RosettaNet and Semantic Web Services

Paavo Kotinurmi
[*]Helsinki University of Technology
Finland
paavo.kotinurmi@tkk.fi

Armin Haller
[†]DERI
National University of Ireland, Galway
armin.haller@deri.org

Abstract

In this paper we present a semantic B2B gateway based on the WSMX semantic Service Oriented Architecture to tackle heterogeneities in RosettaNet messages. We develop a rich RosettaNet ontology and use the axiomatised knowledge and rules to resolve data heterogeneities and to unify unit conversions. We define adaptive executable choreographies, which allow a more flexible integration of suppliers using RosettaNet and make it easy to introduce more competition to the supply chain. The solution is justified with a scenario-based evaluation and by analysing RosettaNet specifications and their implementations.

1 Introduction

B2B integrations offer increased speed, less errors and reduced cost of information exchange between business partners. However, integration efforts still suffer from long implementation times and high-costs [Preist et al. (2005)]. This has lead to business models with simple processes, in which long term rigid partnerships are established [Kotinurmi et al. (2006)]. RosettaNet is a prominent standard for B2B integration that represents an agreement on the message exchange patterns, the message content and a secure transportation mechanism between companies operating in the IT and electronics industries. The message content of structurally valid RosettaNet Partner Interface Processes (PIP) is defined by either DTD for the older PIPs or XML Schema for the recently updated ones. However, the interoperability challenges are only partly solved. DTDs and XML Schemas lack expressive power to capture all necessary constraints and do not make all document semantics explicit. Being able to express constraints in machine-interpretable format is expected by RosettaNet experts as well [Damodaran (2004)].

Companies can use the same PIP messages differently as the messages contain many optional elements. Some companies may support only parts of the enumerated values for packaging-units, unit of measurements and currencies that are in the RosettaNet dictionaries. When the number of partners increases, handling these heterogeneities comes increasingly important and point-to-point syntactic transformations using scripts are not the most lucrative option. This can lead to the situation that buyers use only one supplier as the information systems do not easily support more competitive arrangements. The example scenario that we use in this paper highlights this from the buyer's (requesters) role in a collaboration. Quotes are asked from multiple suppliers (service providers) that use the messages differently. Rules handling the data heterogeneity of their response messages are required to decide on the best supplier for a given order. In the interactions the use of RosettaNet PIPs [1] 3A1 for quotes and 3A4 for purchase orders are handled.

We propose a semantic B2B gateway, which builds upon the Web Service Modelling eXecution environment (WSMX) [Haller et al. (2005)]. This solution applies semantic Web Service (SWS) technologies to RosettaNet collaborations. However, we do not imply the use of SWS technologies to the business partners, as the current knowledge of those technologies is low. In our example scenario only the requester uses SWS technologies, the partners simply use current RosettaNet XML interfaces.

The main contributions of this paper are as follows:

- We encode the information exchanged in RosettaNet PIP3A1 messages in a formal ontology. The ontology includes constraints that cannot be represented in current PIP message schemes, such as dependency between fields.
- We define axioms in the ontology, which constrain the interpretations of certain elements in RosettaNet messages (e.g. units of measurement, packaging size)
- We further add domain specific rules to the knowledge base that are applied at run time to

[1] http://www.rosettanet.org/pipdirectory

resolve data heterogeneities in the RosettaNet messages.

- We define and execute a choreography, which allows to easily integrate new partners to our solution and to handle the data heterogeneities introduced.

The paper is structured as follows: first we give a brief introduction to semantic Web Services in section 2. Section 3 presents the architecture of our semantic B2B gateway and describes its components. Section 4 outlines our use case scenario, its design time modelling tasks and presents evaluation results. In section 5 we position our solution to related work. Finally the paper is concluded in section 6.

