UDC 621.391

Semantic Web Technologies for Enterprise Application Integration

Nenad Anicic[1,2], Nenad Ivezic[1]

[1] National Institute of Standards and Technology
100 Bureau Dr., Gaithersburg, MD 20899, USA
{nanicic, nivezic}@nist.gov
[2] Faculty of Organizational Sciences,
11000 Belgrade, Serbia and Montenegro
anicic.nenad@fon.bg.ac.yu

Abstract. Large industrial interoperability projects use syntax-based Enterprise Application Integration standards, such as XML Schema, to accomplish interoperable data exchange among enterprise applications. In this paper, we describe an approach to assess the potential impact of Semantic Web technologies on these standards and on testability of integration results when using these standards. The experimental approach includes an automated translation of an XML Schema-based representation of business document content models into an OWL-based ontology. Based on this ontology, we use the Semantic Web representation and reasoning mechanisms to validate ontological constructs and constraints in support of data exchange. We demonstrate novel, model-based integration capabilities that go beyond the existing syntax-based approaches. These new capabilities are relevant when managing multiple enterprise ontologies derived from a common ontology.

1. Introduction

Success of large-scale, industry-wide enterprise integration efforts depends on the enterprise application integration (EAI) standards. Examples of such EAI standards include Open Applications Group (OAGIS) [1], RosettaNet [2], and Universal Business Languages (UBL) [3]. Currently, these standards are based on XML specifications that are syntactic formalisms [4,5,6]. Capabilities of these standards and testability of integration results based on these standards are significantly limited as a consequence of the limited reasoning capabilities supported by syntactic formalisms. This follows from the fact that syntax-based approaches to define structure of business documents do not impose a common interpretation of the data and there is no way to achieve a

repeatable and a verifiable procedure to recognize a semantic unit from a domain of interest [7].

Take, for example, the Schematron [8] rules that are typically used to encode constraints for the application content of the messages exchanged among applications. These rules, however, cannot be reasoned about and compared in a context of some integration problem. Consequently, two rules that are perfectly valid syntactically may be conflicting with each other within a certain integration context.

The advent of Semantic Web offers opportunities for more capable EAI standards to capture and manipulate semantic relationships. Semantic formalisms at the foundation of these technologies allow use of computational approaches to reason about formally expressed concepts and make inferences that are useful, yet beyond the capabilities of the syntax-based approaches. Consequently, testability of the application integration efforts may become equally more powerful. Essentially, the reasoning methods, such as satisfiability and consistency checking, may be readily used to perform various types of validations, such as whether two ontologies are compatible and whether a specific business document instance has sufficient and necessary data to belong to a specific class of documents.

In principle, the Semantic Web technologies today enable one to draw automated inferences about relationships between conceptual structures using a subset of the First Order Logic formalism called Description Logics. As an example, it is possible to express constraints on existence of an element in a document schema (e.g., 'The access rights element will appear only if the sensitivity type element appears') and to reason about possible conflicts of such a rule with other document rules (e.g., 'Either the access right or sensitivity type element, but not both, will appear'). These types of reasoning are not possible using purely syntactic approaches.

This paper describes an approach to evaluate capabilities of the Semantic Web technologies for EAI and, particularly, how it affects integration testing capabilities. The specific objectives that drive this work are (1) to develop an experimental tool enabling assessment of Semantic Web technologies for EAI and (2) to design and execute a series of experiments to effectively perform such an assessment. To accomplish these objectives, the paper posits Semantic Web-based integration architecture and an integration methodology that is enabled by such architecture.

The rest of the paper is structured as follows. Section 2 describes a prototypical problem considered for this work. Section 3 describes a current, traditional EAI standards architecture and, then, proposes a Semantic Web-based architecture. Section 4 gives terminology to describe our methodology. Section 5 describes details of the developed integration methodology. Section 6 discusses some initial findings. Section 7 includes

a description of the related work. Finally, Section 8 provides concluding remarks.

2. A Prototypical Problem

The scope of our effort is partially defined by the type of problems identified in this section. First, however, we define a few terms.

By integration of enterprise applications, we mean exchange of *business document instances* (or, simply, *business documents*) between two enterprise applications that are based on two different *business document content models* (or, equivalently, *interface models*) so that interoperable data exchange is achieved. Obviously, business document instances conform to business document content models. *Interoperable data exchange* is clearly the key objective of an integration effort and may be thought of as such an exchange of data that preserves intended meaning of the data.

The prototypical problem discussed here may be readily encountered in the traditional standards usage for enterprise application integration, as described next and shown in Fig. 1

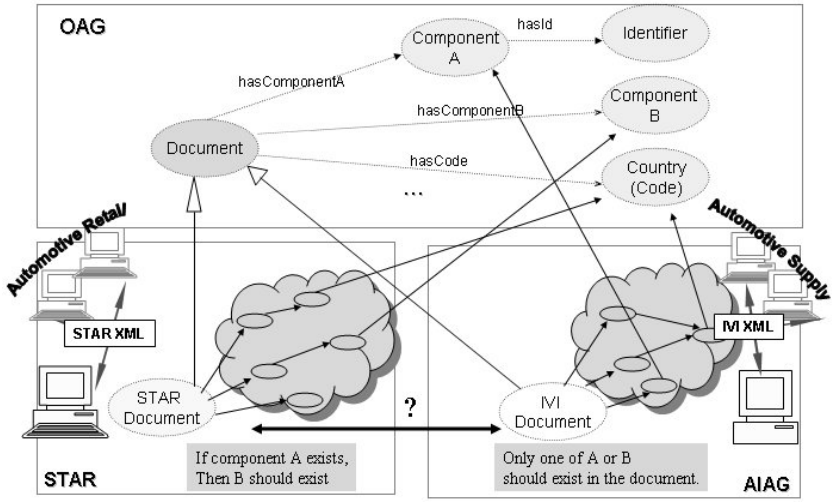

Fig. 1. A Prototypical Problem: Interface Standards Compatibility Checking

Two independent, but related, industry consortia develop respective enterprise application integration standards (or, equivalently, business document content models or interface standards). First, an automotive retail consortium, call it STAR for Standards in Automotive Retail, develops XML Schema-based standards to enable business documents to be transacted by automotive manufacturers and their retail houses[9].

Second, an automotive supply chain consortium, named AIAG for Automotive Industry Action Group, develops XML Schema-based standards to enable business documents exchange by automotive manufacturers and their suppliers [10].

A STAR application adopts and implements the proposed STAR XML-based interface model. However, an additional requirement is posed for the STAR applications: to be able to exchange the automotive parts ordering data with the AIAG applications that adopted the AIAG interface model.

Both STAR and AIAG consortia base their interface models on the same 'horizontal' document standard – The OAGIS Business Object Documents (BODs). BODs are specifications of general XML Schema components and general aggregations that make up business document content models from these components. Each consortium independently uses the OAGIS BODs to customize their own document content models and define usage rules for the components (e.g., mandatory and conditional components).

