{\mathtt{I}} \times {C_2}^{\mathtt{I}}$
[inverseOf (P_2)]	$P \equiv {P_2}^-$	$P^{\mathtt{I}} = {P_2}^{\mathtt{I}-}$
[Symmetric]	$P \equiv P^-$	$P^{\mathtt{I}} = P^{\mathtt{I}-}$
[Functional])	$\top \sqsubseteq \leq 1P$	$P^{\mathtt{I}}$ is functional

$$Party \sqsubseteq Thing \sqcap (Person \sqcup Corporation \sqcup Partnership) \sqcap \forall\ participates\text{-}in.\ Supply\_Chain$$

Similarly, the following axioms in DL describe characteristics of relevant classes and properties in SCO:

Axiom 2. The classes *Person*, *Corporation* and *Partnership* are mutually disjoint. Moreover, the class *Corporation* is equivalent with the class *Company*.

$$Person \sqcap Corporation \sqcap Partnership \equiv \bot$$
$$Corporation \equiv Company$$

Axiom 3. The object property *participates-in* is an inverse of the *composed-of* object property.

$$participates\text{-}in \equiv composed\text{-}of^{-}$$

Axiom 4. The object property *has-structure* is a functional property.

$$\top \sqsubseteq \leq 1\ has\text{-}structure$$

Definitions of key classes and properties in SCO represented by OWL DL are given in Appendix 1. As the SCO model is captured through the use of the Protégé-OWL plug-in, all the formal definitions are created in the form of an OWL document by Protégé. The OWL document can be manipulated by Jena, a Java framework offering a programmatic environment for RDF and OWL documents (Jena 2005). In particular, the Jena APIs can easily integrate an appropriate OWL reasoner, such as Racer (Haarslev and Möller 2003), to automatically perform reasoning on the developed SCO and the corresponding document instances, such as consistency checking, subsumption testing and instance classification.

4. Syntactic and semantic integration based on the SCO model

SCO enables a general formal representation of information in SCM systems, thereby providing semantic foundation for application integration. A SCO-based supply chain application integration framework is shown in figure 3.

SCO defines basic concepts and relations common to SCM, which can be instantiated or specialized to create new proper concepts and relations for describing application knowledge of specific supply chains. Furthermore,

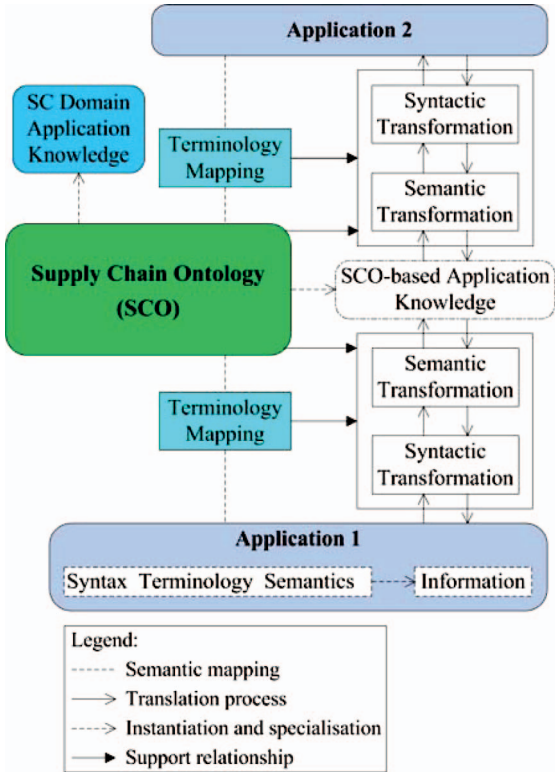

Figure 3. The SCO-based application integration framework.

SCO enables data sharing and integration between heterogeneous applications in supply chains. Typically, point-to-point translation programs for each pair of applications can enable communication from one to another (Ciocoiu *et al*. 2001). However, as there are many dynamically changing applications in SCM systems and the exchanged information has been more complex with the requirement of the resolution of semantic clashes, developing a large number of translators for communication in SCM systems is rather difficult. As a neutral interchange language, the SCO model provides a meaningful means for addressing the problems and for enabling semantic and syntactic integration. The integration framework in figure 3 shows a hub-and-spoke solution, in which SCO acts as a central interchange point for interacting applications in supply chains. In other words, only semantic mappings and transformations between individual application and SCO are necessary, instead of those between any pair of applications in supply chains. Thus, when there are n different applications, the point-to-point approach requires the development of $n*(n-1)$ translation programs, two for each pair enabling bidirectional translations, while the SCO-based integration approach needs the construction of $n*2$ translators. Therefore, the number of translation programs can be drastically reduced when the number of interacting applications is large.

It is assumed that each application in supply chains has its own predefined terminology, syntax and semantics associated with its ontology. In figure 3, typical steps of supply chain integration through the mapping between applications and SCO are as follows.

- *Terminology mapping*. Semantic mapping relationships between terminologies of application ontologies in supply chains and those of SCO are specified.
- *Syntactic transformation*. The information formats of applications are translated into SCO syntax, namely, RDF/XML syntax. The step does not change terminologies and indeed changes the way in that terminologies are combined. In other words, information in an application is represented in SCO syntax without the modification of its semantics.
- *Semantic transformation*. Based on the inference rules of ontology mapping, semantic exchanges between applications and SCO are realized. Thus, information represented by application terminologies, semantics and SCO syntax is translated into the representation by using terminologies, semantics and syntax of SCO.