2 Semantic Web Services

The Web Service Modeling Ontology (WSMO) [Roman et al. (2005)] provides a conceptual model and a language for semantic markup describing all relevant aspects of general services, which are commonly accessible through a web service interface. However, the semantic mark up provided in WSMO can be grounded to any service (e.g. CORBA, GRID etc.) The ultimate goal of such a markup is to enable the (total or partial) automation of tasks (e.g. discovery, selection, composition, mediation, execution and monitoring) involved in both intra- and inter-enterprise integration. The WSMO conceptual model is formalised by the Web Service Modeling Language (WSML) family of ontology languages, used to describe WSMO elements (goals, ontologies, services, mediators). WSML consists of a number of variants based on different logical formalisms. The different variants of the WSML correspond with different levels of logical expressiveness and are both syntactically and semantically layered. In addition, WSMO defines the underlying model for the Web Service Execution Environment (WSMX). WSMX is an integration platform conforming to the principles of a Service Oriented Architecture (SOA) which facilitates the integration between different systems. The integration process is defined by adaptive operational semantics, defining the interactions of middleware services including discovery, mediation, invocation, choreography, repository services, etc. Thus, WSMO, WSML and WSMX form a coherent framework covering all aspects of semantic Web Services (SWS).

Although WSMX aims to allow a dynamic discovery of partners in the collaboration, our focus lies on how SWS technology tackles the data heterogeneities in e-business collaborations. This has two reasons, first current business practice does not consider an integrated discovery and invocation of services. The "discovery" of business partners is conducted when the infrastructure is set up and are commonly based on well-established and long-running business relations. Second, the technology required for a dynamic discovery of rich semantic descriptions is not mature enough in business settings. The reasoners to perform matchmaking only scale on light semantic descriptions of services with very limited expressivity. Thus we omit the functional description of a service (its capability description which is mainly used during the dynamic discovery) and the formalisation of a requester's goal. Instead we avail of the WSMO service interface - a description of the communication patterns according to which a service requester consumes the functionality of the service. The WSMO service choreography [Roman and Scicluna (2006)] is based on the formalism of Abstract State Machines (ASMs) [Gurevich (1994)] and allows to model data exchanged in every state of a service execution. It consists of three core elements (1) a *signature*, (2) a *grounding*, and (3) *transition rules*. The signature describes static parts of state descriptions defined in terms of imported ontologies. Each state has a grounding associated defining how concepts in the ontology are linked with messages. Transition rules express changes of states in the choreography by changing the set of ontology instances (adding, removing and updating instances to the signature ontology). An example of such a choreography description is described in section 4.1.

3 Semantic B2B Gateway

In this section we introduce the architectural consideration taken into account for a semantically enhanced RosettaNet B2B integration. We detail the implementation of a generic semantic B2B gateway, being truly agnostic about whether the partners use WSML messages or any ordinary schema language. Our solution is based upon the WSMX platform. WSMX follows a component based design with adaptive operational semantics. Every component provides some service and can be orchestrated within WSMX as required. A semantic RosettaNet B2B Gateway as defined in this paper relies on the following four components:

- The *Adapter Framework* consists of B2B adapters to lift and lower XML instances in the messages sent from the partners to WSML class hierarchies and a middleware adapter connecting to the back-end applications of the requester.

- The *Choreography Engine* executing external service by evaluating a WSMO choreography description. The actual invocation of the service and the choice of the right transport level is performed by the *Communication Manager*.
- The *Resource Manager* constitutes the knowledge base and its interface and provides access to the local component repositories that store definitions of WSMO entities. In our case, these are the domain ontology, the domain specific rules stored in our knowledge base and the choreography specifications stored in the repository.
- The *Reasoner* allows to perform queries on the knowledge base of facts and rules to determine the answer by logical deduction.

WSMX is shipped with these standard components. However, the adapter instances have to be built based on the requirements from RosettaNet interfaces with service providers. The complete architecture of our solution is depicted in figure 1.

