Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches[1]

David Trastour, Chris Preist, Derek Coleman
HP Laboratories Bristol
HPL-2003-173
November 18th, 2003*

semantic web, interoperability, DAML+OIL, XML schema

Setting up electronic Business-to-Business relationships is time-consuming and costly. It has been eased to a certain extent by standards such as RosettaNet, which use XML and XML Schema technologies to define standardised syntax of messages used in interactions. However, this standardisation has necessarily maintained some flexibility to allow companies with different internal processes to comply with the standard. Furthermore, the standard is syntactic, rather than semantic. Semantic constraints on interactions are currently represented informally. In this paper, we describe an application of Semantic Web technology to enhance RosettaNet and further reduce cost and time. Businesses can represent the possible ways they are able to interact as semantic and syntactic constraints. Two businesses can determine if they are able to interact without altering their business process by sharing constraints, and finding if the overall set is satisfiable. If it is not, they can use the data to determine what changes need to be made to their business processes. They can also use the other business' constraints to verify or generate documents which meet the constraints, and so are usable by the other business. The system integrates with current RosettaNet standards and tools through the use of a translation suite able to transform XML Schema into DAML+OIL and XML into RDF.

[1] 7th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, Sept 16-19th, 2003

Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches

David Trastour
HP Labs
Filton Road , Stoke Gifford
Bristol, BS32 8QZ
United Kingdom

Chris Preist
HP Labs
Filton Road , Stoke Gifford
Bristol, BS32 8QZ
United Kingdom

Derek Coleman
RosettaNet/HP
19111 Pruneridge Ave.
Cupertino, CA 95014
United States

Abstract

Setting up electronic Business-to-Business relationships is time-consuming and costly. It has been eased to a certain extent by standards such as RosettaNet, which use XML and XML Schema technologies to define standardised syntax of messages used in interactions. However, this standardisation has necessarily maintained some flexibility to allow companies with different internal processes to comply with the standard. Furthermore, the standard is syntactic, rather than semantic. Semantic constraints on interactions are currently represented informally.

In this paper, we describe an application of Semantic Web technology to enhance RosettaNet and further reduce cost and time. Businesses can represent the possible ways they are able to interact as semantic and syntactic constraints. Two businesses can determine if they are able to interact without altering their business process by sharing constraints, and finding if the overall set is satisfiable. If it is not, they can use the data to determine what changes need to be made to their business processes. They can also use the other business' constraints to verify or generate documents which meet the constraints, and so are usable by the other business. The system integrates with current RosettaNet standards and tools through the use of a translation suite able to transform XML Schema into DAML+OIL and XML into RDF.

1 Introduction

Despite the bursting of the dot com bubble, electronic commerce continues to be an increasingly important aspect of the economy. To the general public, the most visible face is the increasing number of Business-to-Consumer interactions available via the web. However, the majority of economic transactions occur between businesses, making the Business-to-Business (B2B) aspect of electronic commerce a significantly larger market. Historically, business relationships have been long-term. When such a relationship has been established, EDI (Electronic Data Interchange) technology can be used to automate certain straightforward interactions between the business partners – for example, the purchase of goods at a pre-agreed price, and the delivery of them. Setting up the EDI system requires the two parties to agree on interaction protocols and message formats, and to implement a messaging system that meets these agreements – often over a Virtual Private Network. This can be a time-consuming and costly process.

RosettaNet [27] is an industrial consortium which aims to make this process cheaper and more straightforward, by using XML [4] messaging technology transported over the World Wide Web. It does this by standardising the format, content and sequence of messages between partners for a variety of possible interactions which companies can use in B2B relationships. Hence, companies do not need to go through a lengthy negotiation to specify the way in which they are going to interact. Instead, they simply need to agree on which standard interaction to use. Standardisation also speeds up the development process: products such as WebMethods Trading Networks Server include software libraries and XML templates supporting RosettaNet interactions.

This standardisation effort has substantially reduced the cost and time of setting up a B2B relationship. However, because it is based on XML technology, the tools provided are primarily syntactic. In this paper, we describe an application of Semantic Web technology to enhance RosettaNet and further reduce cost and time. Businesses can represent the possible ways they are able to interact as semantic constraints. Two businesses can determine if they are able to interact without altering their business process by sharing constraints, and finding if the overall set is satisfiable. They can use the other business' constraints to generate documents which meet the constraints, and so are usable by the other business.

This application is evolutionary, rather than revolutionary. It accepts existing RosettaNet design decisions and tools as they currently are, rather than requiring modification or re-design of these. As a result of this, it is more likely to be rapidly accepted and adopted by the current B2B developer community. This means it must use Semantic Web technology in a relatively conservative way. We believe that by doing this the developer community will become more familiar with the basic ideas, allowing more radical approaches to be taken in the future.