4.1. SCO-based representation for specific supply chain domains

To illustrate the ability to represent information of specific application domains through the use of SCO, a printer supply chain (PSC) scenario shown in figure 4 is used. For the sake of simplicity, only two suppliers (TEEC and SHME), one manufacturer (YSH_PRINTER_CO), two distributors (AM_D and AS_D) and three retailers (EU_R, AM_R and AS_R) are considered. The manufacturer YSH_PRINTER_CO is responsible for main manufacturing process of printers (denoted by YSH_MMP), which includes two activities, namely, the printed circuit board assembly and test (PCBAT) and the final assembly and test (FAT). The PCBAT activity processes EC_1 electronic parts provided by an upstream corporation TEEC to produce printer head drive boards (PH_DRIVE_BOARD). These boards and EC_2 parts from the other upstream corporation SHME are inputted into the FAT activity that assembles and tests printers. These statements in the scenario can be represented by terminologies and their semantics of SCO in the form of DL assertions as follows.

Axiom 5. The instance *PSC* of the *Supply_Chain* class has the structure *SERIAL1* that is an instance of the *Serial_Structure* class.

PSC: Supply_Chain
SERIAL1: Serial_Structure
(PSC, SERIAL1): has-structure

Axiom 6. *PCBAT*, *FAT* and *YSH_MMP* are the instances of the *Activity* class and the first two instances are sub-activities of the last one.

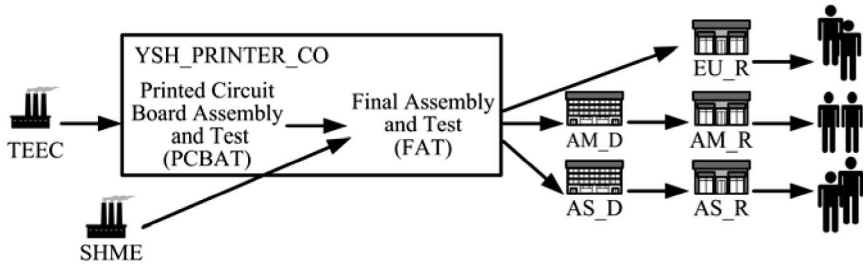

Figure 4. A printer supply chain scenario.

PCBAT: Activity
FAT: Activity
YSH_MMP: Activity
(FAT, YSH_MMP): subactivity
(PCBAT, YSH_MMP): subactivity

Axiom 7. *PH_DRIVE_BOARD* and *EC_2* are the individuals of the *Transfer_Object* class. Both are inputs of the activity *FAT* that produces the individual *PRINTER* of the *Transfer_Object* class.

PH_DRIVE_BOARD: Transfer_Object
EC_2: Transfer_Object
PRINTER: Transfer_Object
(PH_DRIVE_BOARD, FAT): inputs
(EC_2, FAT): inputs
(FAT, PRINTER): outputs

Axiom 8. The instance *YSH_PRINTER_CO* of the *Corporation* class participates in the supply chain individual *PSC* as the individual *M1* of the *Manufacturer* class and performs the activity *YSH_MMP*. Moreover, it is a downstream entity of the individuals *TEEC* and *SHME* of the *Corporation* class and is an upstream entity of the individuals *AM_D*, *AS_D* and *EU_R* of the *Corporation* class.

YSH_PRINTER_CO: Corporation
M1: Manufacturer
(YSH_PRINTER_CO, M1): plays
(YSH_PRINTER_CO, PSC): participates-in
(YSH_PRINTER_CO, YSH_MMP): performs
TEEC: Corporation
SHME: Corporation
AM_D: Corporation
AS_D: Corporation
EU_R: Corporation
(YSH_PRINTER_CO, TEEC): downstream-of
(YSH_PRINTER_CO, SHME): downstream-of
(YSH_PRINTER_CO, AM_D): upstream-of
(YSH_PRINTER_CO, AS_D): upstream-of
(YSH_PRINTER_CO, EU_R): upstream-of

4.2. SCO-based integration methodology

Terminology mapping is a key step in the integration framework shown in figure 3 since it determines instance translations between concepts in applications and corresponding concepts in SCO. It normally specifies semantic equivalences of the entities within two interacting ontologies. A concept A1 is semantically equivalent with another concept B1 if any instance of A1 is a valid instance of B1 and any instance of B1 is also a valid instance of A1. Moreover, these concepts have the same class hierarchy and the same set of properties. Semantic equivalences between concepts in an ontology can be determined by using logic reasoning, such as structural subsumption reasoning in DL. However, two ontologies describing very similar domains are not exactly the same in supply chain systems, possibly owing to different purposes and description granularities. Semantically similar terms from the ontologies may agree on many, but not all, properties, subclasses and super-classes. Moreover, the number of properties, subclasses and super-classes of these terms may differ, as shown in UBL and OAGIS described below. In addition, this situation also appears between the SCO model and applications. The SCO model describes generic terminologies and vocabularies of the entire SCM domain and avoids providing very specific descriptions to facilitate semantic integration of supply chain systems. As a result, more high-level terms and relationships are contained in SCO, compared with UBL and OAGIS. Thus, the determination of semantically similar associations between SCO and the applications necessitates the addition of new specific terms to the SCO model and the provisional neglect of the scopes beyond the applications. To explain the proposed methodology, this subsection firstly describes the generation of OWL-based UBL and OAGIS from the XML Schema form and then presents an approach of similarly semantic mapping based on SWRL rules (Horrocks *et al.* 2004). In addition, the information integration process based on SCO is illustrated by a simple example.