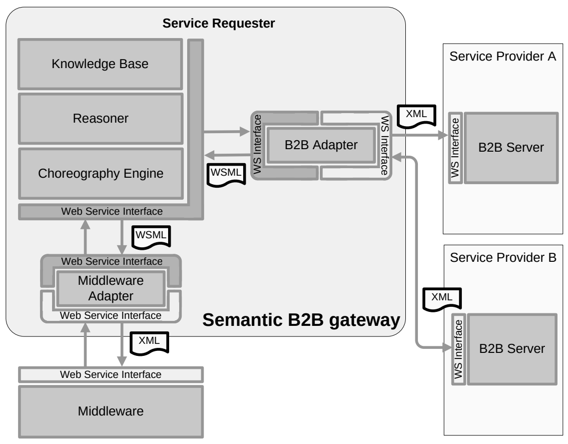

Figure 1: Overview of B2B gateway architecture

3.1 Resource Manager

The resource manager coordinates the access to the repositories of the different components in the architecture and as such gives the illusion of a central repository. This repository acts as a knowledge base and has to be populated with the ontology artifacts used in the different components in our B2B gateway. Most importantly, at least one domain ontology to formally capture the message content exchanged in any of our RosettaNet collaborations is required. Ideally existing *domain ontologies* should be reused. Since an industry wide recognised ontology for business messages has not been established yet, we built the ontology ourselves. It is based on the RosettaNet specification, which itself can be regarded as a lightweight ontology.

However, the task is to not only simply translate the DTDs or XML Schemas to a richer and formal language (i.e. WSML), but to model all the constraints on the semantics of the business documents. We include constraints which are not expressible in DTD or XML Schema and capture implicit knowledge provided in the RosettaNet message guidelines and accompanying documentation to facilitate the promised benefits of automatically resolving data heterogeneities.

An example of the first, a cardinality constraint not expressible in current RosettaNet messages schemas are the constraints imposed on the “BusinessDescription” element used in all RosettaNet PIPs. The “BusinessDescription” element includes business properties that describe a business identity and its location. At least one of the possible three subelements “businessName”, “GlobalBusinessIdentifier” or “PartnerBusinessIdentification” has to be provided in order to be a valid PIP. It is easy to include such constraints in our WSML ontology as shown in listing 1:

```
concept businessDescription
    businessname ofType (0 1) businessName
    globalbusinessidentifier ofType (0 1)
            globalBusinessIdentifier
    partnerbusinessidentification ofType
            partnerBusinessIdentification
    nfp
        dc#relation hasValue validBusinessDescription
    endnfp
axiom validBusinessDescription
    definedBy
        forall ?x ( ?x memberOf businessDescription implies
            ?y memberOf businessName or
            ?y memberOf globalbusinessidentifier or
            ?y memberOf partnerbusinessidentification).
```

Listing 1: Cardinality Constraints spanning multiple elements

Listing 2 shows examples of implicit knowledge we have captured in our RosettaNet ontology. For example, RosettaNet defines a list of 335 possible values for unit of measurements, with the logical relationships between values unspecified. We made these logical relations explicit and included these axiomatisations in our ontology. The first axiom “resolveMeasurementUnitType” in listing 2 shows how the measurement units defined with natural language text in the RosettaNet PIPs can be resolved to its corresponding numerical value. The numerical value can subsequently be used for further calculations (c.f. section 4.1). The second part of the listing defines a function used to relate a kilogram value to its equivalent pound value.

```
axiom resolveMeasurementUnitType
  definedBy
    forall ?x(?x[globalProductUnitOfMeasurementCode
           hasValue ”dozen”] memberOf quoteLineItem implies
           ?x[globalProductUnitOfMeasurementCode hasValue
            ”12”]).
    forall ?y(?y[globalProductUnitOfMeasurementCode
           hasValue ”10−pack”] memberOf quoteLineItem
           implies ?y[globalProductUnitOfMeasurementCode
           hasValue ”10”]).

relation poundKilo (ofType productQuantity, ofType
         productQuantity)
  nfp
    dc#relation hasValue poundKiloDependency
  endnfp

axiom poundKiloDependency
  definedBy
    forall ?x,?y (
      poundKilo(?x,?y) equivalent
        ?x memberOf productQuantity and
        ?x[globalProductUnitOfMeasureCode hasValue
             ”Kilogram”]
          memberOf quoteLineItem and
        ?y memberOf productQuantity and
        ?y[globalProductUnitOfMeasureCode hasValue
             ”Pound”]
          memberOf quoteLineItem and
        ?x = wsml#numericDivide(?y,?x,0.45359237)).
```