The structure of this paper is as follows. In section 2 we give an overview of RosettaNet and present some of the problems that implementors encounter when they deploy RosettaNet solutions. In section 3 we describe the Nile system which aims at solving these problems with Semantic Web techniques. In particular we describe the main components of this system, the Nile translation suite and the Constraint Knowledge Base. In section 4 we describe a typical use of the Nile system and show how it solves the problems exposed in section 2. We then discuss related work (section 5) and we conclude presenting our future work intentions (section 6).

2 RosettaNet Now

RosettaNet standards have gone a long way to ease the process of setting up and executing long-term B2B relationships via the web. The key concept used to do this is the Partner Interface Process (PIP). PIPs are used to define standard ways of interacting between companies to carry out a specified task. They define the aspects of a business process which are common to the two parties, but place no constraints on how the internal processes implement these common aspects. A PIP specification defines the flow of message documents which will take place during an interaction, and also specifies the format of the messages. A message format is defined through 'message guidelines' documentation, and an XML DTD describing the syntactic structure a message should have.

Hence, in theory, all businesses have to do to set up a new partnership is to agree on which PIPs to use, and implement the PIPs according to their specification. However, as different businesses can have different back-end processes, some flexibility within the standards is necessary to enable all businesses to satisfy it. For example, one business may normally represent dates on invoices using ISO 8601 format (YYYY-MM-DD) while another may use UK common practice (DD/MM/YYYY). One business may expect the account details of the buyer on an invoice, while another may not. To allow differences such as these, PIP definitions often make use of generic datatypes (such as strings or integers) and include optional fields or fields with unbounded cardinalities. As a result of this flexibility, there is no guarantee that two RosettaNet compliant companies will be able to communicate with each other: different business practices or back-end systems may impose different conditions on the presence of some information or on its format. Because of this, it is necessary to reconcile the different processes used by two companies which intend to interact via RosettaNet. There is some flexibility in the way in which a PIP can be implemented, and it is necessary that interacting parties agree as to the specific implementation chosen. This process of reconciliation is currently carried out off-line, using spreadsheets to document decisions. Developers then implement these decisions as they encode the PIPs. This can be a very time-consuming process, meaning that it can take many months to create a new RosettaNet partnership. Hence interoperability, one of the advantages of standardisation, is sacrificed in favour of flexibility.

RosettaNet are currently developing Next Generation PIPs [28] in an attempt to produce specifications than are more formal than the message guidelines used in the current standards. For each Next Generation PIP, RosettaNet specifies a UML class diagram [24] and XML Schemas [32, 3] that replace the XML DTDs. The UML class diagram defines the business objects – such as financial documents or purchase requests – that are used in the PIP. To encourage reuse across PIPs, RosettaNet defines a domain model, i.e. a set of base classes that can be reused or subclassed in the UML class diagrams. The XML Schemas define what makes an XML document a syntactically valid PIP document. XML Schemas are currently defined manually from the UML class diagram.

Having an explicit machine-readable representation of the constraints imposed by a PIP makes setting up a partnership quicker and easier. Reconciliation can take place by agreeing a set of further constraints on each XML Schema within the PIP. Furthermore, having the agreed document structure specified in this format allows the developers to use tools such as Contivo [8] to rapidly automate the process of document generation. However, this approach has several disadvantages:

- The constraints that XML Schema are able to represent are mainly constraints on the syntax, not the semantics, of documents. This means certain constraints which appear in a PIP specification which cannot be represented in the XML Schema. A typical example of this sort of constraint is a dependency between fields, for instance the presence of a field implying a cardinality constraint on another field. RosettaNet is using OCL [23] to represent such constraints in the definition of Next Generation PIPs. Currently, these are documented as comments within the XML Schemas.

- As seen earlier, a company's business processes impose constraints on the deployment of PIPs. Some of these constraints are of syntactic nature and can usually be captured in an XML Schema – which must be more specific than the PIP XML Schema. Some may be of semantic nature and so cannot be expressed in XML Schema. Companies deploying RosettaNet PIPs usually document these constraints in the form of spreadsheets that are manually created for the purpose of one deployment.

- Additional constraints imposed during the reconciliation process may also be semantic in nature, and therefore cannot currently be represented in a machine-readable format.
- The same business object class may appear in several documents exchanged during an interaction. Constraints imposed on this class should be applied to all documents that use this class (either directly or through a subclass). Currently, this will mean editing all of the associated schema to include the constraint. This imposes an unnecessary burden on the developers, and can potentially pose maintenance problems.
- Constraints on a business object class depend on the context – i.e. the specific deployment scenario – this class is used. We have identified that the deployment context is a function of:

  1. the PIP Document being used;
  2. the trading partner one is doing business with and whether they act as a buyer or seller (PIPs are often asymmetric);
  3. the business process being used(different business processes using a given PIP with a given trading partner may impose different constraints because of business requirements of back-end systems).