Presently, the usage rules for the business document content models are captured outside the XML Schema using syntactic constructs (e.g., Schematron rules). A significant manual task is required to identify and reconcile differences among constraints and rules of two or more standards. We seek an approach to enable automated checking of compatibility among rules and constraints that are independently developed within the two or more standards groups with a common terminology at their bases. Once such automated checking of compatibility is in place, more capable application integration and testability of integration results are expected.

3. A Semantic Web-based Architecture for EAI Standards

In this section, we compare a traditional and a novel architecture to integrate enterprise applications. We continue to use OAG, STAR, and AIAG terms to indicate a general, a source, and a target standard specification, respectively.

3.1. Traditional EAI Standards Architecture

The left portion of Fig. 2 shows a traditional EAI standards architecture based on a pure XML Schema-based integration approach. The following steps are required to translate data from a previously developed STAR XML Schema interface model to an AIAG XML Schema interface model (and vice versa) and to verify the business document translation:

(1) Identify and resolve manually any semantic and syntactic differences for implementations of the STAR and AIAG XML Schema interface models.
(2) Create two XSLT stylesheet transformations from the source to the target XML Schema interface model and vice versa.
(3) Apply translation to a business document conformant to the source XML Schema interface model to obtain a business document conformant to the target XML Schema interface model (based on the XSLT stylesheet transformations).
(4) Validate the translation:
    a. Validate translated business documents with respect to the target XML Schema interface model (using syntactic approaches such as Schematron rules).
    b. Validate translation using equivalence test. The equivalence test is between the initial source business document and the final source business document that is obtained through a sequence of two (forward and reverse) translations compatible with transformations in step (2).

The validation of translation using an equivalence test (step 4b above) is not straightforward. Specifically, applying the two translations in sequence (forward and reverse) using different mechanisms and comparing the final source business document to the initial source business document is problematic. Namely, some issues arise during the validation stage that require capability beyond a simple, syntax-based equivalence test. For example, despite a syntactically different element order (in the sense of XML Schema), elements may be semantically equivalent, if that order is not significant. In a different example, an equivalent time period can be specified either by a start date with (1) an end date or (2) a duration of time period. This may pose a difficulty to simple syntax-based equivalence tests.

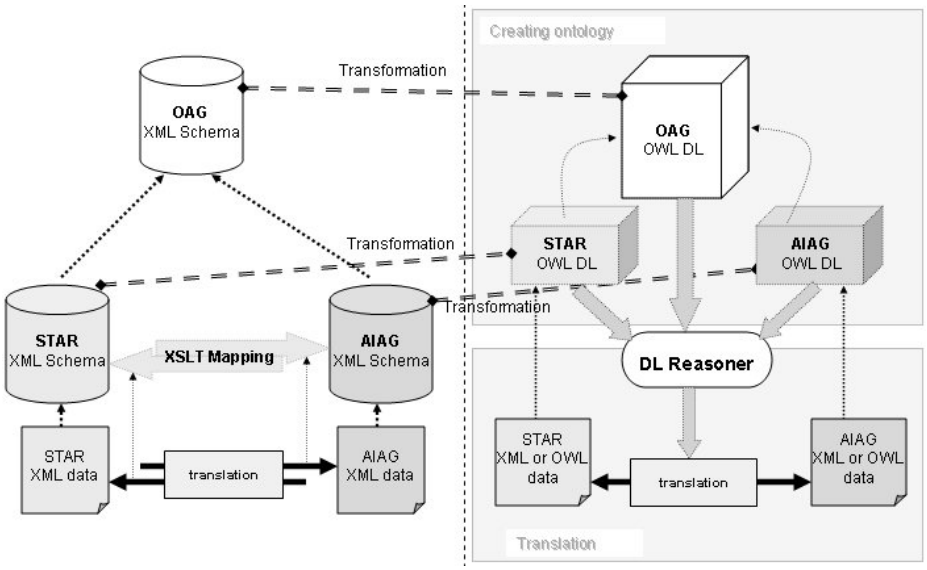

Fig. 2. Traditional and Semantic Web-based EAI Standards Architectures

3.2. A Semantic Web-based EAI Standards Architecture

The right portion of Fig. 2 shows the proposed Semantic Web-based EAI standards architecture. In this approach, OWL-DL language is employed to formally define business document content models [11]. The language is based on a subset of the First Order Logic formalism called Description Logics. This, in turn, enables us to readily use automated reasoning methods provided by DL reasoners (e.g., Racer [12]). These reasoning methods are fundamental enablers of automated transformations (i.e., mapping functions between OWL-DL interface models). The basic assumption is that the interface models are independently developed but have a common terminology as their bases.

As in the previous, traditional approach, we assume previously independently developed STAR and AIAG XML Schema interface models. At this point, we assume that the OAG, STAR, and AIAG OWL-DL ontologies have been created – a step that will be discussed in detail later.

The following steps are envisioned to translate and verify the translation in the proposed architecture:

(1) Perform model-based equivalence analysis of STAR and AIAG schemas. The following steps are involved.
    a. Create a merged ontology from independently developed STAR and AIAG ontologies and check for unsatisfiability

   b. Identify similarity between two schemas based on the comparison of their semantic models and using an automated inference tool.

(2) Apply semantic translation using the merged ontology and an OWL-DL reasoner.
   a. Translate the source (STAR) XML instance to the source (STAR) OWL representation
   b. Check for consistency and sufficiency with respect to the merged (source-STAR + target-AIAG) ontology
   c. Classify the source OWL individual into the target ontology (AIAG) and perform validation and serialization

We maintain reference to two distinct parts of this proposed architecture: the ontology creation part and the translation part.

4. Semantic Web-based Integration Terminology

Before we describe details of the developed integration methodology, we introduce a formalism and terminology to describe the methodology. In this paper, we use the word "concept" (interpreted as a set of individuals) to refer to the expressions that define a class in the OWL-DL language and a terminology to denote a hierarchical structure that provides a representation of the domain of interest. The key features of Description Logics reside in constructs for establishing relationships between concepts. The meaning of concepts is specified with a logical semantics. An important distinction in using logical semantics to describe a concept meaning is between the concept description (i.e., class with necessary conditions only) and concept definition (i.e., class with both necessary and sufficient conditions).

In the following, we use Description Logics formalism to make statements how concepts and roles are related to each other and also to describe testing algorithms and results. The use of the formalism allows automated reasoning techniques to be used to check the consistency of classes and ontologies, and to check entailment relationship. In fact, OWL DL could be easily mapped to $\mathcal{SHOIN}(D_n)$ an expressive Description Logic [13], with an ontology equivalent to a Description Logics knowledge base. An OWL essential feature is that it uses a DL style model theory to formalize the meaning of the language. In order to define formal semantics of OWL DL as Description Logics model, we consider the semantics of concepts in terms of an interpretation $I = (\Delta^I, \cdot^I)$ that consists of a domain of interpretation (nonempty set) Δ^I and an interpretation function $\cdot^I$, which maps every atomic concept C to a subset of Δ^I ($C^I \subseteq \Delta^I$), every atomic role R to a binary relation $R^I \subseteq \Delta^I \times \Delta^I$, and every named individual o to an element of Δ^I ($o^I \in \Delta^I$). The interpretation function can

be extended from concept names to complex concept descriptions in an obvious way.