4.2.1. Syntactic transformation of applications. In the scenario example described in figure 4, supply chain partners need to seamlessly exchange information for building good collaboration relationships. However, due to long-term business practices and cultures, different corporation individuals may use different information models or structures. For example, TEEC and YSH_PRINTER_CO use UBL and OAGIS, respectively, to describe their business information. They may fail to have a common understanding of the exchanged information owing to differences between these models. When they communicate by using SCO as an interlingua, syntactic transformation needs to be done, i.e. the OAGIS and UBL models are represented in OWL syntax.

OAGIS (OAGi 2006) builds a common Business Object Document (BOD) Message Architecture for communication between business applications, which provides a self-describing mechanism of BODs by defining XML schemas for the following four levels. The first level introduces metadata, represented as elements and attributes, to describe the BOD itself and the application creating the BOD, such as *ApplicationArea* and *revision*. The second level contains data types that are based on either predefined types or user-defined types, such as *AddressId*. The third level describes

extensible component types that are the large-grained building blocks of business documents, such as *PaymentTerms*. The top level defines various BODs, which contain business objects and actions that are to be applied to the objects, such as *AddPurchaseOrder*, *PurchaseOrder* and *Add*.

UBL (OASIS 2004) is intended to solve the problems caused by multiple industry-specific data formats, which accomplish the same purpose in different business domains, by defining a generic XML interchange format for business documents that can be extended to meet the requirements of particular industries. Specifically, UBL 1.0 provides a library of XML schemas for reusable data components and common business documents constructed from these components, such as *Address* and *PaymentMeans*. Currently, basic document types designed by UBL to support a generic order-to-invoice business process include *OrderType*, *OrderResponseSimpleType*, *OrderResponseType*, *OrderChangeType*, *OrderCancellationType*, *DespatchAdviceType*, *ReceiptAdviceType* and *InvoiceType*.

Both OAGIS and UBL adopt XML to represent business documents in supply chains and specify structure and content standards of the documents based on the XML Schema representation formalism. To represent OAGIS and UBL in OWL syntax is to translate information in their XML schema specifications into OWL classes, properties and axioms. This task uses the following syntactic translation rules.

- The *schema* of XML Schema is translated to *owl:Ontology*.
- The *targetNamespace* of XML Schema is translated to the OWL namespace with a suffix '#', which is enclosed in an opening *rdf:RDF* tag.
- The *simpleType* and *complexType* of XML Schema are translated to *owl:Class*.
- Global elements are translated to OWL classes. But a global element whose type has the same name is ignored.
- Local elements and attributes with a built-in simple type in XML Schema specification are translated to OWL datatype properties.
- Local elements and attributes with a user-defined type in the XML Schema files of OAGIS and UBL are translated to OWL object properties.
- The *extension* and *restriction* of data types are translated to *rdfs:subClassOf*.
- Cardinality constraints *minOccurs* and *maxOccurs* of XML Schema are translated to restrictions *owl:minCardinality* and *owl:maxCardinality*, respectively, on the corresponding OWL class.

Based on these rules, a syntactic translator is developed by using the Extensible Stylesheet Language Transformations (XSLT) in this research to automatically transform XML Schema to OWL for OAGIS and UBL, as shown in figure 5. For example, the *complexType Order* restricts the *Document* data type in the XML schemas of OAGIS. Through the use of a segment of XSLT stylesheets in figure 5, such restriction is translated to the *rdfs:subClassOf* construct in the corresponding OWL specification. Detailed XSLT stylesheets in the syntactic translator are reported in Appendix 2 and partial class hierarchies and object properties of OAGIS and UBL in OWL obtained by using the translator are shown in figure 6. In addition, opposite syntactic transformation can be realized by developing the translator executing the rules opposite to the above. Moreover, the XSLT-based translators need to be developed only once and then can be reused to perform the transformations of business data in different applications of different supply chain partners that are described in terms of different XML-based standards.

4.2.2. Semantic mapping and interoperability based on SWRL rules. As shown in figure 6, the UBL and OAGIS standards define different hierarchies of concepts for similar business document domains. Moreover, both the number and the names of properties are not exactly the same. By mapping very similar semantics to the SCO model, semantic integration between heterogeneous applications can be implemented. To this end, a methodology of building semantic mapping based on SWRL rules is presented. SWRL (Horrocks *et al.* 2004) combines the OWL DL and OWL Lite sublanguages with the Unary/Binary Datalog RuleML sublanguages. It mainly deals with Horn-like rules that are of the form of an implication between an antecedent (body) and consequent (head), meaning that the conditions specified in the consequent must hold whenever the conditions specified in the antecedent are satisfied. Both the antecedent and consequent consist of zero or more atoms. Atoms are the form $C(x)$, $P(x,y)$, $sameAs(x,y)$ or $differentFrom(x,y)$, where C is an OWL description, P is an OWL property, x and y are variables, OWL individuals or OWL data values. In addition to formal model–theoretic semantics, one of the major advantages of SWRL is the close association with OWL. Consequently, it is instinctive to adopt SWRL to describe semantic mapping rules in this study, considering that the SCO model is represented by OWL. Furthermore, as SWRL is a description language independent of specific rule languages within inference engines, the system implementation can be free to choose certain rule engine, such as Prolog (Endriss 2006) and Jess (Jess 2006), to implement rule-based semantic integration without modifying SWRL rules. In other words, SWRL rules can remain unchanged for the same semantic transformation results even though the engine has been changed from one (e.g. the Prolog rule engine) to the other (e.g. the Jess rule engine). In the implementation architecture of the supply

