Int J Adv Manuf Technol (2008) 39:199–210
DOI 10.1007/s00170-007-1198-2

ORIGINAL ARTICLE

Manufacturing interoperability using a semantic mediation

Seog-Chan Oh · Shang-Tae Yee

Received: 24 April 2007 / Accepted: 6 August 2007 / Published online: 9 September 2007

Abstract Globalization has become a major trend in today's business environment, which has led to the explosion in the amount of information shared amongst business partners. It is critical to share information correctly, inexpensively, and effectively for accomplishing business goals using global information sharing architectures under increasing pressure of competitive markets. One of the major barriers to build such a global information sharing architecture is caused by the absence of universally accepted ontology needed to facilitate business partners to be seamlessly interoperable with one another within a global manufacturing world. In reality, due to heterogeneous enterprise environments in which business partners find themselves, multiple ontologies are defined and used. In this paper, we present a method for semantically mapping different business documents to a conforming document format, given inevitable existence of multiple ontolgoies. Forward reasoning rules are used to express the mapping, and they are applied to exchange messages between heterogeneous business partners. We demonstrate our semantic mapping approach by using a demonstrator that uses Web services for service components and the Jena package for a reasoning engine. A business scenario of the demonstrator consists of a part ordering transaction taking place between an original equipment manufacturer and its supplier. This approach can also be applied to other manufacturing operations such as parts scheduling, order sequencing, and inventory.

S.-C. Oh · S.-T. Yee (✉)
Manufacturing Systems Research Laboratory,
General Motors R&D Center,
30500 Mound Road,
Warren, MI 48090, USA
e-mail: shang-tae.yee@gm.com

S.-C. Oh
e-mail: seogchano@gmail.com

Keywords Semantic mediation · Manufacturing interoperability · Ontology · Logic programming · Web service · Service oriented architecture

1 Introduction

It is evident that today's manufacturing world has experienced fast growth on both volume and scope in information sharing amongst business partners. Because companies have been using heterogeneous information systems, they primarily have used standard-based approaches (e.g, RosettaNet, ebXML) for large scale information sharing.

However, these standard-based approaches have raised several issues and problems. First of all, the standard-based approach forces whole trading partners to follow a single unified standard, ignoring the heterogeneous nature inherent in business partners' environments. An automotive company has vehicle programs and one vehicle production program usually needs 20,000 parts from 8,000 different tier-1 suppliers. Therefore, it does not make sense to use a single unified standard, ignoring the heterogeneous nature of suppliers. Second, it is significantly inefficient and difficult to version, customize, and integrate complex industrial standards. For example, the purchase order schema of AIDIMA[1] and the sale order schema of UBL[2] are not interoperable because of many schema mismatches, such as terminology, structure, data organization, and data

[1] A Spanish research and development association for the wood and furniture industries consisting of more than 650 manufacturers.
[2] OASIS Universal Business Language (UBL) with intension to become an international standard for electronic commerce freely available to everyone without licensing or other fees.

granularity, even though two schemas share the same semantics at higher abstract level. Third, because these standards allow flexibility in terms of message contents and their processes composed, a significant effort is required to implement precisely business transactions [12], even though the partners agreed to use them. Fourth, an excessive lead-time is required to accept new partners and connect them to existing partners. Lastly, the traditional standardization process cannot manage semantics of messages effectively.

To address the issues and problems of the standard-based approaches, semantic mediation-based technologies have been introduced. If business partners can describe their semantics on how to interact with themselves, it is possible to use the semantic mediation technologies to accomplish interoperability and, thereby, to build business transaction processes and share information accordingly. However, there is a major barrier to build a global semantic interoperable architecture because no universally accepted ontology exists. In reality, companies are exposed to multiple ontologies because they are in heterogeneous enterprise information system's environments. In this paper, we present a semantic mediation method that maps different ontologies, translates them into one business document format, and exchanges the translated documents between the partners.

Automated and seamless business transactions between the partners would result in improvement of data accuracy, elimination of repetitive manual operations, and concentration on more important and meaningful tasks.

To motivate our work, we use a typical part ordering scenario as an example. Let's consider a manufacturing company named OEM, which needs to outsource its part to a supplier named T1. When OEM consumes parts, it sends a message to T1 to place an order for the part. T1 receives and processes the part ordering (PO) message based on its inventory availability, and delivers ordered parts to OEM. In this scenario, both OEM and T1 may use different XML document formats for the same purpose. In other words, business partners OEM and T1 may use heterogeneous data formats, implying that they have a significant problem in exchanging information in an automated manner.

As shown in Fig. 1, OEM and T1 have different XML instance documents even for the same purpose, and this issue should be addressed in order to enable automatic communication. In detail, there are two distinct mismatches between their XML documents, which should be translated as follows:

1. 1:1 translation between requestTime of OEM and creationDataTime of T1.
2. 2:1 translation by joining both sender and part/name of OEM to part/name of T1.

In the first case, requestTime of OEM has the direct counterpart entry, that is, creationDataTime of T1. We refer to this case as *sameAs 1:1 translation*, and it can be resolved using the one-to-one pattern matching forward rule that will be introduced in the following sections. In the second case, both sender and part/name of OEM have no direct counterpart entries in T1. Instead, both can be joined together to build a new entry, namely, part/name of T1. Precisely, both sender and part/name of OEM need to be concatenated with a hyphen (-) to build part/name of T1. We denote this case as *joinTo 2:1 translation*. In the following sections, we will specify ways to resolve two translation cases by using the logic programming approach and the bi-translation method between XML and RDF.

When OEM deals with only T1, it would be good enough to unify the document formats between them. However, when the number of suppliers increases, it would not be feasible to unify the partners' document formats, because for even a single business process, numerous XML instance documents are used.

