An Architecture for Semantic Enterprise Application Integration Standards

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

[1] National Institute of Standards and Technology, 100 Bureau Drive
Gaithersburg, MD, USA
{nenad.anicic|nenad.ivezic|albert.jones}@nist.gov

[2] Faculty of Organization Sciences, 154 Jove Ilica Street
11000 Belgrade, Serbia and Montenegro

Abstract. Large, industry-wide interoperability projects use syntax-based standards approaches to accomplish interoperable data exchange among enterprise applications. We are investigating Semantic Web to advance these approaches. In this paper, we describe an architecture for Semantic Enterprise Application Integration Standards as a basis for experimental assessment of the Semantic Web technologies to enhance these standards approaches. The architecture relies on our automated translation of the XML Schema-based representation of business document content models into an OWL-based ontology. Based on this architecture, we use the Semantic Web representation and reasoning mechanisms to support consistency checking of ontological constructs and constraints specified within the ontology. The proposed architecture is relevant (1) when managing multiple enterprise ontologies derived from, and dependent on, a common ontology and (2) when dealing with model-driven integration using customizable interface models and testing of such integration efforts.

1 Introduction

The scope of the effort reported in this paper is defined partially by the type of industrial problems we identify and partially by the traditional standards usage for enterprise application integration (EAI). Both are discussed below.

1.1 A Prototypical Problem

Two independent but related industry consortia have developed enterprise application integration standards in the form of business document content models. Standards in Automotive Retail (STAR), an automotive retail consortium, has developed XML Schema-based standards to encode business document content models enable message exchanges among automotive manufacturers and their retail houses. Each

STAR application adopts and implements the proposed STAR XML-based interface model [1]. Automotive Industry Action Group (AIAG), an automotive supply chain consortium, has developed XML Schema-based standards to encode its own business document content models that enable message exchanges among automotive manufacturers and their suppliers. Each AIAG application adopts and implements the AIAG interface model [2].

Both STAR and AIAG base their interface models on the same 'horizontal' XML document standard – the Open Applications Group (OAG) Business Object Documents (BODs) [3]. The OAG BODs are specifications of general XML Schema components and general aggregations that make up XML Schema-based business document content models from these components. STAR and AIAG independently use the OAG BODs specifications to customize their own business document content models and define rules of usage and constraints. Typically, usage rules and constraints are captured outside of the XML Schema using syntactic constructs such as Schematron. A significant manual task is required to identify and reconcile differences among constraints and rules of the two standards [4]. Consequently, major problems can arise whenever a STAR application attempts to exchange automotive parts ordering data with an AIAG application.

In this paper, we describe an approach to enable automated checking of compatibility among rules and constraints that are independently developed within the two (or more) standards that have a common terminology as their bases. Once this approach is implemented, we expect more capable testability of integration efforts and, consequently, more efficient application integration.

1.2 Traditional EAI Standards Architecture

Enterprise application integration (EAI) is being used extensively today. The left portion of Figure 1 shows how a traditional EAI standards architecture could be applied to our STAR-AIAG integration problem assuming, a pure XML Schema-based approach. The following steps are required to translate data and to verify the business document translation:

1. Identify and resolve manually any semantic and syntactic similarities and differences between the interface models.
2. Create two XSLT stylesheet transformations from source to target and vice versa.
3. Based on 2, apply translation to the source XML Schema interface model to obtain a business document conformant to the target XML Schema interface model.
4. Validate translation using equivalence test. This validation may be done by applying an equivalence test 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 XSLT transformations from step 2.

Validation using an equivalence test is not straightforward because issues that require capabilities beyond a simple, syntax-based equivalence checking arise often. Consider the following two examples. First, elements that are ordered differently

syntactically may, in fact, be equivalent semantically, if that order is not significant. Second, a time period can be specified either by a start date with an end date or with a start date and a duration. While they are semantically equivalent, they are syntactically quite different.