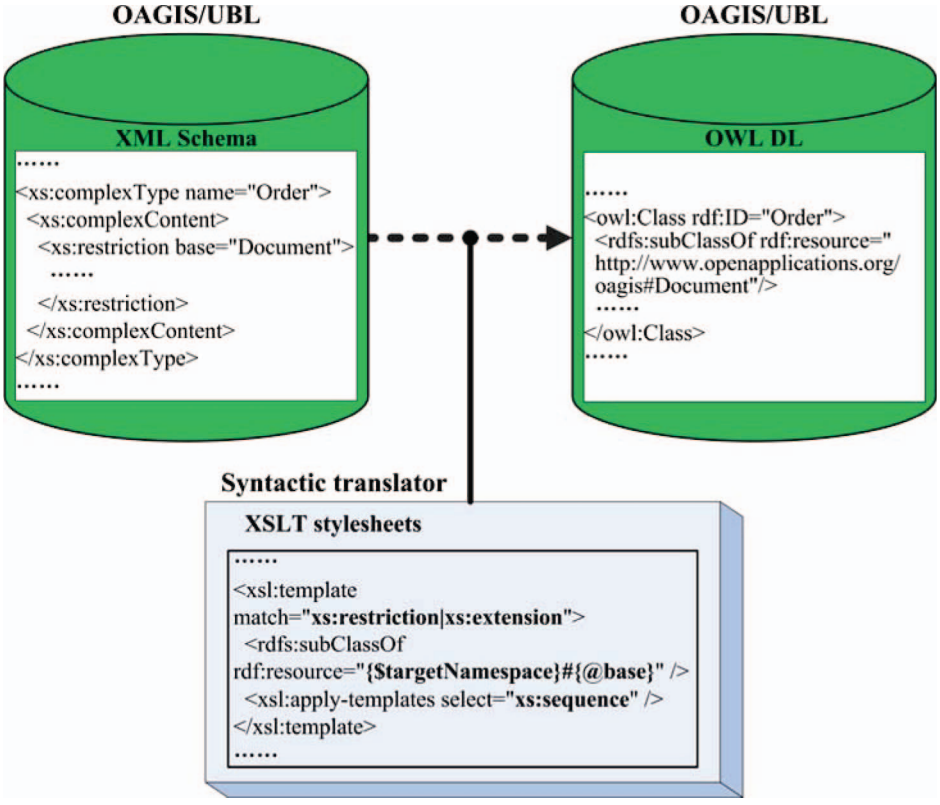

Figure 5. Syntactic transformation through a XSLT-based translator.

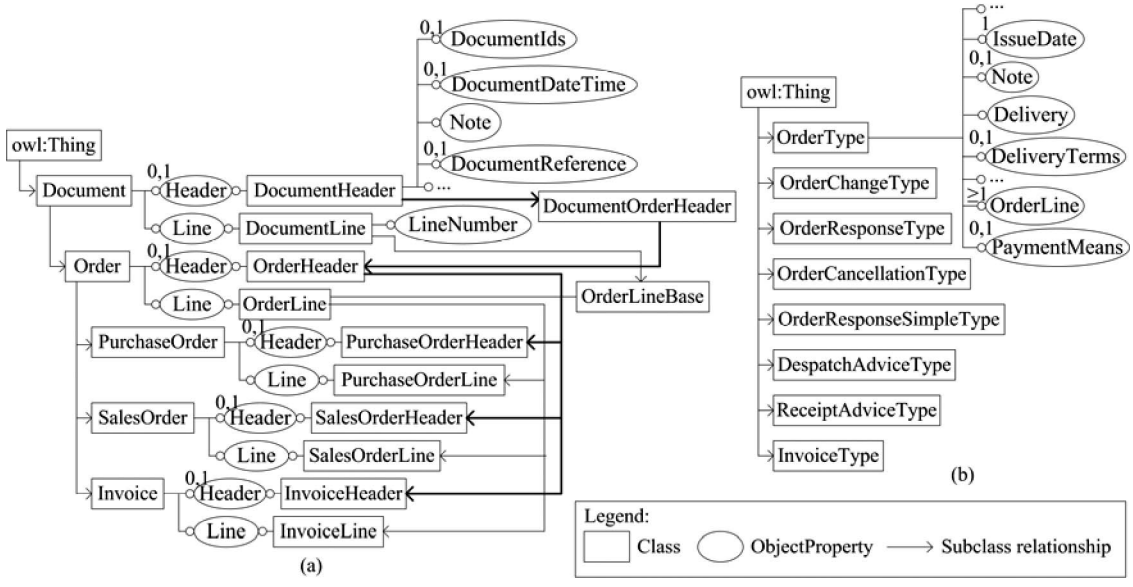

Figure 6. (a) Section of classification hierarchy and properties of OAGIS. (b) Section of classification hierarchy and properties of UBL.

chain integration approach, the Jess rule engine is used to enable semantic transformation by executing mapping rules represented by SWRL.