  The Next Generation PIP cannot adequately manage the application of different constraints in different circumstances. Currently, the developers would have to manually aggregate the constraints corresponding to a deployment context into refined XML Schemas and other informal documents – when XML Schema is not expressive enough. This is inefficient and could pose maintenance problems. Moreover, since these constraints are not captured in a formal and systematic way, some knowledge could be lost from a deployment to the next.
- The same constraint may apply to a certain class of partners. For instance, a back-end system could impose a constraint on a Tax class for all its European partners. Similarly, it should be possible to apply constraints on classes of PIP documents (e.g. Invoicing documents) or business processes (e.g. Electronic component purchasing).

In this paper, we present a way of overcoming these shortfalls. We present a system that can manage the relationship between a business and several different partners by formally capturing the constraints on RosettaNet deployments. It is able to automatically detect if interactions are possible with a new potential partner, and can support the reconciliation process by allowing implementation of both syntactic and semantic constraints agreed by the partners. The system is able to determine exactly which constraints to apply depending on the deployment context, and is able to automatically generate an appropriate schema to communicate with the business partner.

3 The Nile system: Introducing Semantics

We have developed the Nile system to ease the B2B integration process, and to overcome the shortfalls of RosettaNet outlined above. Our approach makes use of XML Schema to define and validate the syntactic constraints on PIP documents. It also makes use of DAML+OIL [17] to define semantic constraints. As we will show in the remainder of this paper, DAML+OIL is used to model:

- the business object class hierarchies and their attributes (or properties in Semantic Web terms);
- the semantic constraints on business objects coming from the PIP definitions (currently modelled in OCL);
- the notion of deployment context;
- the additional semantic constraints imposed by a business with respect to a deployment context.

DAML+OIL provides a single solution to model business objects and their associated constraints (both generic and context dependent). However current B2B standards do not make use of DAML+OIL but often rely on XML Schema to define the syntax of the documents being exchanged.

Nile consists of three key technology components:

- The XML Schema to DAML+OIL translation tool which converts XML Schemas into DAML+OIL class hierarchies and constraints;
- The Constraint Knowledge Base. This is a structured knowledge base, in DAML+OIL, which describes the constraints that a business places on business object classes depending on the deployment context. The Nile Constraint Editor manipulates the Constraint Knowledge Base and it allows a user:

1. to populate the deployment context ontology, i.e. to define the instances and classes representing the set of PIP documents, partners and business processes which characterise a business' RosettaNet deployments;
2. to browse the business object class definitions in the DAML+OIL knowledge base;
3. to create, modify and browse constraints on business objects in a given deployment context.

The Nile Constraint Editor provides a set of constraint templates of the forms typically encountered in RosettaNet implementations. A business will create constraints using these templates.

- The XML document Validator. This set of generic tools is used for translating documents from the 'syntactic' world of XML into the 'semantic' world of RDF [18]. Specifically, it's functionality allows:

  1. A 'best effort' translation of DAML+OIL class hierarchies and constraints back into XML Schema and Schematron [16];
  2. XML documents to be translated into RDF.

3.1 XML Schema to DAML+OIL mapping tool

In this section, we briefly present the generic mapping from XML Schema to DAML+OIL used in the Nile system.

Many XML based applications, like RosettaNet PIPs, would benefit from rich semantic modelling capabilities. XML Schema covers some simple data modeling needs [20]. We propose to lift the data model to DAML+OIL which is a more more expressive modeling language.

XML defines a transfer syntax for tree-structured documents. XML Schema definitions holds the declarations for validating XML instance documents. These declarations are syntactic constraints on what make a valid XML document. In the Semantic Web domain, RDF [18] models data in the form of directed labeled graphs and is layered on top of XML for serialisation. As pointed out in [25], this choice of a different data model makes rich semantic descriptions and inferencing out of reach for XML applications. We would like to benefit from DAML+OIL [17] modeling capabilities in B2B applications such as RosettaNet since it offers an expressive logic while keeping efficient reasoning possible [14].

The XML Schema to DAML+OIL mapping tool generates a DAML+OIL ontology from an XML Schema type hierarchy. The purpose of such a tool is to lift XML Schema to the level of an ontology. It creates a skeleton ontology which can be extended with a DAML+OIL editor (such as OilEd [1]). In the Nile system, we use this tool to populate the Constraint Knowledge Base with business object classes from the XML Schemas provided by RosettaNet. Each business object class hence has a syntactic definition – its XML Schema type – and a semantic definition – its associated DAML+OIL class. We present an overview of the mapping from XML Schema to DAML+OIL in figure 1. An exhaustive discussion on this mapping is out scope of this paper.