The rest of this paper is organized as follows. In Sect. 2, we survey the methodological background of the work with

```
<OEMPartOrder xmlns="http://www.oem.com/ ">
  <sender> OEM </sender>
  <requestTime>
      Wed Mar 14 21:51:18 PDT 2007
  </requestTime>
  <documentID >100</documentID>
  <part >
      <name>Gear ASM</name>
      <id>25737126</id>
  </part>
</ OEMPartOrder >
```

a

```
<T1PartOrder xmlns="http://www.t1.com/">
  <documentID>100</documentID>
  <creationDataTime>
    Wed Mar 14 21:51:18 PDT 2007
  </creationDataTime>
  <part >
    <id>25737126</id>
    <name>OEM-Gear ASM</name>
  </part>
</ T1PartOrder >
```

b

Fig. 1 Different XML documents of OEM and T1 for the same purpose. **a** Instance of OEM's XML document. **b** Instance of T1's XML document

the focus on semantic mediation technologies. In Sect. 3, we present the RDF transformation concept following a logic programming approach with formal definitions about mapping rules. In Sect. 4, we propose how semantic mediation technologies could accomplish interoperability for heterogeneous enterprise networks, especially by resolving the motivating scenario. In Sect. 5, we describe our demonstrator that implements the semantic interoperability between business partners of the motivating scenario, and suggest a scalable framework for implementing large-scale interoperability. In Sect. 6, we position our work with related works on enterprise interoperability. In Sect. 7, we present conclusions and further research work.

2 Survey of techniques for interoperability

In this section, we review various available techniques used to achieve interoperability. These methods can be employed for the semantic mediation approach.

2.1 Web services and Service Oriented Architecture (SOA)

A Web service is a set of related functionalities that can be loosely coupled with other services programmatically through the Web. Web services have taken the concept of services that are designed to fit an environment that needs dynamic discovery, binding, and execution of services. Web services are published and managed using technologies such as XML, WSDL (Web Service Description Language) [4], SOAP (Simple Object Access Protocol) [7], and UDDI (Universal Description Discovery and Integration) [10]. SOA is an architectural style for building software applications that use services distributed in a network such as the Web. The SOA concept is to design individual functions that can be blended to provide hundreds of different services, instead of designing individual programs that perform hundreds of functions. It promotes loose coupling between software components in such a way that they can be reused. In essence, SOA and Web services are two different things, but Web services are the preferred standard for providing services to realize SOA. In theory, SOA should be a perfect architectural model to bridge organizations, services, platforms, and networks in order to achieve interoperability between them.

2.2 Ontology reconciliation

Ontologies are constructed to specify the conceptual model of an information and knowledge domain explicitly. While ontology describes the domain, a knowledge base contains a particular state of affairs based on an ontology. Ontologies can be used in supporting information and knowledge exchange between different organizations. By 2010, ontologies using strong knowledge representations will be the basis for 80% of application integration projects [6]. Since information and knowledge domains are diverse and even evolve, different people and organizations tend to adopt different ontologies. As shown by Madnick [16], we cannot hope that one universally accepted unchanging ontology, even for a small domain, would ever be created. Therefore, in order to achieve interoperability of information and knowledge among heterogeneous organizations, different ontologies must be reconciled.

There exists a variety of alternative architectures to reconcile multiple ontologies for interoperability of heterogeneous organization networks. Hameed et al. [8] suggests three architectures as shown in Fig. 2.

Figure 2a shows the pairwise mapping which is used in the case when there is no need to reconcile all ontologies, but rather just interrelate individual ontologies as needed. While this approach gives great flexibility and simplicity, in the worst case, there will be as many sets of mappings as O (n^2), if n individual ontologies are required to be mapped to the rest of others in a bidirectional way. Figure 2(b) depicts the mapping based on a single common ontology, when

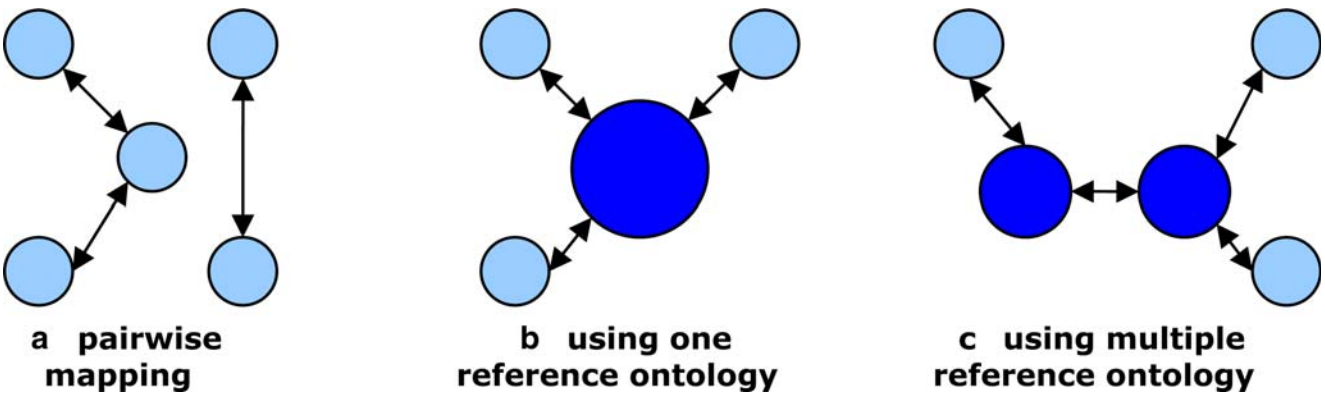

Fig. 2 Various ontology reconciliation architectures where the *nodes* represent ontologies and bidirectional arcs depict mapping; the *large dark-shaded node* is a common ontology; *small dark-shaded nodes* are reference ontologies; *light-shaded nodes* are individual ontologies

there is an attempt to reconcile individual ontologies in a principled, top-down fashion. This approach supposes that a common and standard conceptualization is identified and developed, whatever the cost of the development might be. It also loses some flexibility in the local management level, because all individual ontologies must follow a centralized standard. Figure 2c illustrates the case that uses multiple reference ontologies, forming clusters of interrelated ontologies. Each individual ontologies are mapped to the reference ontology for its cluster, and the reference ontologies are mapped to each other. This hybrid approach is to combine the advantages of Fig. 2a and b—a reduced number of mapping using principled conceptualization, and yet also there is flexibility to extend interoperability through adding different clusters. In this paper, we will simply use the pairwise mapping concept to highlight our approach concretely, where two business partners should translate their heterogeneous message documents for their seamless transaction. And yet, in Sect. 5.2, we will further discuss a scalable interoperability framework using the ontology reconciliation technique in order to accommodate multiple business partners.