Semantic mapping in the integration architecture falls into two major categories: one-to-one mapping and complex mapping. The former shows matches between a pair

of terms belonging to two ontologies, which are represented by SWRL rules, as exemplified in the following two rules.

Rule 1. *oagis:PurchaseOrder(?x) ⇒ sco:Purchase_Order(?x)*
Rule 2. *sco:document-date-time(?x,?y) ⇒ ubl:IssueDate(?x,?y)*

Prefixes *oagis:*, *ubl:* and *sco:* in these rules denote namespaces associated with OAGIS, UBL and SCO, respectively, to refer to terms in three models in an unambiguous manner and without causing name clashes. Rule 1 represents the *PurchaseOrder* class in OAGIS is mapped to the class *Purchase_Order* in SCO. Rule 2 expresses the property *document-date-time* in SCO is translated to the *IssueDate* property in UBL. In addition, inverse translations are held for the two pairs of terms and are represented by additional rules. However, there may be only one-way semantic mappings between other terms. For example, a semantic mapping between the class *Document* in OAGIS and the class *Business_Document* in SCO is defined as follows.

Rule 3. *oagis:Document(?x) ⇒ sco:Business_Document(?x)*

Though there are little differences in the sets of properties of two classes in Rule 3, the scope of the hierarchy of the *Document* class in OAGIS is less than that of the *Business_Document* class in SCO. Hence, the class *Document* can be mapped to the class *Business_Document* and an inverse translation is not held.

A complex mapping specifies that a combination of terminologies in one ontology corresponds to a combination in the other. Such a combination can be created in many ways, such as the disjunction or conjunction of terms and additional constraints. The following rules build complex semantic mappings among terms in OAGIS, UBL and SCO.

Rule 4. *sco:Purchase_Order(?x) ⇒ (ubl:OrderType ∨ ubl:OrderChange)(?x)*
Rule 5. *oagis:SalesOrder(?x) ⇒ (sco:Sales_Order ∨ sco:Customer_Order)(?x)*
Rule 6. *oagis:Header(?x,?y) ∧ oagis:DocumentDateTime(?y,?z) ⇒ sco: document-date-time(?x,?z)*
Rule 7. *ubl:OrderType(?x) ∧ xsd:String(?y) ⇒ sco:Purchase_Order(?x) ∧ sco:has_status(?x,?y) ∧ swrlb:stringEqualIgnoreCase(?y,issued)*

The first two rules describe that a class in an ontology is translated to the disjunction of two classes in the other. Rule 4 shows the class *Purchase_Order* in SCO is transformed to the class *OrderType* or *OrderChange* in UBL. Rule 5 indicates the class *SalesOrder* in OAGIS is mapped to the class *Sales_Order* or *Customer_Order* in SCO. The third rule builds a complex semantic mapping among properties. The fourth rule represents the class *OrderType* in UBL is translated to the class *Purchase_Order* in SCO, of which the *has-status* property has *issued* as its value.

After building similarly semantic mappings among ontologies, SWRL rules can be applied to semantic transformation among applications. This can be easily understood by semantically translating instances in OAGIS into those in UBL. Initially, an application using OAGIS contains an instance *ORDER1* of the class *PurchaseOrder* that has *PO_HEADER1* and *Apr_05_2006* as its header and creation date, respectively. Based on the syntactic translations in section 4.2.1, the information can be represented by the DL assertions as follows:

ORDER1: PurchaseOrder
(ORDER1, PO_HEADER1): Header
(PO_HEADER1, Apr_05_2006): DocumentDateTime

The use of Rules 1 and 6 can realize semantic translation from the information in the application into the following assertions represented by terms, semantics and syntax of SCO.

ORDER1: Purchase_Order
(ORDER1, Apr_05_2006): document-date-time

Further, by applying Rules 2 and 4, these assertions are translated to the following information with terms and semantics of UBL.

ORDER1: (OrderType ∨ OrderChange)
(ORDER1, Apr_05_2006): IssueDate

These mean that the instance *ORDER1* in OAGIS can be mapped to the instance of the class *OrderType* or the class *OrderChange* in UBL. It should be noted that partial semantic mapping rules and instance information are described here. To definitely translate *ORDER1* to the instance of one of two classes requires additional rules and information.

5. Implementation architecture

A prototype system is developed using the Java platform to implement the proposed approach to syntactic and semantic transformations for supply chain integration. The architecture of the system is depicted in figure 7 and consists of three main parts, namely syntactic translator, mapping rule editor and semantic translator.