A DL knowledge base $\sum(T, A)$ consists of a set of terminological axioms T (often called a TBox) and a set of assertions about individuals A (often called an ABox). To construct a knowledge base using concept languages we permit concept and role expressions to be used in assertions on individual, C(a) and R(a,b) where C is a concept of T, R is a role of T and a,b are individuals in A. If $I = (\Delta^I, \cdot^I)$ is an interpretation , C(a) is satisfied by I if $a^I \in C^I$, and R(a,b) is satisfied by I if $(a^I, b^I) \in R^I$.

There are two important tasks that are fundamental to our methodology and that are enabled by the formally defined semantics of OWL DL:

- ***Calculating a concept satisfiability*** means determining whether the concept description is not contradictory with the rest of an ontology. A concept C is satisfiable if it has a model for a concept C (i.e., C^I) that is nonempty; the concept is unsatisfiable otherwise.
- ***Checking consistency of an individual*** means determining whether the individual is an instance of a concept. Let $\sum$ be a knowledge base, then an individual $a \in A$ is an instance of concept C if and only if $\sum \models C(a)$ (i.e., C satisfies all constraints specified for concept description).

To accomplish the above two fundamental tasks, we use two basic functions of an OWL-DL reasoner:

- ***Subsumption computation*** determines whether a concept description is more general than another one. We say that C is subsumed by D ($C \sqsubseteq D$) if $C^I \subseteq D^I$ for every interpretation I.
- ***Individual classification*** determines the most specific concept for the particular individual. An individual a is recognized to be an instance of concept C if and only if $a \in C^I$ for all interpretations I. To check individual consistency we follow the usual logical paradigm where two individuals with different names are indeed different individuals. This characteristic called Unique Named Assumption (UNA), is not characteristic of OWL (that requires explicit statement that two individuals are different or equal), but is very important when we need to perform individual checking. The interpretation function $\cdot^I$ is extended in such way that for every individual $a,b \in A$, $a \neq b$ if $a^I \neq b^I$.

One of the most important inference services of DL systems is computing the subsumption hierarchy of a given finite set of concept descriptions. There are two main approaches to calculate subsumption in DL. The first approach, called structural subsumption algorithm, transforms concept descriptions into a normal form, and then compares the syntactic structure of the normalized concept descriptions [15]. This algorithm cannot handle DL with disjunction and full negation and will not be considered here. The second approach called tableau-based algorithm has been proposed in [17]. The algorithm, instead of directly

testing subsumption of concept descriptions C ⊑ D, reduces to checking unsatisfiability of axiom C ⊓ ¬ D. If the algorithm can find a finite model, then the subsumption relationship does not hold. If the algorithm fails, then the subsumption relationship holds. In this case, the algorithm looks for a 'clash' among constraints, which would preclude a model from existing. A concept C is unsatisfiable if it is impossible to create an individual that is an instance of C.

The individual classification helps us to identify other concepts that the particular individual belongs to. Let us define a set of assertions A_1 that describe an individual a, an instance of the C concept (e.g., C class is defined using two mandatory properties: r1 and r2). For example, the individual *a* has only a property filler r1 (e.g., A_1 ={ C(a), r1(a, value1)}). Because a DL reasoner makes the open world assumption (OWA), if a mandatory property is not present, the reasoner cannot conclude that it is false (as it is wrong to assume it will never be present). For that reason, the reasoner can conclude only contradictory but not insufficient information (i.e., missing properties). In a B2B context, a document being exchanged contains all required information and in order to compute that an instance has all mandatory properties it is necessary to validate instance with "local closed world assumption" (CWA)..

To check whether *a* is a valid instance of a concept C it should be sufficient to check whether most-specific concept of *a* is subsumed by C, turning instance checking into subsumption: Msc_a ⊑ C. To define most-specific concept Msc_a, we need to include close operator which takes an individual and 'closes' roles on the individual, by first counting the known fillers for the roles and than asserting number restriction on the most-specific concept. For example, the most-specific concept for the individual *a* defined above is: Msc_a ≡ C ⊓ (≤1 r1) ⊓ (≤0 r2).

5. Semantic Web-based Integration Methodology: Details

In this section, we describe in detail the proposed Web-based integration methodology. Fig. 3 and 4 illustrate the methodology using a scenario-based view of the semantic integration architecture. Fig. 3 includes a group of steps, which we call 'ontology creation' to define and test possibilities for interoperable data exchange among different XML Schemas (e.g., STAR and AIAG schemas). The ontology creation occurs during design time.

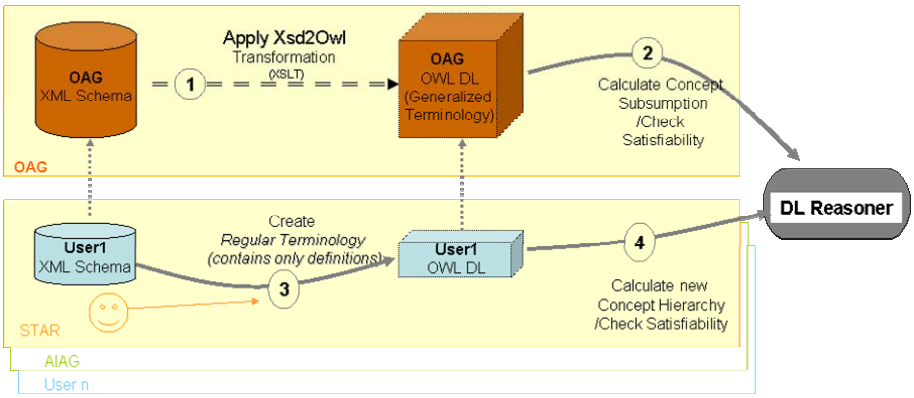

Fig. 3. Ontology Creation: Design Time View of The Semantic Integration Method

The following steps define design phase of ontology building. The OAG consortium has a role to create the satisfiable generalized terminology which can be used by independent users. For that purpose, the first two steps have to be done before a user adds any business context information to describe the user-specific business documents in a form of a satisfiable regular terminology.

(1) **Apply Xsd2Owl Transformation.** Apply an automated transformation to the OAG XML Schema representation to obtain an OAG OWL-based generalized ontology.

(2) **Calculate concept subsumption and check satisfiability of the new OAG ontology.** The outcome of this step is a new subsumption hierarchy for the OAG generalized ontology and an indication from the reasoner that the new ontology is either satisfiable (i.e., not contradictory) or not.