2.3 RDF transformation

The Resource Description Framework (RDF) is used for describing resources that are identifiable by Uniform Resource Identifiers (URIs). RDF is based on the triplet model—subject–property–object—where these triplets together form a graph that serves as the representation of data, information, and knowledge in a machine-understandable way. It is evident that there is a need to transform RDF documents of different domains in order to exchange knowledge. However, at present, there is no standard way to transform RDF documents. Peer [18] surveyed several approaches to transform RDF documents or queries, as shown in Table 1.

While each approach in Table 1 has its different characteristics, in this paper, we will choose the logic programming approach suggested by Bowers and Delcambr [2], because of its high expressiveness as well as simplicity. We will discuss this logic programming approach in Sect. 3 in detail.

2.4 Transformation between XML and RDF

At present, many legacy systems already have developed their XML interfaces in support of business to business (B2B) integration scenarios, but very few of them have implemented RDF(S) interfaces. To address this XML to RDF and RDF to XML transformation, there are a number of approaches available today [12]. In this paper, we choose to use the transformation mechanism suggested in [15], especially because of its simplicity, that is, the approach does not necessarily require an XML schema to be present when transforming an XML instance into an RDF instance and vice versa.

When observing the XML Schema and XML instance documents, we can notice that only XML Schema constructs, such as xsd:element and xsd:attribute, occur in the XML instance documents. In other words, the XML instance document is composed only of elements and attributes.

Based on this observation, the chosen transformation mechanism builds an internal representation of the XML Schema consisting of only necessary elements and attributes from the XML Schema, and forms the internal representation in terms of a tree. The tree has nodes which encapsulate information such as name, type, extension, restriction, and namespace. Eventually, the internal tree is transformed into RDF through predefined transformation rules and a naming convention. For example, the XML instance document in Fig. 3a can be transformed into the RDF instance document in Fig. 3b using the transformation mechanism, guaranteeing that the transformed document is

Table 1 Survey of existing RDF transformation approaches

Approach	Characteristics
DAML+OIL [9]	DAML+OIL sameAs properties (sameClassAs, samePropertyAs and sameIndividualAs) are used to provide the equivalence between classes with classes, properties with properties, and instances with instances. However, it is limited when major structural differences exist between RDF graphs.
XSLT [5]	XSLT can be used to perform transformation on RDF documents, because RDF transformation can be seen as a styling transformation (e.g., styling XML documents in HTML). However, it must show weakness when the transformation requires expression in logic-based rules.
KR-transformation	OntoMorph [3] represents this approach. OntoMorph achieves the high-expressiveness transformation using so-called rewrite rules. One open issue associated with this approach is attributed to its heavy weight expressive features which may give away many opportunities for simplifications.
Logic programming approach [2]	RDF can be interpreted as an application of First Order Predicate Calculus. Therefore, a transformation between two RDF documents can be defined by a set of mapping rules that consist of predicates to deal with RDF triplets.

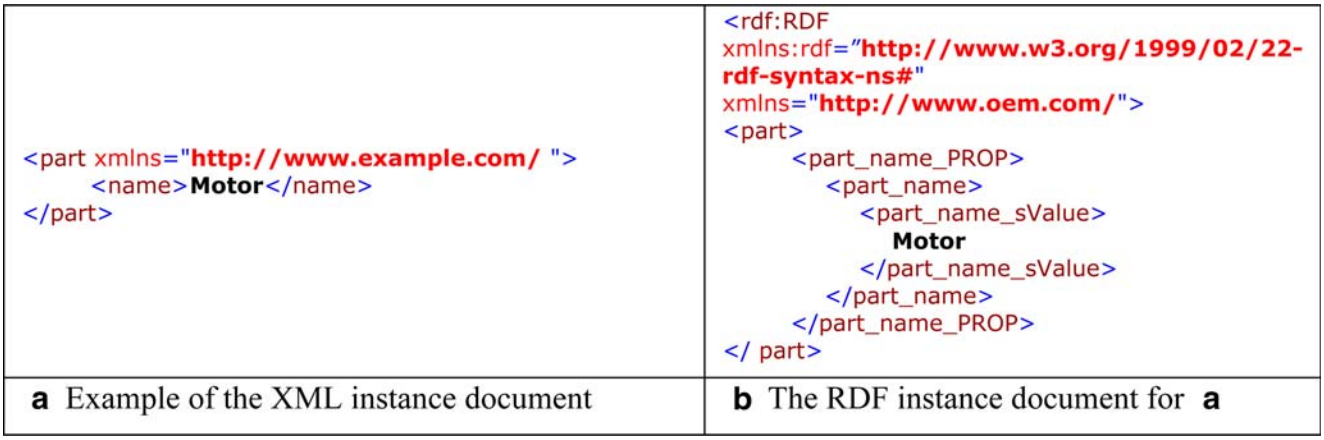

Fig. 3 Example of the XML to RDF transformation

valid according to the corresponding RDF schema. Of course, we can transform an RDF instance into an XML instance in a reverse way, because this transformation mechanism is a bi-translation method. Usually, we will use the term 'lift' to refer to the semantic transformation from XML to RDF. Conversely, we will use the term 'lower' to refer to the reverse transformation from RDF to XML.

3 Logic programming based RDF transformation

In this section, we present the RDF transformation concept following the logic programming approach, providing the high expressiveness required to build generic services for management of RDF documents [13]. Table 2 shows the basic definition used to formalize the RDF transformation. A transformation between two RDF documents is defined by a mapping scheme M that consists of a set of pattern-matching forward rules, where the rules are defined over triplets of the RDF representation.

RDF triplets are represented in τ. For example, "The sky has the color blue" in RDF can be formatted to be τ('sky' 'hasColorof' 'blue'), where (1) a subject, 'sky' denotes "the sky"; (2) a property, 'hasColorof' denotes "has the color"; and (3) an object, 'blue' denotes "blue".