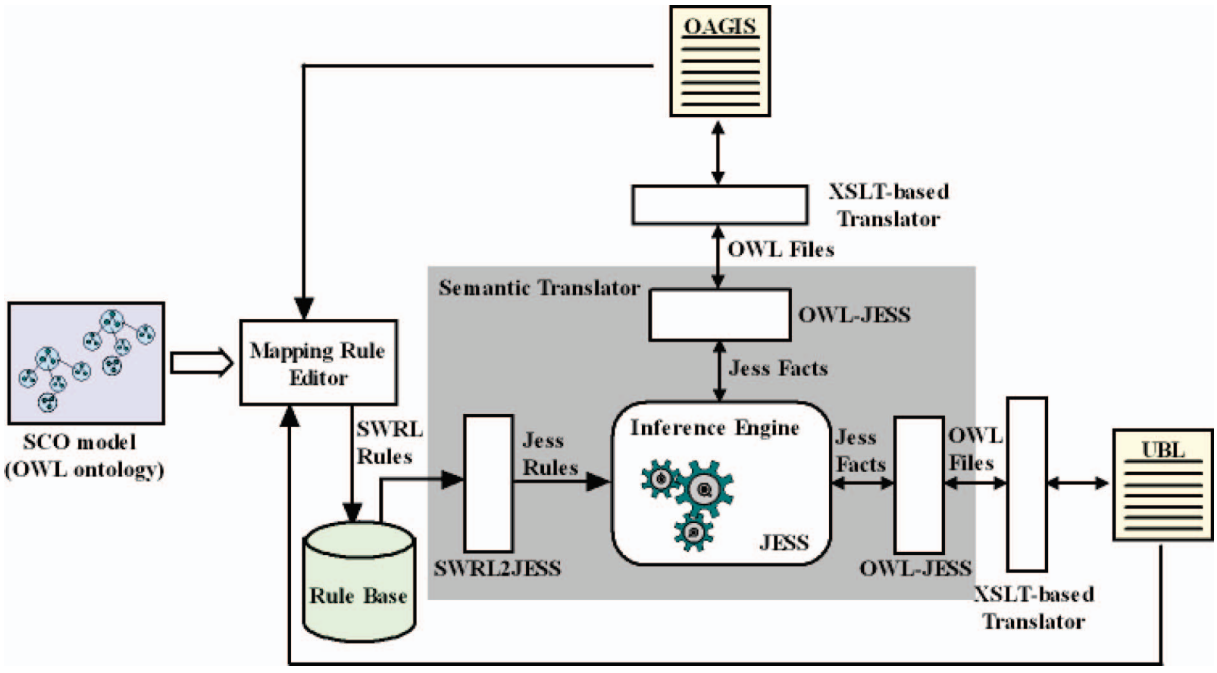

Figure 7. Implementation architecture of the prototype system.

- *Syntactic translator*. Syntactic translators are developed by using XSLT and then executed by the Xerces Java Parser (Xerces 2001), a Java XML parser for manipulating XML documents according to the DOM and SAX specifications. They enable the syntactic equivalence transformations between OAGIS-based or UBL-based documents and the corresponding OWL files, as shown in figure 7.
- *Mapping rule editor*. The editor offers a graphical interface for users to map the terms between application ontologies and the SCO model in a drag-and-drop manner. The mapping results are outputted by this editor in the form of SWRL rules that are saved in the rule base.
- *Semantic translator*. The semantic translator comprises the OWL-JESS, SWRL2JESS and inference engine, as shown in figure 7. Both the OWL-JESS and SWRL2JESS components are developed by using Jena (Jena 2005). The OWL-JESS enables the transformations between classes and instances in the OWL documents and Jess facts. The SWRL2JESS tool transforms SWRL rules into Jess rules that can be processed by the Jess rule engine (Jess 2006). For example, Rule 6 in section 4.2.2 can be translated into the corresponding Jess rule as follows:

Defrule Rule6 (oagis_Header(?x,?y) oagis_DocumentDateTime(?y,?z))
⇒ (assert (sco_document-date-time(?x,?z))

Based on the translated Jess facts and Jess rules, the inference engine can perform reasoning to realize semantic transformation between applications via the SCO model. The engine is implemented through the use of Jess (Jess 2006) that is a rule engine for the Java platform. Jess uses the Rete algorithm—a very efficient mechanism for solving difficult many-to-many matching problems—to process rules. The Jess system contains a rule base, a fact base and an execution engine. The execution engine can automatically perform reasoning by matching the facts in the fact base with the rules in the rule base.

6. Conclusions

Effective and efficient SCM needs to integrate heterogeneous applications in supply chains based on common representation and exchange of information semantics. Current related researches are not adequate to address the problem because they do not deal with knowledge interoperability in the web environment. This paper proposes a SCO-based architecture to implement web-based supply chain integration. The SCO model identifies general terms representing concepts and relationships common to SCM based on the skeletal methodology. Moreover, OWL DL is used to formally define the model to provide semantic foundation for information integration in the semantic supply chain networks. Formal logics and reasoning capabilities of DL provide enough expressiveness for extending SCO to describe information semantics of specific

application domains and assure the consistency of such extension. In addition, the SCO model can act as an interlingua for a supply chain application integration architecture, which includes an integration process composed of terminology mapping, syntactic transformation and semantic transformation. XSLT-based translators are developed to realize syntactic translation rules between XML Schema and OWL, which facilitates syntactic transformation among many XML-based data interchange standards in current SCM systems. In addition, a method of building similarly semantic mapping based on SWRL rules is presented to specify one-to-one mapping rules between application ontologies and SCO and to determine complex mapping rules among various combinations of terminologies. The implementation of these rules can enable semantic interoperability among heterogeneous applications in supply chains. Finally, information integration between different applications in an actual supply chain scenario through SCO illustrates the effectiveness of the approach and the architecture of the developed prototype system is described to explain the implementation of the methodology. More application domains will help to further validate and perfect the proposed model and methodology as well as the system implementation.