The following is a simple example taken from PIP3C3 [29] which is a Notification of Invoice document. From the $PIP3C3\_FinancialDocument$ XML Schema complex type, the following DAML+OIL is automatically generated[1]:

$$\begin{aligned}
&PIP3C3\_FinancialDocument \sqsubseteq \\
&\quad FinancialDocument \sqcap \\
&\quad \forall\, lineItems.PIP3C3\_LineItem \sqcap \\
&\quad \geq 1\; lineItems
\end{aligned}$$

It is now possible to express constraints of a semantic nature on business objects such as $PIP3C3\_FinancialDocument$.

3.2 The Constraint Knowledge Base

We have identified in section 2 that a constraint on a business object depends on its deployment context. A deployment context is characterised by: the particular PIP document the business object it appears in, the buyer and seller trading partners and the business process used.

3.2.1 Nile ontology

We define a simple ontology which models the deployment contexts in which a constraint can apply. We create 3 DAML+OIL classes, $Document$, $Partner$ and $Process$, and 4 properties $document$, $buyer$, $seller$ and $process$. The deployment context ontology is populated with subclasses and instances of the 3 classes mentioned above. For example, $PIP3C3$ is an instance of $Document$, $EuropeanPartner$ is a subclass of $Partner$.

$$\begin{aligned}
Context \;\doteq\; &\forall\, document.Document \sqcap \\
&\forall\, buyer.Partner \sqcap \\
&\forall\, seller.Partner \sqcap \\
&\forall\, process.Process
\end{aligned}$$

A deployment context is created by subclassing the $Context$ class and adding restrictions on one or more of

[1]For the purpose of this paper, we will use the Description Logic notation instead of the DAML+OIL syntax since it allows for more concise expressions. Namespaces will also be omitted.

1. The root `schema` element, `complexType` definitions, model `group` definitions and `attributeGroup` definitions are mapped to DAML+OIL classes.
2. Named `simpleType` definitions stay untouched; anonymous `simpleType` definitions are assigned a unique name and copied to a separate datatype file. These `simpleType` definitions are used to restrict datatype properties.
3. `complexType` elements are mapped to DAML+OIL object properties; `simpleType` elements and attributes are mapped to DAML+OIL datatype properties.
4. Type and occurrence specifiers of elements and attributes are mapped to an intersection of DAML+OIL property type (i.e. `toClass`) restrictions and cardinality restrictions.
5. `extension` and `restriction` definitions are mapped to a DAML+OIL `subClassOf` relationship.
6. Groups with a `choice` compositor are mapped to the DAML+OIL equivalent of an XOR (with `intersectionOf`, `unionOf` and `complementOf`).
7. Groups with an `all` or `sequence` compositor are mapped to a DAML+OIL `intersectionOf` collection.
8. `substitutionGroup` relationships are mapped to a DAML+OIL `subPropertyOf` relationship.
9. Names of components are always mapped to an URI composed of the schema `targetNamespace`, `#` and the component's name.

Figure 1. XML Schema to DAML+OIL Mapping

the four properties, allowing the specification of restricted contexts. For example, the $buyer$ property can be restricted to the subclass $EuropeanPartner$ of $Partner$, to allow the definition of a constraint which applies to all European Buyers.

To represent constraints depending on deployment contexts, we also create the $Constraint$ class and the $inContext$ property.

$$Constraint \quad \doteq \quad \forall\, inContext.Context$$

A constraint on the business object class BO in context Ctx is defined as follow:

$$BO \sqcap \forall\, inContext.Ctx \quad \sqsubseteq \quad Ce$$

where Ce is a constraint expression (section 3.2.2).

Given such a constraint, given BO' a business object class such that $BO' \sqsubseteq BO$ and given Ctx' a context such that $Ctx' \sqsubseteq Ctx$, the constraint Ce also applies on BO' since

$$BO' \sqcap \forall\, inContext.Ctx' \quad \sqsubseteq \quad Ce$$

is also true. For instance, a constraint on the business object $FinancialDocument$ for all European partners buyers ($\forall\, inContext.(Context \sqcap \forall\, buyer.EuropeanPartner)$) will also apply on the $PIP3C3\_FinancialDocument$ (which is a subclass of $FinancialDocument$) in the more restrictive context where the buyer is the European partner A and the business process $BP1$.