For L_{old}={ τ('sky' 'hasColorof' 'blue')}, we can define a predicate $S(L_{\text{old}}, \tau$('sky' 'hasColorof' 'blue')) . The predicate S is evidently true because the RDF triplet, ('sky' 'hasColorof' 'blue') exist in L_{old}. We often want to reflect the weather changing situation, because the color of sky must be changed due to the flexible weather condition. When the current sky does not have its color of blue (e.g., the black cloud sweeps across the sky), we can reflect this situation by firing a simple rule which is a one-to-one mapping rule, where a class or property name can be translated exactly to another class or property name. Of course, it is available only when such exact translation exists. Precisely, we need to translate `hasColorof' to `hasNotColorof' using the following rule m_1.

$$m_1:\ S(L_{old},\ \tau(?x\ \text{`hasColorOf'}\ ?y)) \rightarrow S(L_{old},\ \text{drop}(0)),\ S(L_{old},\ \tau(?x\ \text{`hasNotColorOf'}\ ?y))$$

Note that m_1 can be read as stating that for any subject x and object y, if L_{old} has a triplet (?x 'hasColorof' ?y) , then the triplet is dropped and the new triplet (?x 'hasNotColorof' ?y) is built. The drop(i) is a built-in procedure, meaning that the i-th matched triplet needs to be dropped. The antecedent of a mapping rule (e.g., $S(L_{\text{old}}, \tau$(?x hasColorOf ?y))) is often called the body of the rule, and the consequence of a

Table 2 Mapping rule definitions (modified from [2])

Symbol	Definition
τ (node, node, node)	A predicate that represents an RDF triplet, where node:=?varname | 'a literal' | number. For example, τ('sky' 'hasColorof' 'blue').
L	A set (i.e., knowledge base) of triplet predicates τ.
b (node, ..., node)	A built-in procedure, where node:=?varname | 'a literal' | number. It returns true when the procedure is invoked successfully. For example, drop(1,2, ..., n).
S	A predicate of the form $S(L, \tau)$ or $S(L,b)$ that is true if $\tau \in L$ or b is invoked successfully in L.
M	A mapping scheme that consists of a set of mapping rules.
M	A mapping rule with the form: $T \rightarrow T'$, where T, T' are sets of S predicates. The rule can be read as follows: If the left-hand side matches (i.e., each $S \in T$ is true), then for each $S(L, \tau) \in T'$, add τ to L, or for each $S(L, b) \in T$ invoke b.

mapping rule (e.g., $S(L_{\text{old}}, \text{drop}(0))$, $S(L_{\text{old}}, \tau(?x\ `\text{hasNotColorOf}'\ ?y))$) is often called the head of the rule.

Now, for m_1, we can define a mapping scheme, $M_{\text{skyNotBlue}}$ and denote the transformation procedure to be $M_{\text{skyNotBlue}} \times L_{\text{old}} \rightarrow L_{\text{new}}$, where the transformation takes $M_{\text{skyNotBlue}}$ and applies the fixed-point of mapping rules to the source document L_{old} and returns a new document L_{new}. This transformation is illustrated in Fig. 4.

The same approach can be used for 1:1 mappings, and extended to *n*:*m* mappings between two RDF documents. Note that the head and body of the rule can contain multiple predicates, and so a rule expressing *n*:*m* translation can be defined. We will use a *n*:*m* mapping case in order to resolve our part ordering motivating scenario. Also, the predicates in the rules can include arithmetical operations and custom built-in operations, so more complex mappings are allowed.

4 Interoperability using semantic rules

In our motivating scenario, we observed that OEM and T1 use different message ontologies, but their message contents are very similar at a higher level. In reality, such a case often happens because each of message ontologies is designed differently based on each business partner's best practice. Nonetheless, we can accomplish the interoperability between OEM and T1 using semantic translations, especially sameAs and joinTo translations. In this section, we will provide a concrete process to resolve these translations by using the logic programming approach discussed in Sect. 3.

4.1 SameAs 1:1 translation

There is a name mismatch between OEM and T1. As shown in Fig. 5, requestTime of OEM and creationDataTime of T1 have the same content Wed Mar 14 21:51:18 PDT 2007, while their element names are different (requestTime ≠ creationDataTime). This case can be addressed using sameAs 1:1 translation, because it is a mapping between two semantically identical elements.

Since our logic programming-based transformation requires interoperating documents to be represented in RDF, we first need to lift the OEMs XML document into the corresponding OEMs RDF document using the transformation mechanism introduced in Sect. 2.4. Figure 6a shows the RDF document which is transformed from requestTime element of the OEM XML document by using the XML to RDF transformation mechanism.

Since OEMs RDF document is ready as a source, we can constitute L_{OEM}, such that it consists of triplets extracted from the OEMs RDF document. We also build a mapping schema M_{sameAs}, that has the following rules:

m_1: $S(L_{OEM}, \tau(?x\ \text{'http://www.oem.com/OEMPartOrder\_requestTime\_PROP'}\ ?y)) \rightarrow S(L_{OEM}, \text{drop}(0)),\ S(L_{OEM},\ (?x\ \text{'http://www.t1.com/T1PartOrder\_CreationDataTime\_PROP'}\ ?y))$

m_2: $S(L_{OEM}, \tau\ (?x\ \text{'http://www.oem.com/OEMPartOrder\_requestTime'}\ ?y)) \rightarrow S(L_{OEM}, \text{drop}(0)),\ S(L_{OEM}, \tau\ (?x\ \text{'http://www.t1.com/T1PartOrder\_creationDataTime'}\ ?y))$

m_3: $S(L_{OEM}, \tau(?x\ \text{'http://www.oem.com/OEMPartOrder\_requestTime\_sValue'}\ ?y)) \rightarrow \text{drop}(0),\ S(L_{OEM}, \text{drop}(0)),\ S(L_{OEM}, \tau\ (?x\ \text{'http://www.t1.com/T1PartOrder creationDataTime sValue'}\ ?y))$

After the conversion, that is, $M_{\text{sameAs}} \times L_{\text{OEM}} \rightarrow L_{\text{T1}}$, we can obtain the T1s RDF document as shown in Fig. 6b. Using the RDF to XML transformation mechanism [15], we can lower the RDF Fig. 6b into XML and then we can finally create T1s XML document, meaning that sameAs 1:1 translation described in Fig. 5 is accomplished.