Acknowledgements

The work reported in this paper was supported by a Research Grant from National Natural Science Foundation of China (Project No. 70471023).

Appendix 1: The SCO model in OWL DL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/sco.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
     syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-
    schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-ontologies.com/sco.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Activity">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/
    2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="serves-for"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Purpose"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="requires"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Resource"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="performed-by"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="Party"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="outputs"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Transfer_Object"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Role"/>
  <owl:Class rdf:ID="Performance">
    <rdfs:subClassOf rdf:resource="http://www.w3.org/
    2002/07/owl#Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="Performance_Metric"/>
        </owl:allValuesFrom>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="measured-by"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Business_Order">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Business_Document"/>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:ID="Purchase_Order">
  <rdfs:subClassOf rdf:resource ="#Business_Order"/>
</owl:Class>
<owl:Class rdf:about="#Transfer_Object">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Activity"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="inputs"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="SC_Structure">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="made-up-of"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Party"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="#Purpose">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="controls"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Activity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="affects-use"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Resource"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:about="#Supply_Chain">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has-activity"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Activity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#SC_
      Structure"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has-structure"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#
       Performance"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="realises"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#Purpose"/>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="has-purpose"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="composed-of"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Party"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Resource">
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
  2002/07/owl#Thing"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="required-by"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#Activity"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
.....
<owl:ObjectProperty rdf:about="#upstream-of">
  <owl:inverseOf rdf:resource="#downstream-of"/>
```

```
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#controls">
    <owl:inverseOf rdf:resource="#serves-for"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#performs">
    <owl:inverseOf rdf:resource="#performed-by"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#participates-in">
    <owl:inverseOf rdf:resource="#composed-of"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#inputs">
    <owl:inverseOf rdf:resource="#outputs"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#requires">
    <owl:inverseOf rdf:resource="#required-by"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#has-structure">
    <rdfs:domain rdf:resource="#Supply_Chain"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/
    owl#FunctionalProperty"/>
  </owl:ObjectProperty>
  ....
</rdf:RDF>
```

Appendix 2: XSLT stylesheets for transforming XML Schema to OWL

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="2.0" ......>
  ......
  <xsl:template match="xs:complexType">
    <owl:Class rdf:ID="{@name}">
      <xsl:apply-templates select="xs:complexContent/
      xs:restriction|xs:complexContent/xs:extension"/>
    </owl:Class>
  </xsl:template>
  <xsl:template match="xs:restriction|xs:extension">
    <rdfs:subClassOf rdf:resource="{$targetNamespace}
    #{@base}"/>
    <xsl:apply-templates select="xs:sequence"/>
  </xsl:template>
  <xsl:template match="xs:sequence">
    <xsl:for-each select="xs:element">
      ......
      <xsl:if test="@ref">
        <rdfs:subClassOf>
          <owl:Restriction>
            <owl:onProperty rdf:resource="{$target
            Namespace}#{@ref}"/>
            <xsl:if test="@minOccurs">
              <owl:minCardinality rdf:datatype=
              "&xsd;nonNegativeInteger">
                <xsl:value-of select="@minOccurs"/>
              </owl:minCardinality>
            </xsl:if>
            <xsl:if test="@maxOccurs">
              <owl:maxCardinality rdf:datatype=
              "&xsd;nonNegativeInteger">
                <xsl:value-of select="@maxOccurs"/>
              </owl:maxCardinality>
            </xsl:if>
          </owl:Restriction>
        </rdfs:subClassOf>
      </xsl:if>
      <xsl:if test="@name!=''and @type!=''">
        <rdfs:subClassOf>
          <owl:Restriction>
            <owl:onProperty rdf:resource="{$target
            Namespace}#{@name}"/>
            <owl:allValuesFrom rdf:resource="{$target
            Namespace}#{@type}"/>
          </owl:Restriction>
        </rdfs:subClassOf>
      </xsl:if>
      ......
    </xsl:for-each>
  </xsl:template>
  ......
</xsl:stylesheet>
```

References

Antoniou, G. and van Harmelen, F., *A Semantic Web Primer*, 2004 (MIT Press: Cambridge, MA).

Baader, F., Calvanese, D., McGuineness, D.L., Nardi, D. and Patel-Schneider, P.F., *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003 (Cambridge University Press: Cambridge).

de Bruijn, J., *et al*., D2v1.2 Web Service Modeling Ontology (WSMO), 2005a. Available online at: http://www.wsmo.org/TR/d2/v1.2/D2v1-2_20050414.pdf (accessed 17 August 2006).

de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M. and Fensel, D., D16.1v0.3 The Web Service Modeling Language WSML, 2005b. Available online at: http://www.wsmo.org/TR/d16/d16.1/v0.3/d16.1v0.3_20051005.pdf (accessed 20 August 2006).

Bussler, C., *B2B Integration: Concepts and Architecture*, 2003 (Springer: Berlin).

Ciocoiu, M., Nau, D.S. and Gruninger, M., Ontologies for integrating engineering applications. *ASME J. Comput. Inform. Sci. Eng*., 2001, **1**, 12–22.

Corcho, O., Fernández-lópez, M. and Gómez-pérez, A., Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data Knowl. Eng*., 2003, **46**, 41–64.