Description Logic (DL) reasoners such as Racer [11] or FacT [13] can be used to do inferencing on DAML+OIL ontologies. They can check the satisfiability of an ontology. We have successfully used Racer to check the consistency of the constraints our Knowledge Base. Having an automated way of checking this consistency is very beneficial in the Nile system since constraints could be derived from super-classes or super-contexts.

3.2.2 Constraint expressions

Constraint expressions can be arbitrarily DAML+OIL expressions that restrict the business object class (and subclasses).

As an example, the constraint that restricts the class $PIP3C3\_FinancialDocument$ in context Ctx to have at most 10 $lineItems$ elements and at least 1 $soldTo$ elements could be written:

$$PIP3C3\_FinancialDocument \sqcap \forall\, inContext.Ctx \sqsubseteq \\ \leq 10\; lineItems \sqcap \exists\; soldTo$$

We now give a more complex example constraining the class $PIP3C3\_FinancialDocument$ in context

Ctx. If any $PIP3C3\_FinancialDocument$ instance has a $lineItems$ element, this $lineItems$ element must have at least one $totalLineItemAmount$ element. This is expressed as follow:

$$PIP3C3\_FinancialDocument \sqcap \forall\, inContext.Ctx \sqsubseteq \forall\, lineItems.(\exists\, totalLineItemAmount)$$

Because the RosettaNet NextGen PIPs are designed using UML, semantic constraints on PIP document specifications are written in OCL. Our study of these specifications show that these constraints only use a subset of OCL and can all be represented in DAML+OIL. We have successfully used OilEd [1] to model these constraints in DAML+OIL. We now give an example of an OCL constraint taken from PIP 3C3 and its translation in DAML+OIL.

```
context FinancialDocument
 inv:
  if self.isLockBoxUsed='yes' then
   self.transferTo->size=1 and
   self.remitToAddress.addressLine1->size=1
    and
   self.remitToAddress.globalCountryCode->size=1
    and
   self.remitToAddress.nationalPostalCode->size=1
    and
   self.remitToAddress.regionName->size=1
  endif
```

is translated into:

$$\begin{aligned}&FinancialDocument \sqsubseteq\\ &\quad \neg\exists\, isLockBoxUsed.\text{"}Yes\text{"} \sqcup\\ &\quad (= 1\, transferTo \sqcap\\ &\quad \forall\, remitToAddress.(= 1\, addressLine1 \sqcap\\ &\qquad = 1\, globalCountryCode \sqcap\\ &\qquad = 1\, nationalPostalCode \sqcap\\ &\qquad = 1\, regionName))\end{aligned}$$

DAML+OIL is a powerful ontology language but it is also quite complex for non-expert users. One of the goal of the Nile Constraint Editor is to hide this complexity from RosettaNet implementors. An analysis of RosettaNet deployments has been carried out to determine the kinds of constraint commonly applied to documents by businesses, and 3 key classes have been identified. Here we present templates for each class, together with an example constraint of that class.

1. Cardinality constraints: because of the diversity of deployment scenarios, the RosettaNet specification may leave a lot of flexibility for the cardinality of some fields (many fields in the specifications have the $0..\infty$ cardinality). However, specific deployment contexts may impose more constrained cardinalities.

   To restrict the maximum cardinality of $lineItems$ to 10 on all $Invoice$ classes in context Ctx, the tool generates the following statement:

   $$Invoice \sqcap \forall\, inContext.Ctx \quad \sqsubseteq \quad \leq 10\, lineItems$$

2. Data format constraints: for the same reasons, the format of some data needs to be constrained. Common examples include the size of a string or the format of a date. In DAML+OIL terms, the relevant datatype property needs to have a more restricted XML Schema type. The tool generates a restricted XML Schema type and constraints the datatype property to this newly defined type.

   For instance the XML Schema simple type $ProprietaryDocumentIdentifier$ in PIP 3C3 is defined as being an `xsd:string`. XML Schema datatype restrictions [3] can be used to restrict the length of the string or its format with a regular expression.

3. Interdependency of fields: the presence of a field, or the value of a field, may imply a cardinality constraint of a data format constraint. We give an example of such a constraint: on class $FinancialDocument$, if there is a $soldTo$ element, there must also be a $soldToTax$ element.

   $$FinancialDocument \sqcap \forall\, inContext.Ctx \sqsubseteq \neg\exists soldTo \sqcup\, \exists soltToTax$$

3.3 Validation of XML documents

At design time, the Nile Constraint Editor uses the XML Schema to DAML+OIL mapping tool to create a skeleton ontology; it extends this generated ontology to the specific needs of some business activities. At runtime, when XML documents are being processed, we would like to perform validation on these instance documents based on the extended ontology.

Our approach is to have a 2-phase validation process: first the validation of syntax of incoming documents according to their XML Schemas with a validating XML parser; then the validation of the semantic constraints.