Listing 2: Definitional facts in the domain ontology

3.2 Adapter Framework

The adapter framework provides transformation functionality for every non-WSML message sent to the B2B gateway. In our scenario, two adapter instances are required. One for every RosettaNet collaboration partner, who still use XML Schema or DTD based RosettaNet messages and one to connect to the middleware system, providing access to the back-end applications.

The first adapter receives every RosettaNet message and applies the lifting or lowering rules defined in the adapter to map every message instance based on its source schema (in our case XML-Schema) to a WSML instance based on its ontological schema. Listing 3 shows one incoming XML instance and listing 4 its corresponding WSML instance after the transformation rules were applied[2].

It has to be noted that the adapters act for WSMX as the actual service provider. The adapter functionality (i.e. the Web Service endpoint of the adapter) is registered as a service with WSMX and not the RosettaNet service of the partner itself. Thus the adapter is further responsible to execute the correct endpoint of the partner service. However, adapters only perform data manipulation, their interface behaviour replicates the behaviour of the underlying partner service.

The second adapter instance receives every internal message sent from the middleware bus, representing a gateway to every non Web Service compliant back-end application.

[2]More details on the lifting and lowering of XML Schema to WSML can be found in Kotinurmi et al. (2006).

3.3 Reasoner

The Reasoner is required to perform query answering operations on the knowledge base, which itself is populated with domain ontologies and domain specific rules at design time and ontology instances at run time. Our reasoner has to handle the WSML-Rule variant and should have built-in predicates for handling basic data-types, basic arithmetic functions as well as basic comparison operators.

Dedicated reasoners for the different WSML variants are still under development. However, there are a number of implementations based on existing reasoners for WSML-Rule available: one is based on the $\mathcal{FLORA}$-2 reasoner, which is also used in the mediation component of WSMX. $\mathcal{FLORA}$-2 is a rule-based knowledge representation formalism and a reasoner for this formalism. It is based on F-Logic [Kifer et al. (1995)], a frame-based logic, WSML is also based upon. Further, there are translations from WSML-Rule to F-Logic available. The reasoner includes a Web Service interface which is used by the service requester to perform queries on the knowledge base. Messages received from the back-end systems of the requester are sent through the middleware. Thus, the middleware adapter has to encode rules how to translate requests from these back end systems to queries on the knowledge base and return the results. This is out of the scope of this paper and will be addressed in future work.

3.4 Choreography Engine

The Choreography Engine as part of WSMX is responsible to control the sending and receiving of messages and update the state of the choreography according to the message content. In contrast to the common use of choreography languages as non-executable descriptions, the engine operates and executes WSMO choreography description such as the one defined in section 4.1.

As mentioned earlier WSMO choreographies are modelled as Abstract State Machines and are processed using standard algorithms during runtime. The state in the machine is represented by ontology

```
        <QuantitySchedule>
          <ProductQuantity>204</ProductQuantity>
        </QuantitySchedule>
  <GlobalProductUnitOfMeasureCode>dozen
  </GlobalProductUnitOfMeasureCode>
  <SubstituteProductReference>
    <GlobalProductSubstitutionReasonCode>Better product
    </GlobalProductSubstitutionReasonCode>
  </SubstituteProductReference>
  <FinancialAmount>
    <GlobalCurrencyCode>USD
    </GlobalCurrencyCode>
    <MonetaryAmount>198
    </MonetaryAmount>
  </FinancialAmount>
```