Endriss, U., An introduction to Prolog programming, 2006. Available online at: http://staff.science.uva.nl/~ulle/teaching/prolog/prolog.pdf (accessed 10 August 2006).

Feier, C. and Domingue, J., D3.1v0.2 WSMO Primer, 2005. Available online at: http://www.wsmo.org/TR/d3/d3.1/v0.2/wsmo-d3.1-v0.2.pdf (accessed 17 August 2006).

Fox, M.S. and Gruninger, M., Enterprise modeling. *AI Mag*., 1998, **19**, 109–121.

Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F. and Tu, S.W., The evolution of Protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.*, 2003, **58**, 89–123.

Grüninger, M., Ontology of the Process Specification Language. In *Handbook on Ontologies*, edited by S. Staab and R. Studer, pp. 575–592, 2004 (Springer: Berlin).

Haarslev, V. and Möller, R., Racer: a core inference engine for the semantic web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON-2003)*, 2003, pp. 27–36.

Haller, A., Cimpian, E., Mocan, A., Oren, E. and Bussler, C., WSMX—a semantic service-oriented architecture. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2005)*, 2005, pp. 321–328.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M., SWRL: a semantic web rule language combining OWL and RuleML. World Wide Web Consortium Member Submission, 2004. Available online at: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/ (accessed 1 June 2005).

Horrocks, I., Patel-Schneider, P.F. and van Harmelen, F., From SHIQ and RDF to OWL: the making of a web ontology language. *Web Semantics: Sci., Serv. Agents World Wide Web*, 2003, **1**, 7–26.

Huang, G.Q., Lau, J.S.K. and Mak, K.L., The impacts of sharing production information on supply chain dynamics: a review of the literature. *Int. J. Prod. Res.*, 2003, **41**, 1483–1517.

Jena, Jena—a semantic web framework for java. Available online at: http://jena.sourceforge.net/ (accessed 8 December 2005).

Jess, Jess—the rule engine for the java platform. Available online at: http://www.jessrules.com/jess/index.shtml (accessed 15 December 2006).

Knublauch, H., Fergerson, R.W., Noy, N.F. and Musen, M.A., The Protégé OWL plugin: an open development environment for semantic web applications. *Lecture Notes Comput. Sci.*, 2004, **3298**, 229–243.

Lambert, D.M. and Cooper, M.C., Issues in supply chain management. *Ind. Mark. Manage.*, 2000, **29**, 65–83.

Mocan, A. and Cimpian, E., D13.3v0.2 WSMX data mediation, 2005. Available online at: http://www.wsmo.org/TR/d13/d13.3/v0.2/20051011/d13.3v0.2_20051011.pdf (accessed 25 August 2006).

OAGi, OAGIS 9.0. Open Applications Group, Inc. (OAGi). Available online at: http://www.openapplications.org/downloads/oagidownloads.htm (accessed 21 May 2006).

OASIS, Universal Business Language (UBL) 1.0. OASIS UBL Technical Committee, 2004. Available online at: http://docs.oasis-open.org/ubl/cd-UBL-1.0/ (accessed 21 May 2006).

Owen, J., *STEP: An Introduction*, 1993 (Information Geometers: Winchester).

Pinto, H.S. and Martins, J.P., Ontologies: how can they be built? *Knowl. Inform. Syst.*, 2004, **6**, 441–464.

Ray, S.R. and Jones, A.T., Manufacturing interoperability. In *Proceedings of the 10th ISPE International Conference on Concurrent Engineering: Research and Application*, 2003, pp. 535–540.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D., Web service modeling ontology. *Appl. Ontol.*, 2005, **1**, 77–106.

Shen, H., Wall, B., Zaremba, M., Chen, Y. and Browne, J., Integration of business modelling methods for enterprise information system analysis and user requirements gathering. *Comput. Ind.*, 2004, **54**, 307–323.

Sintek, M., OntoViz. Available online at: http://protege.cim3.net/cgi-bin/wiki.pl?OntoViz (accessed 28 November 2006).

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

Supply Chain Council, Supply-chain operations reference-model (SCOR) version 7.0 overview, 2005. Available online at: http://www.supply-chain.org/galleries/default-file/SCOR%20Overview%207.0%201_06.pdf (accessed 9 November 2005).

Ulieru, M. and Cobazru, M., Building holonic supply chain management systems: an e-logistics application for the telephone manufacturing industry. *IEEE Trans. Ind. Informatics*, 2005, **1**, 18–30.

Uschold, M. and King, M., Towards a methodology for building ontologies. In *Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995. Available online at: http://citeseer.ist.psu.edu/uschold95toward.html (accessed 2 November 2004).

Uschold, M., King, M., Moralee, S. and Zorgios, Y., The enterprise ontology. *Knowl. Eng. Rev.*, 1998, **13**, 31–89.

Vasiliu, L., Zaremba, M., Moran, M., Bussler, C. and Browne, J., Web-service semantic enabled implementation of machine vs. machine business negotiation. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2004)*, 2004, pp. 106–115.

Xerces, Xerces java parser readme, 2001. Available online at: http://xerces.apache.org/xerces-j/ (accessed 29 December 2005).

Zaremba, M., Moran, M., Haselwanter, T., Lee, H.K. and Han, S.K., D13.4 v0.3 WSMX Architecture, 2005 Available online at: http://www.wsmo.org/TR/d13/d13.4/v0.3/20051012 (accessed 25 August 2006).