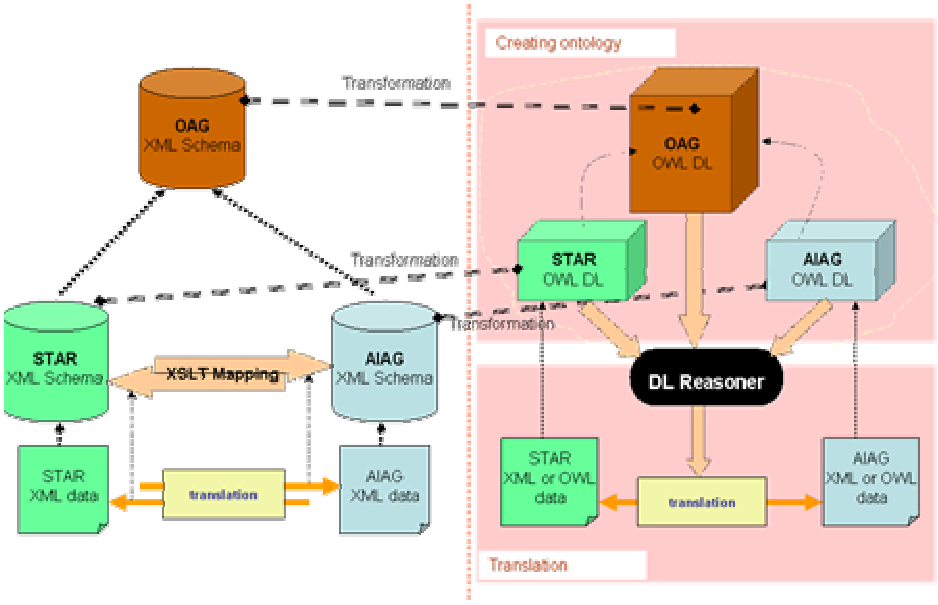

Fig. 1. Traditional and Semantic Web-based EAI Standards Architectures.

2 A Semantic Web-Based EAI Standards Architecture

For our approach, which is shown in the right portion of Figure 2, we use the OWL-DL Web ontology language to integrate enterprise applications. The language is based on a subset of the First Order Logic formalism called Description Logics. To do this, we assume that the OAG, STAR, and AIAG business document content models have been rendered into OWL-DL ontologies – a step that will be discussed in detail later in the document. This, in turn, enables us to readily use automated reasoning methods provided by DL reasoners such as Racer [5]. These reasoning methods are fundamental enablers of automated transformations, mapping and translation functions, between OWL-DL interface models that are independently developed but based on a common terminology.

The following steps are envisioned to translate and verify the translations in the proposed architecture. We provide details of executing these steps below.

- Perform model-based equivalence analysis of STAR and AIAG schemas.
    - Create ontologies of the common OAG-based general terminology and from respective XML Schemas for STAR and AIAG.

- Create a merged ontology from independently developed STAR and AIAG ontologies and check for unsatisfiability.
- Identify similarity between two schemas based on the comparison of their respective semantic models. (We assume that a high degree of equivalence may be obtained as a result of common usage of core components of the OAG standard.)

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

3 A Semantic Web-based Integration Methodology

Figure 2 illustrates the proposed Web-based integration methodology using a scenario-based view of the semantic integration architecture. The top portion shows the ontology-creation activities. These activities, which occur at design time, help us to define and test possible interoperable data exchanges. The bottom portion shows translation activities. These activities, which occur at run time, help us to reason about concrete XML data to transform the data from one format to another.

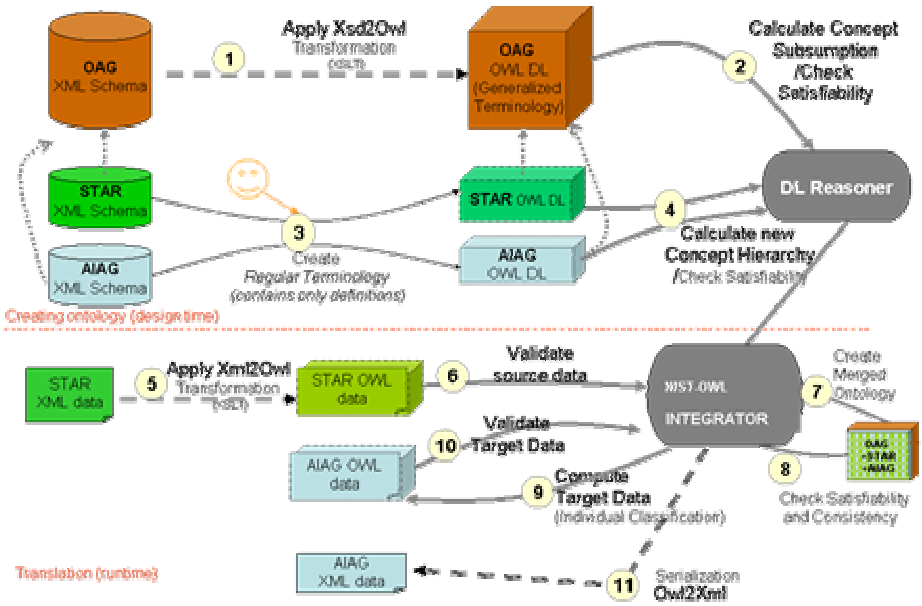

Fig. 2. Scenario-based view of the semantic integration architecture.

We give a brief summary of the sequence of the eleven activities shown in Figure 2.