(3) **Create an OAG regular terminology (that requires human designer input).** The original STAR and AIAG Schemas include free text description of the additional document constraints that need to be 'layered on top' of the OAG generalized terminology. In this step, for each of the schemas, these constraints are used to specify concept definitions (based on the original concept descriptions). The outcome of this step is a regular terminology.

(4) **Check satisfiability of each individual regular ontology.** Similar to Step 2, the outcome of this step is an indication from the reasoner whether each individual ontology is satisfiable. In addition, we may choose here (during design time) to merge the resulting ontologies (same as run time Step 3) and check whether the merged ontologies are satisfiable (same as run time Step 4). This is a necessary condition for an individual translation from one to the other ontology.

Fig. 4 includes a group of steps, which we call 'data translation', that helps us to reason about concrete XML data based on the XML Schema and the possibility to transform the data from one format to another and actually achieve interoperable data exchange using the specific data. The following steps define the data translation group of steps at runtime.

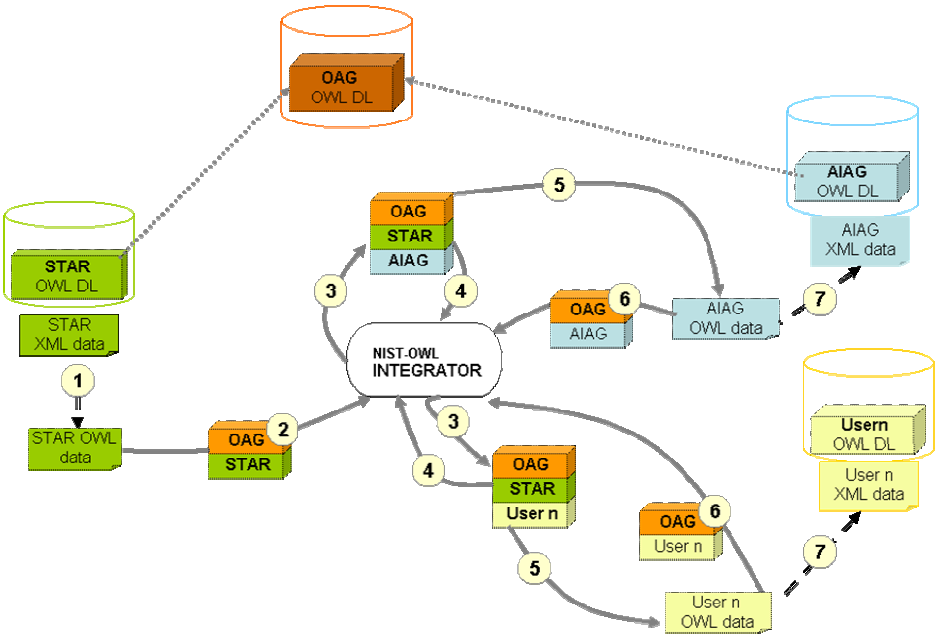

Fig. 4. Data Translation: Run Time View of The Semantic Integration Method

(1) **Apply automated transformation from source XML data to OWL data.** This transformation is dependent on the transformation defined in design phase step 1. The outcome of this step is transformed source (e.g. STAR) OWL data that correspond to the initial XML data.

(2) **Validate source data.** This step includes consistency checking under both Open World Assumption (OWA) and Closed World Assumption (CWA). Reasoning about individuals in OWL-DL assumes 'Open World'. The outcome of this step, if successful, is an indication from the reasoner that the source OWL data are consistent with respect to the source ontology. An individual is valid only if it is consistent (i.e., belongs to a specific concept) in both OWA reasoning and CWA reasoning.

(3) **Create a satisfiable merged ontology**. In order to translate from STAR to AIAG OWL data, we need to perform this step. (This and the following steps can performed by any other target system 'User n' similar.) The outcome of this step is the new merged ontology and the new concept hierarchy. (As explained in design phase Step 4, this and

the satisfiability check from the following step may be either run here, at run time, or at design time.).

(4) **Check satisfiability of the merged ontology and consistency of the STAR data with the new merged ontology.** The successful outcome of this step is an indication from the reasoner that the merged ontology is satisfiable and, similarly, that the STAR OWL source data are consistent with respect to the merged ontology.

(5) **Compute classification of the STAR OWL data in the AIAG ontology**. The successful outcome of this step is an assignment of the STAR OWL data to the specific AIAG class(es). At this point we have a result that the specific STAR XML data (instance) may be successfully translated into target AIAG XML data. This, however, doesn't mean that all STAR data may be successfully translated to AIAG, but only that the specific data may be translated.

(6) **Validate target OWL data**. The outcome of this step, if successful, is an indication from the reasoner that the target OWL data are consistent with respect to the target ontology.

(7) **Apply serialization of OWL data into XML data**. The outcome of this step is an target XML instance (e.g., AIAG) that preserves semantics defined in the original STAR OWL data.

5.1. Apply Xsd2Owl Transformation

An automated transformation was devised for the OAG XML Schema representation to obtain an OAG OWL-based ontology. This is a generalized ontology that contains concept descriptions only (i.e., necessary conditions) and no definitions (i.e., sufficient and necessary conditions). The automated transformation was possible because we took into account decisions for the OAG components and document design. Fig. 5 gives some of these rules while a detailed account of the transformation is out of scope of this paper and is a subject of a future publication.

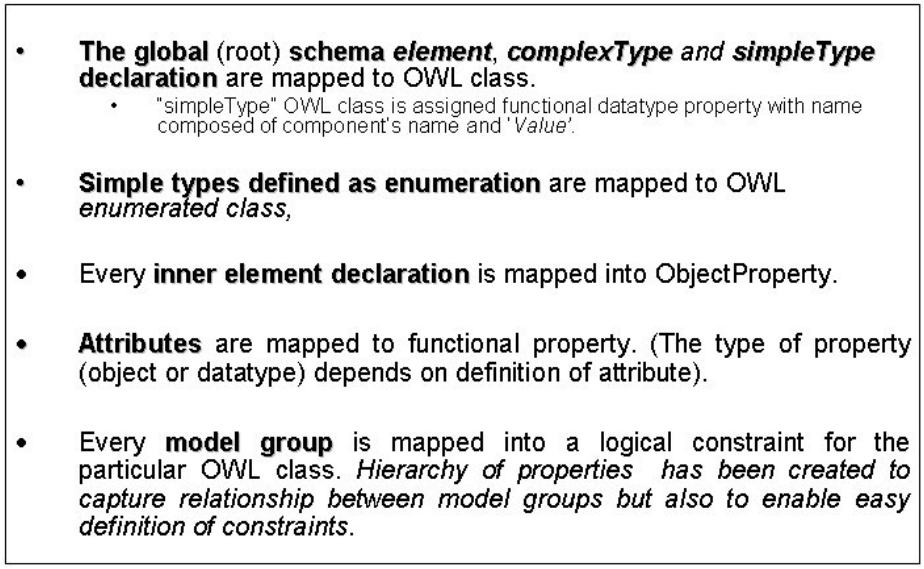

Fig. 5. Example Transformation Rules from OAG XML Schema into OAG OWL-DL Generalized Ontology

An application of the transformation rules is illustrated in the following. Fig. 6 shows a rendering of the simplified XML Schema for the OAG aggregate component AddressBase. The AddressBase component has a relatively complex structure and it describes all possible elements that an OAG Address instance (that uses AddressBase) may have. For that reason, all the elements are optional in the complexType definition of the AddressBase.