Listing 3: XML message instance

```
instance QuoteLineItem1 memberOf rfq#quoteLineItem
  rfq#globalProductUnitOfMeasurementCode hasValue ”dozen”
instance quantitySchedule1 memberOf
    core#quantitySchedule
  core#productQuantity hasValue ”204”
instance substituteProductReference1 memberOf
    core#substituteProductReference
  core#GlobalProductSubstitutionReasonCode
    hasValue ”Better product”
instance totalPrice1 memberOf core#totalPrice
  core#financialAmount hasValue FinancialAmountTot
instance FinancialAmountTot memberOf
    core#FinancialAmount
  core#globalCurrencyCode hasValue USD
  core#monetaryAmount hasValue ”198”
```

Listing 4: and its lifted WSML instance

instances. According to the instance data, a transition rule is selected from the rule base within a choreography. The consequent of the rule is interpreted and the ontology instance is modified accordingly. It is the responsibility of the Choreography Engine to maintain the state of a conversation and to take the correct action when that state is updated. For example, the update of the state of a choreography instance may be the result of a message received from a service provider.

Since WSMO choreography descriptions are expressed in a constraint-based fashion, they are very suitable to be changed and extended once new partners are introduced into a collaboration. It is easy to include transition rules triggering on certain parts of a message sent only by one partner to introduce partner specific data handling (c.f. section 4.1.

4 Evaluation

In our example scenario, we discuss the handling of heterogeneous partners engaged in quoting for a product with a PIP 3A1 Request for Quote (RFQ). We describe the collaboration set up and orchestration through examples from the requester's (buyer's) point-of-view. Then we discuss evaluation results through analysing existing PIP message specifications and example instances and the general benefits from our example scenario.

4.1 Collaboration Set Up

In order to retrieve RFQ messages from different partners at run-time and still being able to process their response messages, the requester has to set up **mapping rules** and define a **choreography description** in the B2B gateway at design time.

Mapping rules developed in the Collaboration Set-Up phase capture any data heterogeneity that is not resolved by the definitional facts in the domain ontology (c.f. section 3.1).

These domain specific rules (conversion relations in our case) define how attribute values in the different WSML instances can be transformed. One such example is given in listing 5. It defines a function to calculate the unit price by taking the “financialAmount” and “productQuantity” given in the RosettaNet ontology instance. This rule can be used by the requester to compare the prices of two or more partners. The “financialAmount” in a PIP3A1 RFQ can refer to different quantities of the product. We made the different packaging sizes and its corresponding value explicit in the ontology as described earlier in section 3.1. Now the requester can query the knowledge base to automatically compare the prices provided as “financialAmount” by the partners transparently on a unit basis.

```
relation unitPrice (ofType financialAmount, ofType
      productQuantity, ofType decimal)
  nfp
    dc#relation hasValue unitPriceDependency
  endnfp

axiom unitPriceDependency
  definedBy
    forall ?x,?y,?z (
      unitPrice(?x,?y,?z) equivalent
        ?x memberOf financialAmount and
        ?y memberOf productQuantity and
        ?z = wsml#numericDivide(?z,?x,?y)).
```

Listing 5: Definition of a conversion relation

Next the choreography description has to be defined. In order to allow a collaboration with its suppliers, the choreography interface of the requester in our scenario has to be compliant with the interface behaviours of the partners providing a RFQ response. Since all suppliers in our Supply Chain use RosettaNet, there is already agreement on the message exchange patterns. However, there are still mismatches on the messages sent and received in the collaboration. Thus, we introduce rules in the choreography (c.f. listing 6) of the requester handling data mismatches in the RosettaNet messages.

It has to be noted that the model defined in listing 6 can actually be regarded as an orchestration in terms of the WSMO conceptual model. Orchestrations in WSMO are modelled in the same way as choreographies. In fact the only difference is on the conceptual level. Whereas WSMO choreographies describe the interface behaviour of one service, orchestrations describe how a service makes use of other WSMO services or goals in order to achieve its capability. In our case, we alter the interface behaviour of the requester by introducing rules, which are for example calling a currency conversion service of an external party. This information in fact would not be part of the choreography description. However, in the WSMX system it is executed in the same engine. Only when the requester wants to publish its interface behaviour to a partner, it is relevant to decide on the abstraction level of what parts of the orchestration he wants to publish.