1. **Apply Xsd2Owl Transformation.** We began by applying an automated transformation to the OAG XML Schema representation to obtain an OAG OWL-based generalized ontology. This is a generalized ontology that contains concept descriptions only and no definitions [1]. The automated transformation was possible because we took into account the decisions and the rationale that led to the OAG components and document design. The automated transformation is captured in the XSLT transformation rules.
2. **Calculate concept subsumption and check satisfiability of the new OAG ontology.** This results in a new subsumption hierarchy for the OAG generalized ontology and an indication from the reasoner that the new ontology is either satisfiable or not. It can be unsatisfiable for several reasons. For example, an element that is mandatory in the super-type declaration is only optional in the new declaration. All such conditions must be resolved before proceeding.
3. **Create an OAG regular terminology.** Once we have a satisfiable generalized terminology, any individual application integrator, typically a human being, can use the terminology to specify additional constraints or to provide definitions for concepts in a particular context. The original STAR and AIAG Schemas include free-text descriptions of the additional document constraints that need to be 'layered on top' of the OAG generalized terminology. For each such schema, these constraints are used to specify concept definitions based on the original concept descriptions. The outcome of this step is a regular STAR or AIAG 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. All unsatisfiable conditions must be resolved before proceeding to step 5.
5. **Apply automated transformation from source (STAR) XML data to OWL data.** This step initiates the run-time activities required for a successful and interoperable data exchange. We transform XMLSchema instances into OWL-DL individuals that conform to the OWL model-based assumptions used in ontological reasoning. The outcome is transformed STAR OWL data that corresponds to the initial XML data.
6. **Validate source data.** Validation of STAR OWL data includes consistency checking under both Open World Assumption (OWA) and Closed World Assumption (CWA). The outcome of this step, if successful, is an indication from the reasoner that the STAR OWL data are consistent with respect to the STAR ontology. An individual is valid only if it is consistent (belongs to specific concept) in both OWA reasoning and CWA reasoning. Validation is necessary to check the transformation and to check other semantic constraints, which are de-

[1] Concept refers to expressions that define a class in the OWL-DL language, which also provides constructs for establishing relationships between concepts. The meaning of concepts is specified using logical semantics, which distinguishes between concept description and concept definition. Concept description refers to a class with necessary conditions only; concept definition refers to a class with both necessary and sufficient conditions.

fined in the corresponding ontology. Examples of such constraints include additional semantic business rules based on Schematron rules and free-text descriptions provided with a schema.

7. **Create a merged ontology.** In order to translate from STAR to AIAG OWL data, we need to create a merged ontology from the two individual ones and calculate new, concept-subsumption hierarchy[2]. Because new independently defined ontologies are based on the same generalized OAG terminology, a reasoner may combine axioms and calculate the 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).
8. **Check satisfiability of the merged ontology and consistency of the source (STAR) data with the new ontology.** The successful outcome of this step is an indication from the reasoner that the merged ontology is satisfiable and, similarly, that all STAR OWL source data are consistent with respect to the merged ontology. 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 and consistency of source individuals with respect to the merged ontology.
9. **Compute classification of the source (STAR) OWL data in the target (AIAG) ontology.** Assuming successful completion of the preceding steps, then we can use the individual classification capability of a DL reasoner to compute target data in the AIAG ontology. The result is an assignment of the STAR OWL data to the specific AIAG class(es). At this point, specific STAR XML data may be successfully translated into target AIAG XML data. This, however, doesn't mean that all STAR data may be successfully translated to AIAG.
10. **Apply validation of newly created target (AIAG) OWL data**. The outcome of this step, if successful, is an indication from the reasoner that the AIAG OWL data are consistent with respect to the AIAG ontology. As discussed above, this requires OWA consistency and validation 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 an individual is inconsistent in CWA, then translation is not possible. If successful, however, then we can be sure that specific XML source data can be translated into a target OWL data and that the integration will succeed.
11. **Apply serialization of AIAG OWL data into AIAG XML data**. The outcome of this step is an AIAG XML instance that preserves semantics defined in the original STAR OWL data. 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

[2] We include this step in the run-time activities, but it could also be done at design time.

to specific XMLSchema syntax (e.g., a sequence of elements defined in separate file), then that serialization is dependent on the mapping rules.

4 Initial Findings and Lessons Learned

In this section, we discuss some initial findings and lessons learned.

4.1 Individual Equivalence Test

When dealing with B2B application integration, it is imperative to determine if two business documents are semantically equal. 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 business document instance may be transformed to individuals with different URI identifiers but the 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. We have developed a tool that helps us to do this. It is described below.