Within the AddressBase schema definition we can see choice between 'unstructured' AddressLine and the 'structured Line' that consists of parts such as StreetName, BuldingNumber, Unit, and Floor. We capture this constraint in the resulting OWL description by using a hierarchy of properties. Every property has a strictly defined range when it is used as a property of this class. The ranges for those properties are extensions of other concept (class) descriptions (e.g. IdentifierType and TextType classes).

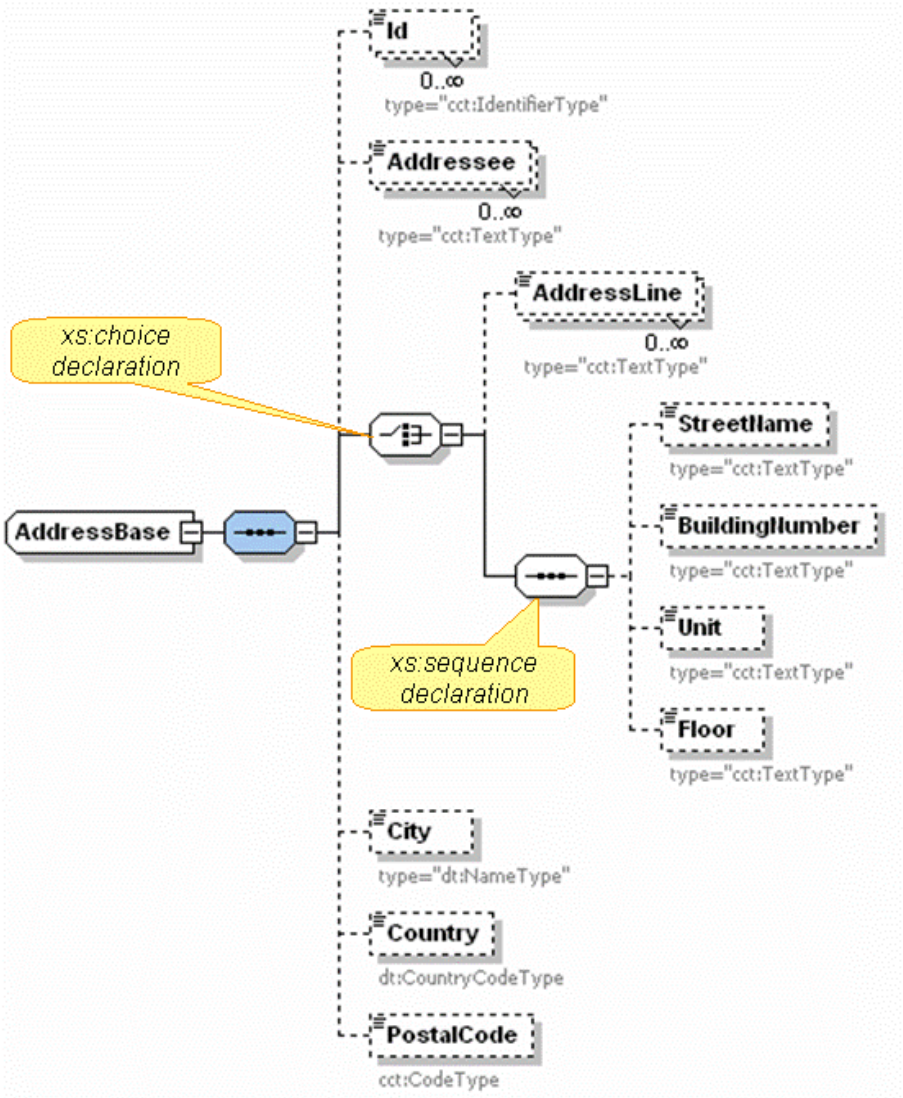

Fig. 6. A Rendering of the simplified XML Schema for the OAG AddressBase component

The generalized terminology $\mathcal{T}_1$ contains basic concepts such as IdentifierType, TextType, NameCode, CountryCodeType, CodeType,…etc, and the following complex axiom description:

$$\texttt{AddressBase} \sqsubseteq (\forall\ \texttt{hasId.IdentifierType}) \sqcap (\forall\ \texttt{hasAdressee.TextType}) \sqcap (\forall\ \texttt{hasCity.NameType}) \sqcap (\leq 1\ \texttt{hasCity})$$

```
⊓ (∀ hasCountry.CountryCodeType) ⊓ (≤1 hasCountry)
⊓ (∀ hasPostalCode.CodeType) ⊓ (≤1 hasPostalCode)
⊓ (∀ hasAddressLine.TextType)
⊓ (∀ hasStreetName.TextType)
⊓ (∀ hasBuldingNumber.TextType)
⊓ (∀ hasUnit.TextType) ⊓ (∀ hasFloor.TextType)
⊓ (¬(≥1 hasAddressLine) ⊔  ¬(≥1 sequence63736952))
⊓ (¬ (≥1 sequence63736952)
    ⊔ ((≤1 hasStreetName) ⊓ (≤1 hasBulidingNumber)
                ⊓ (≤1 hasUnit) ⊓ (≤1 hasFloor)))
```

All roles are defined as atomic. Furthermore, to represent relationships between roles we can also use inclusion. This set of inclusion role axioms defines a role hierarchy:

```
hasAddressLine ⊑  choice49723144
sequence63736952 ⊑ choice49723144
hasStreetName ⊑ sequence63736952
hasBulidingNumber ⊑ sequence63736952
hasUnit ⊑ sequence63736952
hasFloor ⊑ sequence63736952
```

where `sequence63736952` and `choice49723144` are computer generated identifiers for the role names.

5.2. Calculate Concept Subsumption and Check Satisfiability

When dealing with large ontology transformations, a designer may specify concept descriptions that may turn out to be contradictory. A DL reasoner may calculate concept subsumption and check whether any concept description is contradictory in the resulting ontology. An example situation that results in an unsatisfiable concept (using our transformation approach) is when a complexType definition is specified as a restriction of an existing type with different cardinality constraints (e.g., an element that is mandatory in the super-type definition is prohibited in the new definition).

5.3. Create Regular Terminologies

Once we have a satisfiable generalized terminology, every individual application integrator (i.e., a human responsible for defining a business document content model and an application integration) can

independently use the terminology to specify additional constraints and to provide definition for concepts in a particular context of the intended application.

