TOWARDS SEMANTIC-BASED SUPPLY CHAIN INTEGRATION

Subtitle

Nenad Ivezic[1], Nenad Anicic[2], Albert Jones[1], Zoran Marjanovic[2]
[1]*NIST, 100 Bureau Drive, Gaithersburg, MD, 20817, USA*
Tel: +001 301 975-3536, Fax: + 001 301 975 4482,
Email: {nivezic|ajones@nist.gov}

2Faculty of Organizational Sciences, Jove Ilica 154, 11000 Belgrade, Serbia & Montenegro
Tel: +381 11 395-0800, Fax: +381 11 461-221,
Email: {anicic.nenad!marjanovic.zoran@fon.bg.ac.yu}

Abstract: We describe a novel Enterprise Application Integration (EAI) architecture based on existing B2B standards. We show how such an EAI architecture can use automated reasoning tools to provide application integration support and validation capabilities. We give a description of a transformation approach to help transition from the current XML-Schema syntax-based standards to the new OWL-based semantic standards, which are meant to support supply chain integration. Finally, we outline initial experimental results and point how one existing industrial EAI standard, the OAGIS specification, may be transformed into an OWL-based representation.

Key words: Automated reasoning, OWL, semantic integration, EXL schema

1. INTRODUCTION

Achieving interoperability among enterprise applications such as Enterprise Resource Planning, Supply Chain Management, and Inventory Visibility systems is a high priority for many manufacturing companies. On one hand, industry-wide enterprise application interoperability efforts may provide significant cost savings. For example, the Inventory Visibility and Interoperability (IV&I) project shows a potential savings of more than $250M in the U.S. automotive industry alone from making only the IV

systems interoperable [1]. Other interoperability studies [2,3] from the Capital Facilities Industry and Health Care industries point to the potential for multi-billion dollar savings. On the other hand, achieving industry-wide interoperability takes significant time and resources. In case of the IV&I project, the process of developing specifications and testing implementations has been a multi-year effort that involved tens of representatives from both the automotive and software industries.

There exists a major opportunity to move towards more capable and less costly enterprise application interoperability efforts by enhancing one of the fundamental building blocks on which these efforts rely - Enterprise Application Integration (EAI) standards. Typically, industry-specific consortia (e.g., OAG, RosettaNet) develop these standards [4,5] that use XML specifications based on syntactic formalisms. Our capabilities to accelerate, and test results of, interoperability efforts based on these standards alone are severely limited by these formalisms. In this paper, we describe a vision for semantics-based standards called Semantic Enterprise Application Integration (SEAI) standards. These new standards use formal representations and allow automated reasoning mechanisms that will provide a basis for less costly and more capable industry-wide interoperability projects.

The formal representation we will use is based on emerging Semantic Web technologies. In this paper, we will assess the potential for using these technologies to support ongoing, industrial, interoperability efforts. We summarize our methodological approach in the following way:

- Embed the new SEAI standards within a novel enterprise application integration (EAI) architecture detailing innovative integration and validation capabilities.
- Offer application integration support and validation capabilities by providing shared ontology construction and specialization, and automated reasoning capabilities.
- Assure realistic transitioning from the current EAI standards to the new SEAI standards in industrial interoperability scenarios.
- Perform capability assessment using experimental data from industrial interoperability projects.

We begin by describing this novel EAI architecture and the steps we use to implement that architecture.

2. OUR METHODOLOGY

First, we describe the essential steps involved in the novel enterprise application integration architecture that uses the proposed SEAI standard approach. Then, we provide an overview of a key enabler of the SEAI standards – the Xsd2Owl transformer.

2.1 A Novel Enterprise Application Integration Architecture: A Process View

In this section, we describe our new architecture, which we believe will enable model-based specification of business data exchanges [6, 7]. This architecture includes integration and validation steps performed both at design time and run time of an integration process.

We illustrate the architecture by considering an integration situation where two industrial consortia, STAR and AIAG [8, 9], base their interface models on the same 'horizontal' document standard – the OAGIS Business Object Documents (BODs) [4]. BODs are specifications of general XML Schema components and general aggregations from these components that make up business document content models. Each consortium independently uses the OAGIS BODs to customize their own document content models and define usage rules for the components. The problem is to recognize the differences in those customizations and build mapping between those differences. The process we developed for doing this has both design-time and run-time steps.

2.1.1 Design Time: Ontology Creation Steps

We accomplish three things at design time (see Figure 1). First, we develop a generalized ontology, which is a shared ontology. Second, we develop normalized ontologies, which describe application interface models. Third, we perform a model-based compatibility (i.e., satisfiability) analysis of these ontological models. Detailed steps are given below.