For every individual, the testing tool creates a temporary concept definition that contains the values constrained to the values specified in the individual properties. In addition, cardinality constraints on the properties of the temporary concept definition are based on the property occurrence in the particular individual. All the temporary concepts are considered by the reasoner to determine mutual equivalence. Then, for every pair of equivalent concepts, the tool asserts *sameAs* mapping between two individuals. This means that the tool creates an assertion that explicitly defines equality between the two individuals. That new assertion will help the reasoner to calculate any new equivalence. For example, two aggregate concepts may be determined equivalent based on individual concept equivalence. The process is iterative and will end when no new concept equivalence is identified.

We emphasize that a reasoner can calculate a new subsumption tree and identify new concept equivalence by taking into account both concept definitions and individual assertions.

4.2 Concept equivalence with inconsistent business document instances

We investigated whether two ontologies can facilitate interoperable data exchange and we used reasoner capabilities to perform satisfiability check between them. We determined that a necessary condition for interoperable data exchange is that there are no contradictory concepts in the merged ontology. It is, unfortunately, not a sufficient condition because some individuals may violate business constraints defined for a particular concept. Consider the following example.

Suppose a mandatory property, a necessary condition, is present within the target concept. Since the reasoner uses only definitions to calculate subsumption and equivalence among concepts and since a mandatory property is only a necessary condition, it will not be part of definition. This may give rise to an inconsistent source individual if the source concept specifies that property as optional. 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.

Human designers face a similar problem every time they create a normalized ontology for a new business document content model specification. This problem deals with establishing whether an axiom is a part of concept definition, which includes necessary and sufficient conditions, or only a part of concept description, which includes necessary conditions only. This distinction is critical because a concept definition is the mechanism used to classify concepts.

We are in a position to develop a tool that helps the designer evaluate alternative ways to specify concept description and concept definitions and to determine the potential drawbacks of each. We also plan to investigate ontology design patterns, which may avoid the "concept equivalence-individual inconsistency" type of translation problem. Additionally, we may expand default reasoner satisfiability checking to provide additional testing capability.

4.3 Lessons Learned

Based on our initial examination of Semantic Web technologies, we believe that they are sufficiently mature to allow experimental assessment in a fairly realistic enterprise-application-integration context. Our experiments used production-level XML Schema encoding of OAGIS 9.0 core components. The Xsd2Owl transformation takes a complete set of core component types from OAGIS 9.0 and is capable of creating OWL-DL ontology successfully. We worked successfully with some, but not all, examples of XML Schema definitions for OAGIS Business Object Documents (BODs).

The currently available tools are clearly not sufficiently robust and scalable to warrant risk-free development of specialized add-on tools to support industrial interoperability efforts with all the complexities typically encountered. However, the rate of maturation and adoption of these tools is encouraging and it seems that these issues of robustness and scalability may be addressed in the near future.

When planning for future usage of the Semantic Web technologies within realistic industrial enterprise application integration efforts, it is important to provide a transitioning path for moving from XML Schema-based to OWL-based application integration. While there are potentially other significant issues, one that we addressed successfully early on is to show that it is possible to translate XML instances into OWL individuals and, in the opposite direction, to serialize OWL individuals as XML instances conforming to a specific XML Schema. This capability is important when presenting these new approaches to industrial communities as it shows that the existing legacy applications do not have to change all their interfaces over night to enjoy

the benefits of this new technology. We were encouraged by the initial positive reactions from industry to the initial results from our experimental approach.

5 Conclusions

In this paper, we described a Semantic Web-based approach and a methodology to enhance syntax-based standards for enterprise applications integration (EAI). The methodology contains a number 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 normalized ontologies and allows model-based similarity analysis of these ontological models. During run time, the methodology enables semantic translation of instances of business documents using the previously developed ontologies and automated reasoning tools.

Initial findings in testing the methodology show interesting capabilities such as the ability to perform individual equivalence tests that are content based. Through experimental work, we have gained a significant insight into the issues of necessary and sufficient conditions for achieving interoperable data exchange. The lessons learned so far indicate that the Semantic Web technologies are sufficiently mature to allow experimental assessment in a fairly realistic enterprise application integration context.

Our immediate future work will focus on further 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 issues may or may not be addressed using the proposed approach. Our key contribution, we anticipate, will be to provide an increased 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. Standards for Technology in Automotive Retail (STAR) Web Site, accessed November 2004. Available at http://www.starstandard.org/
2. Automotive Industry Action Group (AIAG) Web Site, accessed November 2004. Available at http://www.aiag.org/

3. Open Applications Group (OAG) Web Site, accessed November 2004. Available at http://www.openapplications.org/
4. D.Trastour, C.Preist , and D.Coleman, *"Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches"*. 7th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2003, Brisbane, Australia, Sept 16-19th , 2003
5. V. Haarslev and R. Moller. Description of the RACER system and its applications. In *Proceedings InternationalWorkshop on Description Logics (DL-2001)*, 2001