4.2 JoinTo semantic relation

Besides the name mismatch discussed in Sect. 4.1, there is a structural (granularity) mismatch between OEM and T1. As shown in Fig. 7, part/name of T1 has its content which must be built by concatenating contents of both sender and part/name of OEM. Therefore, it can be called a case of 2:1 mappings. Similar to the sameAs case, we can lift sender and part/name of OEM XML to the corresponding RDF document as shown in Fig. 8a. We already named this case

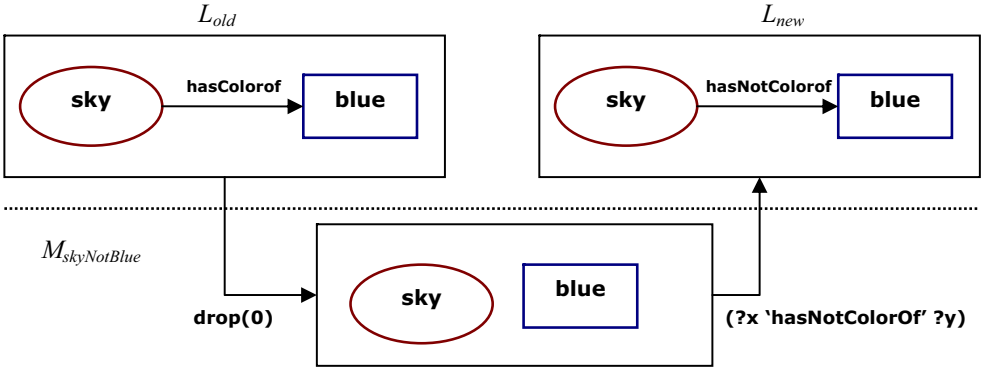

Fig. 4 RDF graph transformation by $M_{\text{skyNotBlue}} \times L_{\text{old}} \rightarrow L_{\text{new}}$

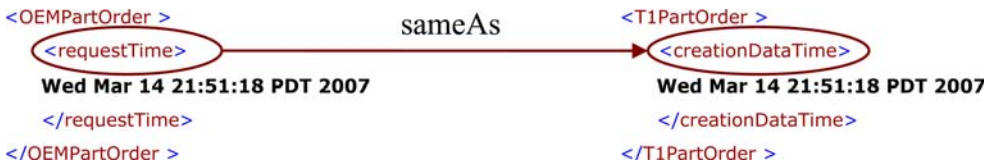

Fig. 5 SameAs semantic relationship between OEM and T1

to be joinTo 2:1 mapping, because two elements are joined to one element.

For the OEMs RDF document, we can constitute L_{OEM} (of course, it is different from L_{OEM} in the previous section). We also build a mapping schema M_{joinTo}, whereby some of the mapping rules are the following:

m_1 : $S(L_{OEM}, \tau$ (?x 'http://www.oem.com/OEMPartOrder_part_PROP' ?y))$\rightarrow S(L_{OEM},$ drop(0)), $S(L_{OEM},$ (?x 'http://www.t1.com/T1PartOrder_part_PROP' ?y)]

m_2 : $S(L_{OEM},$ (?x 'http://www.oem.com/OEMPartOrder_part' ?y))$\rightarrow S(L_{OEM},$ drop(0)), $S(L_{OEM}, \tau$(?x 'http://www.t1.com/T1PartOrder_part' ?y))

m_3 : $S(L_{OEM}, \tau$(?x 'http://www.oem.com/OEMPartOrder_part_name_PROP' ?y))$\rightarrow S(L_{OEM},$ drop(0)), $S(L_{OEM},$ (?x 'http://www.t1.com/T1PartOrder_part_name_PROP' ?y))

m_4: $S(L_{OEM}, \tau$(?x 'http://www.oem.com/OEMPartOrder_part_name' ?y))$\rightarrow S(L_{OEM},$ drop(0)), $S(L_{OEM},$ (?x 'http://www.t1.com/T1_part_name' ?y))

m_5: $S(L_{OEM}, \tau$(?x 'http://www.oem.com/OEMPartOrder_sender_sValue' ?y)), $S(L_{OEM},$ (?i 'http://www.oem.com/OEMPartOrder_part_name_sValue' ?j)), $S(L_{OEM},$ hypenconcat(?y ?j ?z))$\rightarrow S(L_{OEM},$ drop(0,1)), $S(L_{OEM},$ (?i 'http://www.t1.com/T1 part name sValue' ?z))

Note that the body of m_5 has hypenconcat (?y ?j ?z), which is a custom built-in procedure. It aims to concatenate ?y and ?j through dash (-), and assign the concatenated new content to ?z. In our case, OEM and Gear ASM are concatenated, and then OEM-Gear ASM becomes the object (value) with respect to the element of http://www.t1.com/T1_part_name_sValue through this procedure. After the conversion, that is, $M_{\text{joinTo}} \times L_{\text{OEM}} \rightarrow L_{\text{T1}}$, we can obtain the T1s RDF document as shown in the Fig. 8b. We can lower the RDF of Fig. 8b into XML and then create T1s XML document. This implies that joinTo 2:1 mapping is achieved.

5 Implementation and suggestion of a scalable interoperability framework

In this section, we present our demonstrator and how the fundamental knowledge introduced in Sects. 2, 3, 4 became integrated to implement our demonstrator. In addition, we

a

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.oem.com/">
  <OEMPartOrder>
    < OEMPartOrder _requestTime_PROP>
      < OEMPartOrder _ requestTime >
        < OEMPartOrder _ requestTime _sValue>
          Wed Mar 14 21:51:18 PDT 2007
        </ OEMPartOrder _ requestTime _sValue>
      </ OEMPartOrder _ requestTime >
    </ OEMPartOrder _ requestTime _PROP>
  </ OEMPartOrder >
</rdf:RDF>
```

b