When an integrator describes a component or builds a new one, likelihood of automated interoperable data exchange will be increased if new axioms (introduced by the integrator) reference the generalized terminology. The new axioms are defined as extensions where all new concepts, defined as definitions, agree with the atomic concept and roles defined in the generalized terminology. That new terminology we call regular terminology. Terminological axioms to represent defined concepts are given in form called equality (≡).

For example, with such axioms, we associate the left-hand side concept name Address to the description on right-hand side AddressBase with two cardinality constraints:

Address ≡ AddressBase ⊓ (≥1 hasCity) ⊓ (≥1 hasCountry)

The new concept Address is introduced using the OAG AddressBase description. Address is defined as AddressBase with mandatory properties hasCity and hasCountry. When an integrator customizes component for a particular context (i.e., a BOD document), he or she needs to specify required fields and business rules for the document in that particular context.

5.4. Check Satisfiability of the Regular Terminologies

For a created regular terminology, a reasoner will calculate a new subsumption hierarchy. (All OAG concept descriptions (axioms) are imported into the new ontology). Within the new hierarchy, every concept contains both inherited and its own axioms. These axioms are either part of definition or description and need to be non-contradictory to each other. If all concepts are satisfiable, than this regular terminology (that contains definitions) can be used for application integration.

Every time when we make changes in an ontology, we need to check for the ontology satisfiability. Suppose that a logical constraint is specified for the OAG AddressBase component to state an exclusive option between an unstructured (free) text address line and a structured line (that contains hasStreetName, hasAddressLine, hasCity, hasCountry, and other elements of address). If the integrator defines a new address concept with mandatory properties hasStreetName (that is a part of 'structured' line defined via sequence63736952 super property) and hasAddressLine using the OAG AddressBase defined above, a reasoner will find that the concept is unsatisfiable.

Testing Integration Capabilities

Once we determine satisfiability of two independently defined regular terminologies, we may proceed to determine whether two interface models based on those ontologies can facilitate interoperable data exchange.

The first step is to create a merged ontology from the two regular terminologies. The merged ontology contains concept axioms from both ontologies. As both ontologies use the same generalized terminology, a new subsumption hierarchy will be calculated and new relationships may emerge among concepts. A reasoner is utilized to check satisfiability of each concept in the merged ontology. If there are no contradictory concepts, then we can say that it is possible that two interface models may support interoperable data exchange. Fig. 7 shows this testing step.

A reasoner can calculate relationships such as *subClassOf* or *equivalent*. When the subsumption or equivalency relationship cannot be calculated (i.e., when *subClassOf* or *equivalent* relationships do not hold for two concepts), an individual may still be classified to belong to either one or both of the concepts depending only on the particular individual assertion.

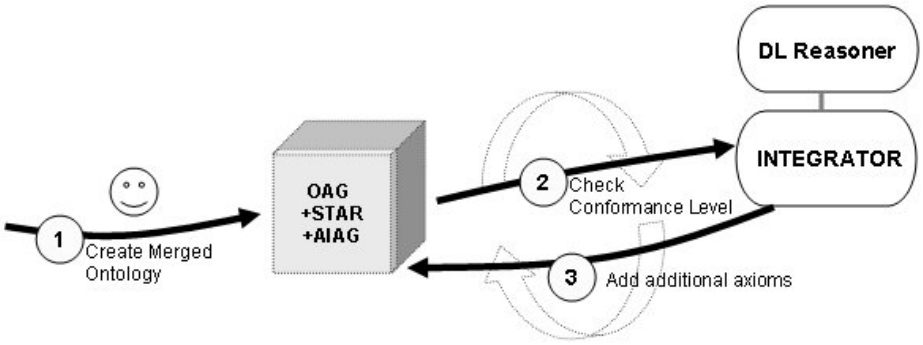

Fig. 7. Testing for Necessary Integration Conditions

The result of this satisfiability checking can be that business document content models (i.e., interface models) are either compatible (i.e., allowing bidirectional interoperable data exchange), incompatible, unidirectional, or unknown (i.e. the reasoner does not have enough information to make any conclusion and reasoning should include individuals).

If the result is unknown, a designer can provide new axioms such as conditional equivalence relationships among concepts, as indicated in step 3 in Fig. 7. New axioms might change subsumption hierarchy, produce new relationships, and may increase compatibility between two ontologies.

5.5. Transforming Source Data into OWL Individuals

In this step, we transform XMLSchema instances into OWL-DL individuals in order to conform with the OWL model-based assumptions used in ontological reasoning (i.e., satisfiability checking). We have developed a tool that enables this translation. The translation rules include the following:

- For every element (including root element) we create an OWL individual with a corresponding type.
- Parent-child relationships are translated to class-property relationships: every child element is a value of the respective property of parent class.
- The text content (the data) of element/attribute is mapped into datatype property with an RDF literal as a value for that property.

An individual that is created during this transformation gets a unique ID (URI) generated by the transformation tool. The ID of an individual is important for classification but it is not significant, which means that for the same message we can have generated different ids. Two individuals can be content equivalent if they have identical content (property values). According to the previously defined rules, an AddressBase XML Schema instance is transformed, as shown in Fig. 8.

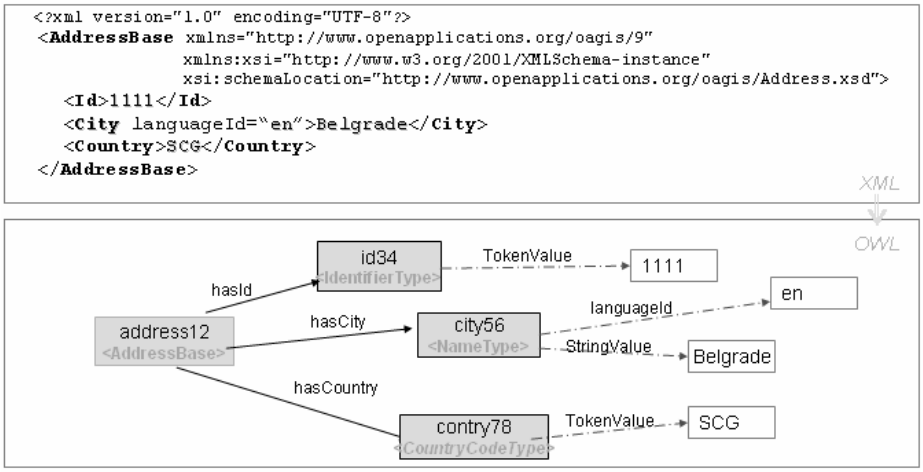

Fig. 8. An Example Transformation of XML Data into OWL Individuals

Conjunction of things asserted about an individual forms the descriptions of the individual. DL allows the user to specify that an individual is an instance of a primitive concept. For example, address12 is asserted to be an instance of AddressBase and contains roles hasId, hasCity, hasCountry filled by id34, city56, country78 respectively.