In order to validate the syntax, we generate XML Schemas that convey all the syntactic restrictions, i.e. the data format constraints, that have been added in the knowledge base. These specific XML Schemas are constrained versions of the original RosettaNet XML Schemas where:

- the cardinality of certain fields have been restricted;
- some simple types have been restricted.

We have tried two approaches to validate the semantic constraints: to generate 'best effort' schemas and to use a Description Logic reasoner.

Generating 'best effort' schemas. XML Schema is not the only schema language for XML. Other schema languages are available, and all have been designed with different assumptions and different emphasis [19]. Where the constraints expressed in DAML+OIL follow certain known patterns – as is the case for those generated by the Nile system –, we can translate those into Schematron schemas [16]. We have successfully translated the constraints showing cardinality interdependency between fields into Schematron schemas and believe we could also translate other patterns. The benefit of this approach is that it integrates well with current XML processing (since implementations are usually XSLT [7] based) and would be easily accepted by developers. Its drawbacks are that constraint patterns must be known in advance, and only selected DAML+OIL can be converted. We have not investigated the difficulty of generating Schematron schemas out of arbitrary DAML+OIL.

Using a Description Logic reasoner. DAML+OIL reasoners are meant to do inference on RDF data. Since XML and RDF have different modeling foundation, we have developed a tool that translates XML documents to RDF data so that they can be processed by a DAML+OIL reasoner. This tool outputs an RDF directed graph representation of the XML document; more precisely is uses the Post-Schema Validation Infoset (PSVI) [32] which augments the XML Infoset [9] with information such as the type of an element or its default value. Hence this tool is only capable of generating RDF models out of XML documents which validate an XML Schema. We give an overview of the mapping in figure 2 but a detailed explanation of the mapping is out of scope of this paper.

Before the generated documents can be processed by a DL reasoner, the generated RDF model must be pre-processed. Because DL reasoners make the open-world assumption [30], we have to 'close the world'. The open-world assumption means that what cannot be proven to be true is not necessarily false. For instance, if a property is not present, it is wrong to assume it will never be present. This fits very well with the nature of the Semantic Web, as statements will be added as the Web is browsed or crawled. In a B2B context however the closed-world assumption is usally made: the documents being exchanged usually contain all the information that is required. To 'close the world', for each individual and each property, we count the number of times the property appears on the individual; this property on this individual is then restricted to have its occurence number as its maximum cardinality. We also take the opportunity to assert on some properties the default values from the PSVI. The output of this process can then be submitted to Racer which can validate the instance.

1. Create an RDF resource representing the XML document.
2. Add an `rdf:type` property to the RDF resource. Its value is the URI of the DAML+OIL class corresponding to the XML Schema type of the XML element.
3. Each sub-elements and attributes are translated to RDF properties on this RDF resource.
4. If an element is a leaf node, the data is represented as an RDF literal on this property. Otherwise, the element contains attributes and/or sub-elements and it is transformed to an anonymous resource which becomes the value of the respective property. This resource is recursively transformed starting at step 2.

Figure 2. XML to RDF Mapping

4 Using Nile

The Nile tool can be used to commission a new B2B partnership, and to manage an existing one. Here we present the stages this process goes through, and show how Nile is used at each stage.

1. For a new relationship to be set up between two partners, they must first identify the appropriate PIPs and associated documents which will be transferred. Given this (assuming the PIP is next-generation), they will have access to the RosettaNet XML Schemas for the documents. The XML Schemas are loaded into the Nile Constraint Editor and automatically translated into DAML+OIL and loaded into the Knowledge Base (figure 3, step 1). For instance to set up a relationship involving PIP3C3, the PIP3C3 XML Schema needs to be loaded in the Nile Constraint Editor. If the PIP specifies additional constraints, these can also be entered in the Knowledge Base (figure 3, step 2). If a business has already used this PIP with another partner, appropriate information will already appear in their knowledge base, so they can skip this stage.