```
<rdf:RDF xmlns="http://www.t1.com/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#">
  <rdf:Description rdf:nodeID="A0">
    <rdf:type rdf:resource="http://www.t1.com/T1PartOrder" />
    <T1PartOrder_creationDataTime_PROP rdf:nodeID="A1" />
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <rdf:type rdf:resource=" http://www.t1.com/ T1PartOrder _creationDataTime" />
    < T1PartOrder _creationDataTime_sValue>
      Wed Mar 14 21:51:18 PDT 2007
    </ T1PartOrder _creationDataTime_sValue>
  </rdf:Description>
</rdf:RDF>
```

Fig. 6 RDF transformation of XML messages for OEM and T1. **a** RDF transformation of requestTime element in ODM. **b** RDF transformation of creationdataTime element in T1

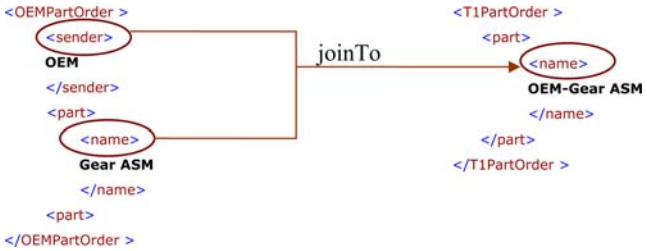

Fig. 7 joinTo semantic relationship between OEM and T1

suggest a scalable interoperability framework to address high implementation complexity that developers would encounter when trying to achieve large-scale interoperability in real world applications.

5.1 Implementation

We have implemented a demonstrator that enables OEM and T1 to realize interoperability using semantic rules between them. For the reasoning engine in support of the logic programming-based RDF message translation, we have used the Jena package [11] that can directly process

a

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.oem.com/">
   <OEMPartOrder>
      < OEMPartOrder_sender_PROP>
         < OEMPartOrder_sender>
           < OEMPartOrder_sender_sValue>
            OEM
           </ OEMPartOrder_sender_sValue>
         </ OEMPartOrder_sender>
      </ OEMPartOrder_sender_PROP>
      < OEMPartOrder_part_PROP>
         < OEMPartOrder_part>
            < OEMPartOrder_part_name_PROP>
               < OEMPartOrder_part_name>
                  < OEMPartOrder_part_name_sValue>
                   Gear ASM
                  </ OEMPartOrder_part_name_sValue >
               </ OEMPartOrder_part_name >
            </ OEMPartOrder_part_name_PROP >
         </ OEMPartOrder_part>
      </ OEMPartOrder_part_PROP>
    </ OEMPartOrder >
</rdf:RDF>
```

b

```
<rdf:RDF   xmlns="http://www.t1.com/"   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#">
   <rdf:Description rdf:nodeID="A0">
        <rdf:type rdf:resource="http://www.t1.com/T1PartOrder" />
        <T1PartOrder_part_PROP rdf:nodeID="A1" />
   </rdf:Description>
   <rdf:Description rdf:nodeID="A1">
        <rdf:type rdf:resource=" http://www.t1.com/ T1PartOrder_part" />
        < T1PartOrder_part_name_PROP rdf:nodeID="A2"/>
   </rdf:Description>
   <rdf:Description rdf:nodeID="A2">
        <rdf:type rdf:resource=" http://www.t1.com/ T1PartOrder_part_name" />
        < T1PartOrder_part_name_sValue> OEM-Gear ASM
        < T1PartOrder_part_name_sValue>
   </rdf:Description>
</rdf:RDF>
```

Fig. 8 RDF transformation of XML messages for OEM and T1. **a** RDF transformation of sender and part/name elements in OEM. **b** RDF transformation of part/name element in T1

and reason our forward chaining pattern matching rules. Since we anticipate the expansion of our system so as to encompass additional business partners in the future, we have architected our demonstrator based on a gateway approach. It indicates that we have developed a semantic gateway in the middle between OEM and T1 systems, which aims to maintain mapping rules in Jena, and to translate messages between partners on request. For the real-time message transaction, we have adopted a Web service based SOA architecture, so that each system is designed to send or receive messages using the SOAP standard. Figure 9 illustrates the overview of message flow between OEM and T1. All three systems (OEM, Semantic Gateway, T1) were implemented in JAVA 1.5, and were deployed as Web applications on a Tomcat. Each system has its own java servlet page (JSP) for providing human interface as GUI.

A detailed step-by-step description on the message flow is as follows:

1. OEM generates a part ordering XML message in its proprietary format, and wraps the message with a SOAP envelope. The SOAP-wrapped message is delivered to a Semantic Gateway through the Web.
2. The Semantic Gateway receives the SOAP message from OEM and extracts its SOAP body, that is, the original OEMs part ordering XML message.
3. Semantic Gateway lifts the original OEMs part ordering XML message to a corresponding RDF document according to the XML to RDF transformation mechanism explained in Sect. 2.4.
4. Semantic Gateway, equipped with the Jena package, fires predefined mapping rules, and translates OEMs RDF document to T1s RDF document.
5. Semantic Gateway lowers T1s RDF document into the corresponding XML file, which is now understandable by T1.
6. Semantic Gateway wraps the resultant XML file into a SOAP envelope, and sends the SOAP-wrapped message to T1.
7. Finally, T1s Web service receives the SOAP message, and extracts the payload which is the T1-understandable part ordering XML file.

Figure 10 shows JSP-based GUIs for OEM, T1, and Semantic Gateway that constitute our demonstrator. The OEM GUI allows the user to place an order by completing the part ordering form, and the T1 GUI allows the user to monitor the history of ordering messages received from customers like OEM. Semantic Gateway keeps track of each step of document conversion flow from when a SOAP-wrapped message arrives to when the corresponding SOAP-wrapped message leaves. In other words, Semantic Gateway records all the documents generated during the transformation process including the original OEMs SOAP-wrapped message, the OEMs RDF document, the T1s RDF document, and the T1-understandable XML document that is the final outcome of this semantic transformation.