5.6. Validating Source Data

Validation is a necessary step not only to check transformation result with respect to the concept definition but also to check other semantic constraints which are defined in the corresponding ontology. Validate the data means to check whether data are consistent in OWA and valid in CWA reasoning.

If we have an individual address1 with explicit assertion that it is an instance of the star:Address with two hasCity role fillers, then using UNA, a particular individual will be calculated to have property hasCity=2, which violates the constraint for concept description ($\geq$ 1 hasCity) and, consequently, results in an inconsistent individual.

Consider the following example: star:Address(address2). We have an individual with an explicit assertion that it is an instance of the star:Address class and without any roles. Based on the instance checking in OWA, one can conclude that this individual is consistent. However, when an individual is not complete, as in this case, we can still recognize concept membership. If we know that an individual address2 is star:Address, then adding more information to the model cannot cause it to become false.

To check whether *address2* is an valid instance of a concept star:Address it should be sufficient to check whether most-specific concept of *address* is subsumed by star:Address, turning instance checking into subsumption. As this subsumption does not hold we can conclude that individual address2 is not a valid instance of star:Address.

5.7. Create Merged Ontology

In order to translate XML data from one format to another, we need to create a merged ontology. The merged ontology contains all concept axioms from relevant ontology sources (e.g. OAG, STAR and AIAG). The Semantic Web is a universally accessible platform that enables those ontologies to be shared and processed by the integration tool. Because new independently defined ontologies are based on the same generalized OAG terminology, a reasoner may combine axioms and calculate a new concept subsumption hierarchy. In the merged ontology one concept might be dependent on some concepts in the other ontology namespace. The merged semantics provides support for inferences over the source data that may yield unexpected results (such as those we discussed in the previous section). If integration capability between those specific ontologies is done at design time, as described in section 5.4, and enriched with new mapping axioms, then this entire additional axiom set will be included in the merged ontology.

5.8. Check Satisfiability and Consistency

Because the integration tool is a complete reasoner that includes consistency checkers, all axioms of the merged ontology must be loaded. The tool has to check satisifiability for every concept of the merged ontology (as described in section 5.4). An additional checking is the individual consistency checking for source individuals with respect to the merged ontology. An individual that belongs to the source concept and which satisfies all constraints in the source definition has to satisfy all constraints defined for the equivalent concept definitions in the target ontology. (For details, see Section 5.6)

5.9. Compute Target Data

In order to compute target data, we use the merged ontology to calculate a new concept subsumption hierarchy, as described in the previous two steps. In addition, we checked consistency of the source individuals with respect to the merged ontology. If the outcome of these steps included satisfiable concepts and consistent individuals, then we can use the individual classification capability of a DL reasoner to compute target data (i.e., individuals). The individual classification allows us to find what the most-specific concept is for every individual in the target ontology. The individual classification helps us to identify other concepts that the particular individual belongs to.

By using the merged ontology T' (that combines axioms from source and target terminology), we check satisfiability between the auxiliary most-specific concepts and other concept in the merged ontology. For terminology T', new axioms might be calculated (e.g., equivalence). The equivalence between two concepts may force an individual to be checked for consistency with respect to both concepts (i.e., equivalence between two concepts means that the two concepts share exactly the same set of individuals.)

5.10. Validating Target Data

As we saw in the discussion of validating a source (i.e., STAR OWL) data, it is necessary to have not only OWA consistency but also to check that the same individual is a valid instance of the target concept in the CWA reasoning. The individual consistency checking in OWA is already done with respect to the merged ontology. The OWL individuals classified to the AIAG concept hierarchy have to be checked for sufficiency with respective to target (AIAG) concepts. If the individual is inconsistent in CWA with respect to the target ontology, then translation is not possible

(e.g. the individual does not have all the required properties or violates some of the business rule constraints). If successful, however, the specific XML source data (i.e., STAR_Address from step 5), can be said to be translatable into a target OWL data and allowing interoperable data exchange.

As we have seen above, new individuals can be calculated from source data (described in 5.9.) in a way that all new individuals are consistent. However, having new individuals consistent is still not sufficient for interoperability in B2B context. This is the case when a consistent individual in OWA sense might not have all mandatory elements, as required by the necessary conditions in the subsumption between the auxiliary most-specific concept for target ontology and the intended concept.

5.11. Serializing Target Data

The serialization into OWL format is straightforward. A new file will contain a set of individuals with types from target (AIAG) ontology.

For serialization into XML format we use concept and property hierarchy. If we use default XSD serialization from our OWL ontology, then the serialization is also provided. If we have a customized mapping to specific XMLSchema syntax (e.g., a sequence of elements defined in a separate file), then that serialization is dependent on the mapping rules. The serialization algorithm will be discussed in a future publication.

6. Initial Findings

6.1. Individual Equivalence Test

One of the important tests when dealing with enterprise application integrations in a B2B setting is to check for content equivalence between two business documents. As mentioned before, during XML to OWL transformation, every new OWL individual is assigned a new URI identifier. That identifier is only necessary for individual classification and its actual value is not significant. That means that the same XML data business document instance may be transformed to individuals with different URI identifiers but same content. For datatypes 'semantically equal' means that the lexical representation of the literals maps to the same value. For individuals it means that they either have the same URI reference or are defined as being the same individual.

Two individuals are '*semantically equal*' if their respective auxiliary most-specific concepts are equivalent. Let us consider an example where we have ABox *A* with assertion about two individuals *a* and *b* as following. The individuals contain oag:hasCity with different fillers: 'city56' and 'city78' respectively. If it is given (or calculated) that two individuals are equal (e.g. city56=city78), then applying subsumption checking algorithm we can conclude that the corresponding most-specific concepts Msc_a and Msc_b are equivalent, and infer that equality between two individuals, a = b, holds.

6.2. Concept equivalence with inconsistent business document instances

In Section 5.4, we investigated whether two ontologies can facilitate interoperable data exchange and we used reasoning capabilities to perform satisfiability check between the two ontologies. We determined that if there are no contradictory concepts in the merged ontology, then we can say that it is possible that two interface models (i.e., ontologies) may support interoperable data exchange. However, that is only a necessary condition to accomplish interoperable data exchange. The following is a description of a translation problem when the necessary condition is satisfied but interoperable data exchange may not be accomplished because some individuals may violate business constraints defined for that concept.

For example, presence of a mandatory property (i.e., a necessary condition) within the target concept, may give rise to an inconsistent source individual if the source concept specifies that property as optional. It is important to keep in mind that for calculating subsumption and equivalence among concepts we use only axioms there are part of definitions – constraints, however, are not always a part of definition, they might be part of concept description. As indicated above, the mandatory property was not part of definition but only defined as a necessary condition. In a general case, any logical constraint that is not a part of either target or source concept definition but only their necessary conditions may cause a similar inconsistency and prevent interoperable data exchange.