An extract[3] of such a choreography is shown in listing 6. Please note that the "//..." symbol denotes parts omitted in the listing. The namespace declarations in the first lines of a WSMO choreography definition are also omitted in the listing.

```
31  choreography
32    stateSignature
33      importsOntology {
34        _"http://www.wsmx.org/ontologies/rosetta/coreelements",
35        _"http://www.m3pe.org/ontologies/rosetta/CTRLASM"
36        }
37      out rfq#Pip3A1RFQRequest withGrounding _"http://example
              .org/webServices#wsdl.interface(ServicePortType/
              RFQ/Out)"
38      out curr#currConvRequest withGrounding _"http://www.
              webcontinuum.net/webservices/ccydemo.wsdl#"
39      in rfq#Pip3A1RFQResponse withGrounding _"http://
              example.org/webServices#wsdl.interface(
              ServicePortType/RFQ/In)"
40    transitionRules
41      if (?Pip3A1RequestForQuoteRequest[
42        fromRole hasValue ?fromRole,
43        globalDocumentFunctionCode hasValue
44          ?globalDocumentFunctionCode,
45        quote hasValue ?quote,
46            thisDocumentGenerationDate hasValue
47          ?thisDocumentGenerationDate,
48            thisDocumentIdentifier hasValue ?
                  thisDocumentIdentifier,
49            toRole hasValue ?toRole
50      ] memberOf rfq#Pip3A1RFQRequest) and
51      //...
171       update(?controlledState[currentState hasValue 1]
              memberOf ctrlasm#controlledState)
172     endIf
173
174     if (?controlledState[
175       currentState hasValue 1
176       ] memberOf ctrlasm#controlledState) and
177       exists ?Pip3A1RequestForQuoteRespond
178         (?Pip3A1RequestForQuoteRespond memberOf rfq#
              Pip3A1RFQResponse) and
179       //...
357       update(?controlledState[currentState hasValue 2]
              memberOf ctrlasm#controlledState)
358     endIf
359
360     if (?controlledState[
361       currentState hasValue 2
362       ] memberOf ctrlasm#controlledState) and
363       exists ?globalProductSubstitutionReasonCode,
364           ?productIdentification
365         (?substituteProductReference[
366             globalProductSubstitutionReasonCode hasValue
367           ?globalProductSubstitutionReasonCode,
368             productIdentification hasValue ?productIdentification
369             ] memberOf core#substituteProductReference) then
370             add(_# memberOf rfq#Pip3A1RFQRequest)
371         endIf
372
373     if (?controlledState[
374       currentState hasValue 2
375       ] memberOf ctrlasm#controlledState) and
376         exists ?globalCurrencyCode, ?monetaryAmount
377         (?totalPrice[
378           globalCurrencyCode hasValue ?globalCurrencyCode,
379           monetaryAmount hasValue "USD"
380         ] memberOf rfq#totalPrice) then
381             add(_# memberOf curr#currConvRequest)
382         endIf
```

Listing 6: Choreography in WSML

As described in section 2, a WSMO choreography definition consists of a state signature (c.f. line number 32-39 in listing 6), which imports one or possibly many ontologies and transition rules (c.f. line number 40-382), which are basic operations, such as adding, removing and updating on the instance data of the signature ontology. In our example we import two ontologies, the message ontology capturing concepts in RosettaNet messages and a control State ASM ontology, which allows us to define termination and to impose an explicitly modelled control flow for certain parts of the choreography.

The transition rules in listing 6 starting at line 39 capture only a small number of heterogeneities possibly occurring in a RFQ collaboration. New rules can be added when new partners are introduced into the scenario with different data heterogeneities.