In Step 1, we apply an Xsd2Owl transformation to the OAG XML Schema representation to obtain an OAG OWL-based generalized ontology. This ontology contains concept descriptions only and no definitions. Concepts refer to expressions that define a class in the OWL-DL language, which also provides constructs for establishing relationships between

concepts. The meaning of the 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. The automated transformation was possible because we took into account the decisions and the rationale that led to the OAG components and document design. This transformation is explained in more detail in a later section.

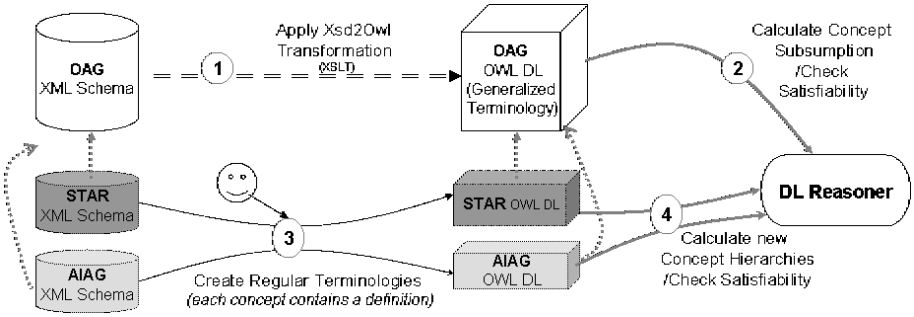

Figure 1. Ontology Creation: Design Time View of the Semantic Integration Method

In Step 2, we calculate a concept subsumption and check satisfiability of the new OAG ontology. Here, we utilize an automated reasoner to compute a new subsumption hierarchy for the OAG generalized ontology and determine whether the new ontology is satisfiable. For example, the resulting ontology would be unsatisfiable if a mandatory element in a type declaration were declared optional within a sub-type declaration.

In Step 3, the application integrators individually create regular ontologies based on the satisfiable generalized ontology created in Step 2. The integrators would use the generalized ontology and specify additional constraints or provide definitions for concepts in a particular integration context. For example, 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 ontology. For each such schema, these constraints are used to specify concept definitions based on the original concept descriptions. This step produces regular (or normalized) STAR and AIAG ontologies.

In Step 4, we calculate new concept hierarchies and check satisfiability for the newly created regular ontologies from Step 3. Similar to Step 2, we employ a reasoner to compute whether each individual ontology (i.e., regular terminology) is satisfiable. All unsatisfiable conditions are resolved before proceeding.

A DL reasoner can find a contradictory concept in the following way. For example, suppose we map the model group XML Schema concepts (e.g., choice, all, sequence) into OWL property hierarchies. Suppose further that a logical constraint is specified for a component A to state an 'exclusive or' option between two composite elements B and C. Assume that an integrator defines a new component concept A* that refines A by combining the two exclusive concepts B and C in a new element D which is now a mandatory component in A*. The reasoner will find that the new concept A* is unsatisfiable because the components of the new concepts were originally stated to be exclusive. That is, no individual of the specified new concept exists such that it satisfies all the necessary class conditions.

2.1.2 Design Time: Testing Integration Capabilities

Once we determine satisfiability of two independently defined regular ontologies, we must determine whether the 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 ones. As both ontologies use the same generalized ontology, a new subsumption hierarchy will be calculated and new relationships may emerge among concepts. We can use a DL reasoner to check satisfiability of each concept in the merged ontology. The reasoner can calculate relationships such as subClassOf or equivalent. 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 based on the particular individual assertion.

The result of this satisfiability checking can be that the interface models are compatible, incompatible, unidirectional, or unknown. If compatible, then bidirectional interoperable data exchange can occur. If unidirectional, then the exchange can only take in one direction. If the result is incompatible or unknown, a designer can provide new axioms such as conditional equivalence among concepts. New axioms might change subsumption hierarchy, produce new relationships, and may increase compatibility between two ontologies.

2.1.3 Run Time: Data Translation Steps

Figure 2 shows that, 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. Detailed steps are provided below.

In Step 1, we apply the Xml2Owl transformation from source (STAR) XML data to OWL data. We transform XML Schema instances into OWL-DL individuals that conform to the OWL model-based assumptions used in ontological reasoning. The outcome is STAR OWL data that corresponds to the initial XML data and transformed with respect to STAR ontology. The transformation rules depend only on XML Schema to OWL mapping. This means that the transformation includes annotation of XML data with corresponding ontology (e.g., STAR ontology)).