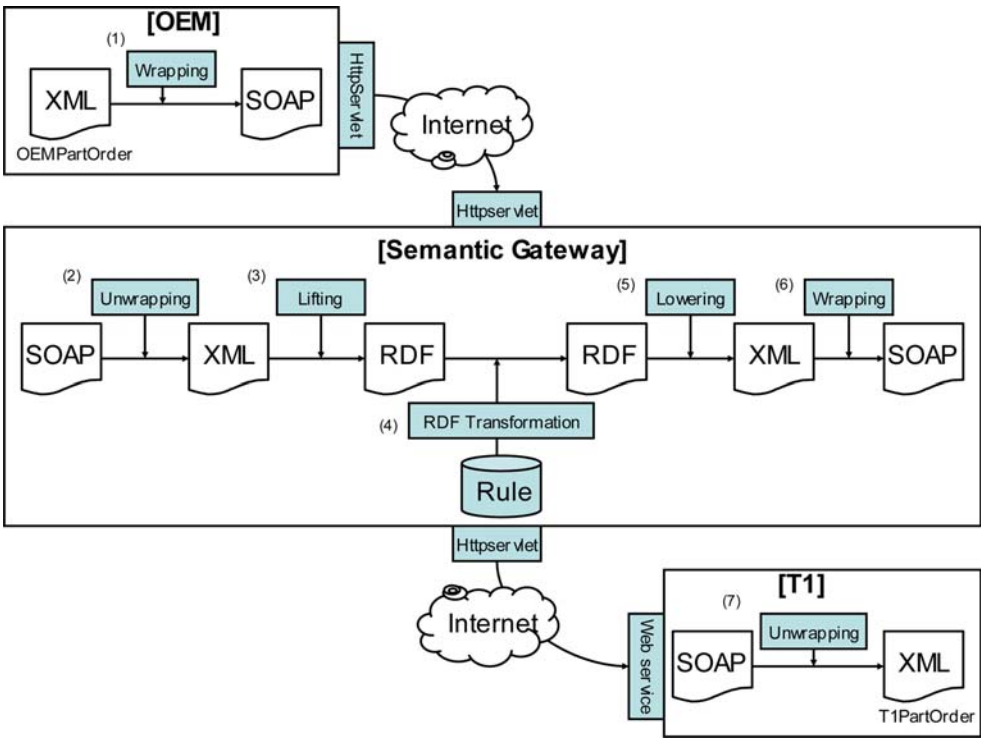

Fig. 9 Overview of the message flow between OEM and T1

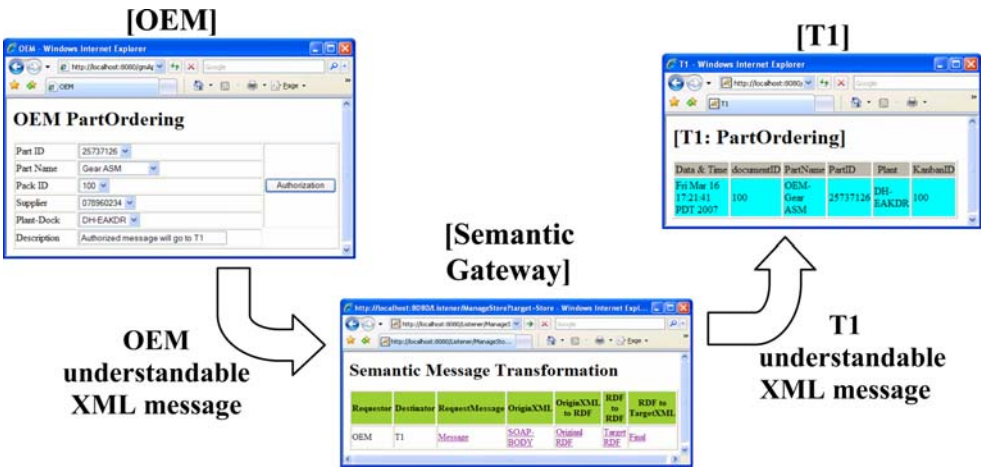

Fig. 10 GUIs for partners in the demonstrator

5.2 Suggestion of a scalable interoperability framework

Although our demonstrator succeeded in showing how to realize interoperability between heterogeneous business partners, the interoperability scenario implemented by our demonstrator is limited in scope. That is, only two partners were involved in the scenario which may not be a realistic scenario. Real world industry scenarios need to consider multiple partners. However, the large scale of interoperability implementation is often limited due to its high complexity. For example, given a supply chain consisting of n number of customers and n number of suppliers, if each customer communicates with n suppliers using rule-mapping adapters like our demonstrator, then the supply chain requires $n \times n$ number of adapters in total. It is manifest that such a system with complexity of n^2 is not scalable because the complexity increases much faster than n increases linearly.

In Sect. 2.2, we introduced the concept of ontology reconciliation, and how the technique can reduce the interoperability complexity by using reference ontologies in a technology-agnostic level abstraction. In this section, we describe the reference ontology (RO)-based semantic reconciliation from the implementation perspective, especially with the assumption that we accept the logic programming based RDF transformation approach. Figure 11 illustrates the RO-based framework, where all mappings among business partners are expressed in terms of the reference ontology. In other words, each partner has two sets of mapping rules such as forward and backward mapping rule sets, where the forward mapping rule set is used to transform partner's proprietary data scheme to a RO-conforming scheme. Reversely, the backward mapping rule set aims to translate the RO-conforming scheme to a partner's proprietary data scheme. This framework results in reduction of interoperability complexity, because each partner needs only two sets of mapping rules, where the interoperability complexity becomes O(n). This implies that when a new partner wants to enter a community, it just considers the interoperability between itself and the community's reference ontology. As a result, the suggested framework can be effective and efficient, when multiple business partners cooperate for achieving their interoperability.

6 Related works

Several related works are found in the literature. Preist et al. [19] presented a demonstrator system which applies semantic Web-service technology to B2B integration, focusing specifically on a logistics supply chain. Their demonstration system is able to cope with all stages of the service life cycle—discovery, service selection, and service execution. The proposed demonstrator system allows a requestor to discover logistics service providers, select appropriate logistics services, coordinate the services to form a composite service chain, and communicate with the service providers using arbitrary protocols through dynamic mediation.

Obitko and Marik [17] reported an approach of expressing mappings between ontologies, and using these mappings for communication between agents. They chose several ontologies that describe transportation domain and

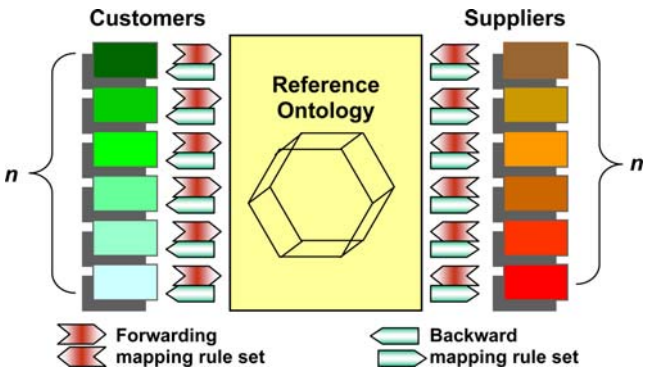

Fig. 11 RO-based framework for semantic reconciliation

created partial mappings between them. The mappings were expressed as matching forward rules.