The first transition rule (lines 41-172) defines the ontology schema of the message sent by the buyer A (the mode "out" in the "stateSignature" states the passing direction, thus that the message is sent) and that the "currentState" of the machine is updated to the value "1" after a successful transmission of the message. The grounding only defines one WSDL interfaces. However, since the requesters wants to get quote response from multiple providers, the actual grounding list would include more endpoints.

The second rule (line 174-358) checks the control state, to ensure that the "Pip3A1RFQRequest" message was successfully sent and that the

[3]The full listing can be found at: http://www.m3pe.org/ontologies/rosettaNet/

"Pip3A1RequestForQuoteRespond" is received. The constraints in the antecedent of the rule act as a schema validation in the sense that the rule only triggers if the message returned by a partner includes all required fields. For space reasons these parts of the choreography (between lines 179-357) are not shown in listing 6.

The last two transition rules only trigger on a part of the message instance which differs between suppliers. These are rules the requester includes to anticipate possible data heterogeneities. If the requester knows that some partners will provide the amount in USD, the fourth transition rule (lines 373-382) ensures that a currency conversion service is used to calculate the value in Euro. The third transition rule (lines 360-371) fires if a partner provides a substitute product. It results in sending out a new RFQ request.

4.2 Evaluation Results

The evaluation is accomplished by analysing RosettaNet PIPs, real instances obtained from companies and by using a descriptive scenario to demonstrate the utility of our solutions.

We first took a sample 56 PIP message specifications representing 29,5 % of the total 190 messages. The PIPs cover product information, order management, inventory management and manufacturing clusters in RosettaNet classification. Measuring units were used in 24 (43%) and currency information in 28 (50 %) of the PIP messages in the sample. Moreover, we analysed two production message instances and their processing instructions. The PIPs were PIP 3A4 Request Purchase Order and 2A13 Distribute Material Composition Information. The instances use only a part of the whole PIP data model and most of the optional elements are not used. Both messages used unit of measure information, but only one contained currency information. Both supported only kilograms as mass units and Euros as currencies. In both cases, the PIP had just been taken into use with a couple of partners.

The axiomatised message semantics and mapping rules introduced in this paper would have both wide usage across different PIPs and that current real-life XML-based integrations lack this kind of functionality motivates the need for this kinds of solutions. This also provides practical examples of the benefits of SWS technologies to current implementations. Although axiomatised knowledge can also be represented in XSLT scripts or custom codings, the main advantage of expressing these kinds of definitional facts in an ontology is its reusability. The "pound-Kilo" relation captures a conversion that is generally applicable across a wide range of PIP interactions. This reuse is a major advantage over scripts, which are typically rewritten for every schema conversion causing point-to-point integrations that are hard to maintain and test that validation covers the necessary constraints.

The example scenario discussed throughout the paper on quoting and purchasing highlights the requesters need to save on purchasing[4]. Having suppliers from different countries brings heterogeneities as the partners are likely to use different currencies or other measuring or packaging units. Benefits for the buyer result from decreased costs of purchasing as the best value deals can be selected based on best quotes. The suppliers benefit from being able to easier integrate to the buyer without having to make potentially costly changes to their current integration interfaces. In addition, the heterogeneities of using different standards, such as UBL, are easier handled by lifting them to ontologies where mappings can easily be performed.

5 Related Work

There are a number of papers discussing the use of SWS to enhance current B2B standards. Some concentrate on ontologising B2B standards. Foxvog and Bussler (2005) describe how EDI X12 can be presented using WSML, OWL and CycL ontology languages. The paper focuses on the issues encountered when building a general purpose B2B ontology, but does not provide an architecture or implementation. Anicic et al. (2006) present how two XML Schema-based automotive B2B standards are lifted using XSLT to OWL-based ontology. They use a two-phase design and run-time approach similar to our's. The paper is based on different B2B standards and focuses only on the lifting and lowering to the ontology level.