7. Related Work

Early work in development of Semantic Web technologies pointed at the fact that semantic interoperability requires standards not only for the syntactic form of documents, but also for the semantic content [7].

A previous effort investigated use of Semantic Web technologies (e.g., DAML+OIL) in support of semantic constraints definitions and management for RosettaNet, a B2B integration standard [18]. Here, an approach for mapping from XML Schema to DAML+OIL was outlined. This approach uses RosettaNet XML Schema design decisions which are different from OAG and , consequently the mapping rules are slightly different. The authors' evolutionary approach that uses (but does not change) the integration standard and their focus on automatic validation of XML documents is similar to ours. However, the main difference is in our focus on evaluation and validation of integration results in the EAI domain. Another paper describes an initial exploration of OWL as a model-based language for integrating XML data sources [19]. In this work, OWL is introduced as a top layer of heterogeneous XML data sources. The focus here is on a query language for OWL as an extension of XQuery that may be used for hybrid reasoning (i.e., relies on procedural computation) in our approach. Recently, a new layered model for XML schemas was proposed, which offers a semantic view for XML schemas through the specification of concepts and semantic relationships among them [20]. The work introduces a transformation framework that encompasses the whole XML document transformation process, from modeling and semantic matching to transformation script generation. In this paper, conceptual modeling is used to automate the transformation algorithm. Unlike this work, that deals with diversity of schema constructs and semantic matching, our approach is based on OWL DL representation of a conceptual model using a core set of concept descriptions that may be customized. Moreover, our approach enables automated inferred relationships among concepts (in models) using logical matching of their definitions.

8. Conclusion

In this paper, we described a Semantic Web-based integration methodology to serve as a blueprint to assess capabilities of these emerging technologies to enhance syntax-based standards approaches for enterprise applications integration. In particular, we were interested to investigate possible advances in testability of integration efforts using the new technologies. This novel integration methodology is described through a scenario of integration and validation steps that are performed both at design time and run time. During design time, the methodology supports development of generalized and regular ontologies (that describe application interface models) and allow model-based similarity analysis of these ontological models. During run time, the methodology enables semantic translation of instances of business documents (conforming to

the developed ontologies) using the previously developed ontologies and automated reasoning tools.

Initial experimental results in testing the methodology show interesting capabilities such as the ability to perform individual equivalence test that is content based. Through experimental work, we have also gained a significant number of insights into the issues of necessary and sufficient conditions for achieving interoperable data exchange.

Our immediate future work will focus on experimental assessment of the initial ideas for Semantic Web-based EAI standards. The work will draw from on-going industrial standards-based integration efforts such as the ones going within STAR and AIAG industrial groups. We expect to identify key technical issues for the proposed approach, and through experimental demonstration show how such issue may or may not be addressed using the proposed approach. Our key contribution, we anticipate, will be to increase significantly understanding of whether and how Semantic Web technologies may be applied in a near future to realistic industrial integration efforts.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply these products are necessarily the best available for the purpose.

References

1. Open Applications Group (OAG) Web Site, [Online]. Available: http://www.openapplications.org/, (current November 2004)
2. RosettaNet Web Site, [Online]. Available: http://www.reosettanet.org , (current November 2004)
3. Meadows, B., Seaburg, L. (eds.): Universal Business Language 1.0, [Online]. Available: http://docs.oasis-open.org/ubl/cd-UBL-1.0/, (current November 2004)
4. Extensible Markup Language (XML), Version 1.0 (second edition). W3C Recommendation,2000, [Online]. Available: http://www.w3.org/TR/1998/REC-xml-19980210/, (current November 2004)
5. XML Schema Part 1: Structures Second Edition, W3C Recommendation 2004, [Online]. Available: http://www.w3.org/TR/xmlschema-1/, (current November 2004)
6. XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 2004, [Online]. Available: http://www.w3.org/TR/xmlschema-2/, (current November 2004)
7. Decker, S., Harmelen, F., Broekstra, et al.: The Semantic Web - on the Roles of XML and RDF . In: IEEE Internet Computing, Vol. 4,No. 5,63-74, (2000)

8. Jelliffe R.,: Schematron - Pattern-based schema language, accessed [Online]. Available: http://www.ascc.net/xml/resource/schematron/schematron.html, (current November 2004)
9. Standards for Technology in Automotive Retail (STAR) Web Site, [Online]. Available: http://www.starstandard.org/, (current November 2004)
10. Automotive Industry Action Group (AIAG) Web Site, [Online]. Available: http://www.aiag.org/, (current November 2004)
11. McGuinness, D.L., Harmelen, F. (eds.): OWL Web Ontology Language Overview [Online]. Available: http://www.w3c.org/TR/owl-features/, (current November 2004)
12. Haarslev V., Moller, R.: Description of the RACER system and its applications. In Proceedings International Workshop on Description Logics, (2001)
13. Horrocks, I., Patel-Schneider, P.F., Harmelen, F.: From {SHIQ} and {RDF} to {OWL}: The Making of a Web Ontology Language, Journal of Web Semantics, 7-26, (2003)
14. Baader, F., Horrocks, I., Sattler, U., : Description Logics as Ontology Languages for the Semantic Web: Lecture Notes in Artificial Intelligence. Springer-Verlag, (2003)
15. D.Nardi, R. J. Brachman. An Introduction to Description s. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 1- 39,(2003)
16. Baader, F., Nutt, W.: Basic Description Logics. In the Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 43-95,(2003)
17. Schmidt-Schauß, M., Smolka, G., Attributive concept descriptions with complements. Artificial Intelligance, 48(1),1-26,(1991)
18. Trastour, D., Preist, C., Coleman, D.: Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches. 7th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, (2003)
19. Lehti, P., Fankhauser, P.: XML data integration with OWL: experiences and challenges. Applications and the Internet, 2004. Proceedings. 2004 International Symposium, ,160 – 167, (2004)
20. Boukottaya, A., Vanoirbeek, C., Paganelli, F., Khaled, A.o.: Automating XML document Transformations: A conceptual modeling based approach, The First Asia-Pacific Conference on Conceptual Modeling, Dunedin, New Zealand, (2004)

Nenad Aničić is a teaching assistant and PhD Student at the Faculty of Organizational Sciences, University of Belgrade. He received his BS and MS degree from the Faculty of Organizational Sciences, University of Belgrade. His main teaching and research areas are databases systems, information system development in modern software environments, semantic technologies, and interoperable application systems.

Nenad Ivezić is a Guest Researcher at the National Institute of Standards and Technology (NIST), on assignment from the Oak Ridge National Laboratory where he is on the staff of the Applied Software Engineering Research Group. He received his BS degree from University of Belgrade and his MS and PhD degrees from Carnegie Mellon University in Pittsburgh. He was a principal investigator on industry-funded research projects involving machine learning, agent-based systems, ontology-based integration, and application integration testing. His current research interests include semantic technologies, interoperable application systems, and integration standards development and testing.