Michalowski et al. [14] presented a running application, titled in Building Finder, that integrates satellite imagery, geospatial data, and structured and semi-structured data from various online data sources using semantic-Web technologies. Users can query an integrated view of these sources and request Building Finder to accurately superimpose buildings and streets obtained from various sources on satellite imagery. Building Finder is a real example that promises seamless integration of heterogeneous data from distributed sources, letting agents perform sophisticated and detailed data analysis.

ATHENA [1]—Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications—is an integrated project sponsored by the European Commission, aiming to make a major contribution to interoperability by identifying and meeting a set of interrelated business, scientific & technical, and strategic objectives. To deal with data interoperability, ATHENA is working with semantic data transformation as a way to translate information stored in different formats and systems between different enterprises. In the services level, ATHENA adopts the model-driven service oriented architecture to solve the problem of running different applications on different architectures.

Our approach is aligned with [1] and [2] in terms of using the same RDF transformation approach by means of the logic programming approach. In addition to the approaches described in [1] and [2], we focus on several additional issues important for the real-life application, especially for manufacturing industries, including the practical manufacturing industry scenario, the formal introduction of the transformation mechanism, and the SOA based implementation for run-time transformation. We plan to extend our approach by encompassing agent and semantic query techniques presented in [14] and [17], so that our demonstrator can deal with anonymous business partners on the fly, and search for the multiple heterogeneous distributed service information to accomplish the concept of dynamic supply chain.

7 Conclusions

In this paper, we have presented a semantic mediation approach that allows many heterogeneous business partners to achieve seamless business transactions across organizational boundaries and, thus, could eventually realize networked organizations. The semantic mediation approach has been implemented by developing a demonstrator system. The demonstrator itself is not of commercial product quality, and yet we plan to enhance it to be more secure and robust so that it can be deployed in a real world enterprise network environment. Overall, we believe that our demonstrator succeeded in demonstrating the feasibility of our approach to resolve interoperability problems, especially in the manufacturing domain. Based on lessons learned from this project, we expect the use of semantic mediation technologies to become critical to achieve interoperability between heterogeneous manufacturing enterprise networks in the near future.

As future work, there are many directions for improving our current work. We are currently applying our method to more complicated but practical manufacturing operations, such as electronic Kanban inventory management workflow and engineering change order processing. We also focus on addressing scalability issues for large scale interoperability.

Acknowledgements The authors would like to thank Pat Snack, Nenad Ivezic, Boonserm Kulvatunyou, Marko Vujasinovic, Jaewook Kim, Jungyub Woo, and all other researchers having worked for the ATHENA (EU-FP6) B5.10, for stimulating and guiding this work.

References

1. Athena (2004) Athena, European integrated project. http://www.athena-ip.org. Accessed 20 August 2007
2. Bowers S, Delcamre L (2000) Representing and transforming model based information. Proceedings of the 4th European conference on research and advanced technology for digital library (ECDL-2000), Lisbon, Portugal, pp 5–18
3. Chalupsky H (2000) OntoMorph: a translation system for symbolic knowledge. http://www.isi.edu/~hans/ontomorph/presentation/ontomorph.html. Accessed 20 August 2007
4. Chinnici R et al (2006) Web services description language (WSDL) version 2.0. http://www.w3.org/TR/wsdl20. Accessed 20 August 2007
5. Clark J (1999) XSL Transformations (XSLT) version 1.0. W3C recommendation. http://www.w3c.org/TR/xslt. Accessed 20 August 2007
6. Gartner (2002) Semantic web technologies take middleware to the next level. http://www.gartner.com/DisplayDocument?doc_cd=109295. Accessed 20 August 2007
7. Gudgin M et al (2003) SOAP version 1.2. http://www.w3.org/TR/soap12-part1. Accessed 20 August 2007
8. Hameed A, Preece A, Sleeman D (2003) Ontology reconciliation. In: Staab S, Studer R (eds) Handbook on ontologies in information systems. Springer, Berlin Heidelberg New York, pp 231–250
9. Horrocks I, van Harmelen F, Patel-Schneider P (2001) Reference description of the DAML+OIL ontology markup language. Language specification. http://www.daml.org/2001/03/daml+oil-index.html. Accessed 20 August 2007
10. Januszewski K, Monney E (2004) UDDI Spec TC. http://uddi.org/pubs/uddi_v3.htm. Accessed 20 August 2007
11. Jena 2 (2007) A semantic web framework. http://www.hpl.hp.com/semweb/jena2.htm. Accessed 20 August 2007
12. Kotinurmi P (2005) Towards more intelligent business-to-business integration with semantic web service technologies. Proceedings of the CIMRU-DERI-HP research seminar, The Digital Enterprise Research Institute, Galway, Ireland, pp 33–35

13. Lassila O, Swick R (1999) Resource description framework (RDF) model and syntax specification. W3C recommendation. http://www.w3.org/TR/2000/CR-rdf-schema-20000327. Accessed 20 August 2007
14. Michalowski M, Ambite JL, Thakkar S, Tuchinda R, Knoblock CA, Minton S (2004) Retrieving and semantically integrating heterogenous data from the web. IEEE Intell Syst 19(3):72–79
15. Miletic I, Vujasinovic M, Ivezic N, Marjanovic Z (2007) Enabling semantic mediation for business applications: XML-RDF, RDF-XML and XSD-RDFS transformations. Proceedings of the 3rd international conference on interoperability for enterprise software and applications (IESA-07), Madeira Island, Portugal
16. Madnick SE (1995) From VLDB to VMLDB (Very MANY Large Data Bases): dealing with large-scale semantic heterogeneity. Proceedings of the 21st very large data base conference, Zurich, pp 11–16
17. Obitko M, Marik V (2005) Integrating transportation ontologies using semantic web languages. Proceedings of the 2nd international conference on applications of holonic and multi-agent systems, Copenhagen, Denmark, pp 99–110
18. Peer J (2003) A logic programming approach to RDF document and query transformation. Proceedings of the workshop on knowledge transformation for the semantic web at the 15th European conference on artificial intelligence, Lyon, France, pp 115–121
19. Preist C, Esplugas-Cuadrado J, Battle S, Grimm S, Williams S (2005) Automated business-to-business integration of a logistics supply chain using semantic web services technology. Proceedings of the 4th international semantic web conference, Galway, Ireland, pp 987–1001