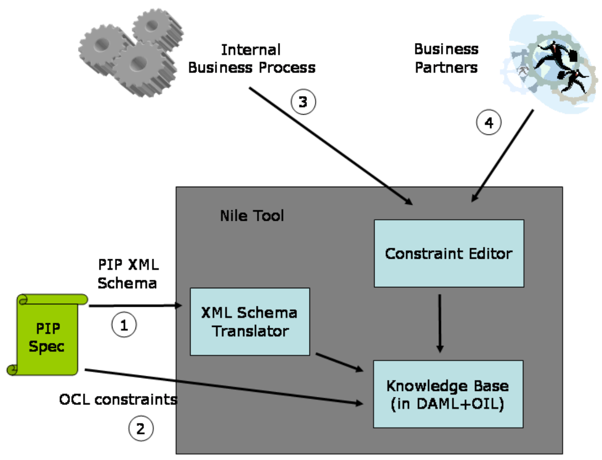

Figure 3. The Nile System

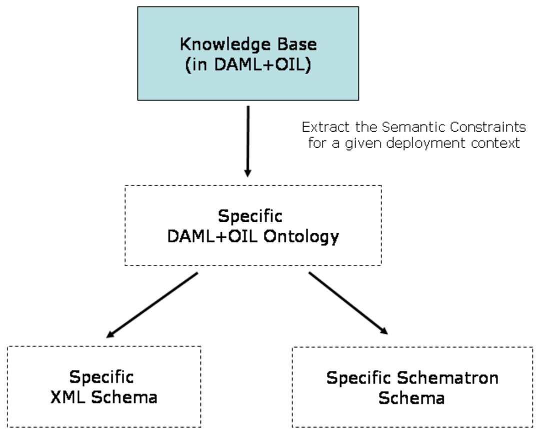

Figure 4. Deploying a PIP in a given context

2. The partners augment the set of constraints with personal constraints which are imposed by their internal business processes and the specifics of the relationship they are trying to set up (figure 3, step 3 and 4). These should represent constraints which would require business re-engineering to alter, not simply preferences. Legacy systems often impose hard constraints that cannot be altered. Often, a constraint applies to a feature of the business process which may appear in many documents and many processes. To encourage maintainability and re-usability, it is best to generate a single constraint which applies in a more general context. Previously entered constraints that are compatible with the current deployment context are automatically inherited.

3. When both partners have prepared their constraints, they can determine if their processes are potentially compatible. They do this by determining if the union of both constraint sets is satisfiable. A DL reasoner, such as Racer, is used to verify this. This can either be done by a third party, or by one or both of the partners. If the constraints are not satisfiable, it means there is a fundamental mismatch between the two business processes, and re-engineering will be required to achieve compatibility. The two partners will need to enter into off-line negotiations to determine how to handle this. When one or both partners have altered their business process, they should adjust their constraints to reflect this, and return to stage 1. If the constraints are satisfiable, the intersection of the two defines a subclass of the document definition which is acceptable to both parties. This is used as input to the next stage.

4. For a given relationship and PIP, Nile is used to generate the specific syntactic and semantic constraints for all documents which will be exchanged (figure 4). This is done by generating the subset of the Constraint Knowledge Base that is specific to the context of this relationship – i.e. the constraints that are subclass of $\forall\ inContext.Ctx1$, where $Ctx1$ is the instance deployment context specifying the PIP instance, the partners and the business process. The tool also produces a restricted version of the PIP XML Schema that includes the extra syntactic restrictions. Optionally, it produces a 'best effort' Schematron schema that captures the constraints from the subset of the Knowledge Base.

5. Developers use Contivo, together with the XML Schema, to enable generation of documents at runtime as required by the execution of a PIP.

6. Optionally, runtime validation can be carried out. Syntactic validation is carried with the generated XML Schema. For semantic validation, one of the two methods presented in section 3.3 can be used. If a Schematron Schema has been generated, it is deployed in a Schematron engine and validation can be performed directly on the XML documents. Otherwise, the XML document is translated and processed into a closed-world RDF representation as presented in 3.3. A DAML+OIL reasoner can check whether this RDF instance document is compatible with the DAML+OIL Kowledge Base.

In this way, developers can use Nile in conjunction with existing products to rapidly set up new RosettaNet relationships.

5 Related Work

In [12], it has already been pointed out that many B2B vocabularies are not interoperable because businesses use different subsets of the standards. The proposed solution of generating specific syntax (with XML Schema) out of a semantic layer (with RDF Schema [5]) capturing business requirements is similar to ours. We extend this work by adding the notion of reconciliation of processes by automatically checking the compatibility of business partners' constraints. Also we try to propose solutions for the automatic validation of the XML documents.

Several solutions have already been proposed to bridge the gap between XML and RDF to provide rich semantic descriptions to XML applications. Some solutions are application specific – like the one in [15] which uses a combination of XML Schema and RDF Schema –, while some others are more general but typically require changes to XML or RDF [21, 25, 26]. Our approach does not require changes to the standards but is focussed on XML Schema.

In RosettaNet PIPs, XML Schemas are normative and the UML diagrams document the schemas. However we think that Nile could be enhanced by also supporting UML. It could then be used for other standards such as ebXML [22] that attach more importance to UML. Also, we could benefit from existing work on reasoning on UML [2] and OCL [31] with description logics.

6 Conclusions and Future Work

The Nile tool allows developers to explicitly represent the constraints on interactions in different contexts and to re-use constraints between messages and businesses. This makes the process of setting up new relationships faster and the resulting software is more reliable and re-usable.