Others apply SWS technologies to B2B integrations. Preist et al. (2005) presented a solution covering all phases of a B2B integration life-cycle.The paper addresses the lifting and lowering of RosettaNet XML messages to ontologies, but no richer knowledge is formalised or used on the ontological level. Trastour et al. (2003b,a) augment RosettaNet PIPs with partner-specific DAML+OIL constraints and use agent technologies to automatically propose modifications if partners use messages differently. Their

[4]See http://www.m3pe.org/ontologies/rosettaNet/ for complete examples of XML instances from which the heterogeneities can be automatically solved.

approach of accepting RosettaNet in its current form and lifting to semantic languages is similar to ours, but we go further by axiomatising implicit knowledge and by providing mappings to resolve data heterogeneities.

6 Conclusion

Our semantic B2B gateway allows a buyer to tackle heterogeneities in RosettaNet interactions. The solution applies axiomatised knowledge and rules to resolve data heterogeneities and to unify various unit conversions using functions to capture definitional facts, such as the relation between pounds and kilograms. Such relations between different enumerations are not specified by RosettaNet. The conversions have potential use in significant portion of the 190 RosettaNet PIP messages as shown in the evaluation of the PIP schemas and the current observed integrations lack the support for such heterogeneity. We further defined adaptive executable choreographies, which allow a more flexible integration of suppliers and make it easy to introduce more competition to the supply chain.

Our future work includes further extending and testing our solution and developing an adapter to the reasoning component, which translates the requests sent from the back-end systems of the requester to queries on the knowledge base and returns the results to the applications.

Acknowledgement

The authors thank Eyal Oren and Tomas Vitvar for fruitful discussions leading to the ideas of this paper. This material is based upon works supported by the Finnish Funding Agency for Technology and Innovation (Tekes) and the Graduate School for Electronic Business and Software Industry and the Science Foundation Ireland under Grant No. SFI/04/BR/CS0694.

References

Nenad Anicic, Nenad Ivezic, and Albert Jones. An Architecture for Semantic Enterprise Application Integration Standards. In *Interoperability of Enterprise Software and Applications*, pages 25–34. Springer, 2006. ISBN 1-84628-151-2.

Suresh Damodaran. B2b integration over the internet with xml: Rosettanet successes and challenges. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 188–195, 2004.

Douglas Foxvog and Christoph Bussler. Ontologizing EDI: First Steps and Initial Experience. In *Int. Workshop on Data Engineering Issues in E-Commerce and Services*, pages 49–58, 2005.

Yuri Gurevich. Evolving algebras 1993: Lipari Guide. In Egon Börger, editor, *Specification and Validation Methods*, pages 9–37. Oxford University Press, 1994.

Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler. WSMX – A Semantic Service-Oriented Architecture. In *Proc. of the 3rd Int. Conf. on Web Services*, pages 321 – 328. IEEE Computer Society, 2005.

Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

Paavo Kotinurmi, Tomas Vitvar, Armin Haller, Ray Boran, and Aidan Richardson. Semantic web services enabled b2b integration. In *Int. Workshop on Data Engineering Issues in E-Commerce and Services*, June 2006.

Chris Preist, Javier Esplugas Cuadrado, Steve Battle, Stuart Williams, and Stephan Grimm. Automated Business-to-Business Integration of a Logistics Supply Chain using Semantic Web Services Technology. In *Proc. of 4th Int. Semantic Web Conference*, 2005.

Dumitru Roman and James Scicluna. Ontology-based choreography of wsmo services. Wsmo final draft v0.3, DERI, 2006. `http://www.wsmo.org/TR/d14/v0.3/`.

Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubn Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontologies*, 1(1):77 – 106, 2005.

David Trastour, Claudio Bartolini, and Chris Preist. Semantic Web support for the business-to-business e-commerce pre-contractual lifecycle. *Computer Networks*, 42(5):661–673, 2003a.

David Trastour, Chris Preist, and Derek Coleman. Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches. In *Proc. of the Int. Enterprise Distributed Object Computing Conference*, pages 222–231, 2003b.