We have developed a suite of tools to allow the use of DAML+OIL rich semantic modeling features in XML documents and their associated XML Schemas. We plan to produce a formal mapping of XML Schema to the Web Ontology Language (OWL) [10] and will release the translation suite as an add-on to the Jena toolkit [6].

The approach we have taken is evolutionary, in that it accepts RosettaNet in its current form. However, we believe that our experiences can provide valuable input into future enhancements of RosettaNet. In particular, we believe that it should adopt OWL, successor of DAML+OIL, as the ontology language to formally specify PIP messages. The increased expressivity will avoid unnecessary ambiguity in the specification. Furthermore, it will allow new tools to be developed which support business interaction by reasoning with constraints.

7 Acknowledgments

Thanks to Matthias Ferdinand for his contributions on the XML Schema to DAML+OIL tool set. Thanks to Abdel Boulmakoul, Jeremy Carroll, Michel Morciniec and Giacomo Piccinelli.

References

[1] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Working Notes of the 2001 Int. Description Logics Workshop (DL-2001)*, pages 1–9, 2001.

[2] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML Class Diagrams using Description Logic Based Systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-44/, 2001.

[3] P. V. Biron and A. Malhotra. XML Schema Part 2: Datatypes. W3C Recommendation, May 2001.

[4] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 (second edition). W3C Recommendation, October 2000.

[5] D. Brickley and R. Guha. Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation, March 2000.

[6] B. M. Bride. Jena: Implementating the RDF Model and Syntax Specification. In *Proceedings of the Second International Workshop on the Semantic Web (SemWeb2001)*, 2001.

[7] J. Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999.

[8] Contivo. http://www.contivo.com.

[9] J. Cowan and R. Tobin. XML Information Set title. W3C Recommendation, October 2001.

[10] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Web Ontology Language (OWL) Reference Version 1.0. W3C Working Draft, November 2002.

[11] V. Haarslev and R. Möller. Description of the RACER system and its applications. In *Proceedings International Workshop on Description Logics (DL-2001)*, 2001.

[12] B. Hofreiter, C.Huemer, and W. Winiwarter. Towards syntax-independent B2B. *ERCIM News*, 51:25–26, October 2002.

[13] I. Horrocks. FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.

[14] I. Horrocks. Reasoning with expressive description logics: Theory and practice. In A. Voronkov, editor, *Proc. of the 18th Int. Conf. on Automated Deduction (CADE-18)*, number 2392 in Lecture Notes in Artificial Intelligence, pages 1–15. Springer-Verlag, 2002.

[15] J. Hunter and C. Lagoze. Combining RDF and XML Schemas to Enhance Interoperability Between Metadata Application Profiles. In *Proceedings 10th International World Wide Web Conference (WWW10)*, 2001.

[16] R. Jelliffe. *The Schematron Assertion Language 1.5*.

[17] Joint US/EU ad hoc Agent Markup Language Committee. DAML+OIL (March 2001), http://www.daml.org, 2001.

[18] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. W3C Recommendation, February 1999.

[19] D. Lee and W. W. Chu. Comparative analysis of six XML schema languages. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(3):76–87, 2000.

[20] M. Mani, D. Lee, and R. Muntz. Semantic Data Modeling using XML Schemas. In *Proceedings 20th Intl. Conf. on Conceptual Modeling (ER)*, 2001.

[21] S. Melnik. Bridging the Gap between RDF and XML, 1999. http://www-db.stanford.edu/m̃elnik/rdf/fusion.html.

[22] D. Nickull and B. Eisenberg. ebXML technical architecture specification. Technical report, UN/CEFACT, 2000.

[23] Object Management Group. *Object Constraint Language Specification*, September 2001. Published as part of [24].

[24] Object Management Group. *Unified Modeling Language Specification, Version 1.4*, September 2001.

[25] P. Patel-Schneider and J. Simon. The Yin/Yang Web: XML Syntax and RDF Semantics. In *Proceedings 11th International World Wide Web Conference (WWW2002)*, 2002.

[26] P. F. Patel-Schneider and J. Simon. Building the Semantic Web on XML. In *Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002)*, 2002.

[27] RosettaNet. http://www.rosettanet.org.

[28] RosettaNet. Next generation architecture. http://www.rosettanet.org/nextgenarchitecture.

[29] RosettaNet. *PIP 3C3: Notify of Invoice*, November 2001.

[30] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 2000.

[31] R. V. D. Straeten. Using Description Logic in Object-Oriented Software Development. In *Proceedings of DL2002 International Workshop on Description Logic*, 2002.

[32] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. W3C Recommendation, May 2001.