In Step 2, we validate source data be performing 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) under both OWA reasoning and CWA reasoning. Validation is necessary to check the transformation and to check other semantic constraints. Examples of such constraints include additional semantic business rules and free-text descriptions provided with a schema. Because a DL reasoner makes the open world assumption, if a mandatory property is not present, the reasoner cannot conclude that it is false (since it is wrong to assume it will never be present). Consequently, the reasoner can conclude only contradictory but not insufficient information. To assert that an instance has all mandatory properties, we must validate the instance with CWA.

In Step 3, we create a satisfiable merged ontology. To translate from STAR to AIAG OWL data, we must create a merged ontology from the two individuals and calculate a new, concept-subsumption hierarchy. Because new independently defined ontologies are based on the same generalized OAG terminology, a reasoner may combine axioms when calculating a new hierarchy. In the merged ontology, one concept might be dependent on concepts in the other ontology namespace. The merged semantics can lead to inferences over the source data that may yield unexpected results. Also, it is possible that the merged ontology is created at design time. In that case, the merged ontology will be referenced and can be reduced to only the set of concepts that is needed during the data transformation step. This step also includes satisfiability checking of merged concepts from both the source and the target ontology. The tool has to check satisfiability for every concept of the merged ontology.

In Step 4, we check (STAR) data consistency with the new merged ontology. The successful outcome of this step is an indication from the reasoner that all STAR OWL source data are consistent with respect to the merged ontology.

In Step 5, we compute target data. This is a classification of the source OWL data, which is in the STAR ontology, in the target ontology, which is AIAG. 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.

In Step 6, we validate 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 respect to the target (AIAG) concepts. If an individual is inconsistent in CWA, then translation is not possible. If successful, however, we can be sure that specific XML source data can be translated and that the integration will succeed.

In Step 7, we apply Owl2Xml 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 to specific XMLSchema syntax (e.g., a sequence of elements defined in separate file), then that serialization is dependent on the mapping rules. The algorithm for serialization takes into account information about the source XML Schema.

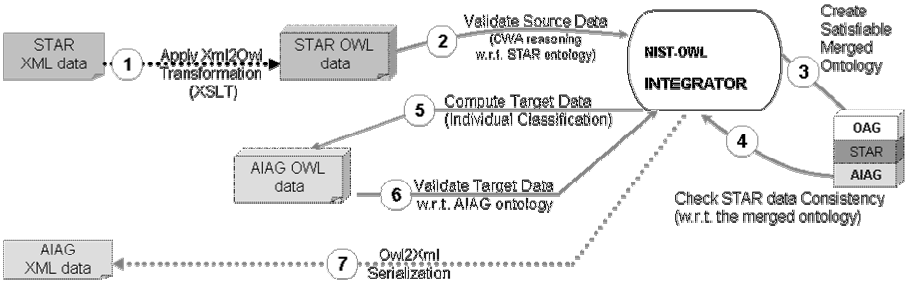

Figure 2. Data Translation: Run Time View of the Semantic Integration Method

2.2 XML Schema Transformation to OWL

The SEAIS layer uses the OWL language and is architected "on top of" the current EAI standards layer, which is based on XML Schema representation formalism. By virtue of our automated translation tools, we can take an existing XML Schema-based business document specification, transform it into an OWL-based document, and apply Semantic Web facilities and reasoning tools. Our goal is to enable industrial transitions from the current syntax-based integration world into a model-based, Semantic Web-based integration world.

To map from XML Schema to OWL, we must know the architecture of the EAI standard. OAG defines three logical levels. The first level defines core component types defined as extensions of predefined simple types. The second level introduces compound components represented as elements and attributes that can be thought of as one atomic concept. The top level contains business object documents (BODs), which define different document types.

In constructing the mapping, we must to take into account the OAG design rules for standard representation of component semantics. For example, every an OAG component is represented using a type definition. The following transformation rules have been defined on the basis of OAG design rules:

1. Each XML Schema namespace is mapped to a namespace, which is composed of schema namespace and a # sign, and included in an opening rdf:RDF tag.
2. Each XML Schema definition is mapped to owl:Ontology definition.
3. Each included namespace is defined as owl:imports, which provides an include-style mechanism for importing an entire set of assertions provided by that ontology into the current ontology.
4. The XML Schema type definitions (either simple or complex) are mapped to OWL classes. Each XML Schema component name is used as the corresponding class name. The OWL class that represents a simpleType has a functional datatype property that represents the value.
5. The extension and restriction definitions are mapped to OWL subClassOf relationships.
6. Global element declarations are mapped to OWL classes. (A global name element that has a corresponding type with the same name is skipped.)
7. Local element declarations are mapped into object properties.

8. The XML Schema attributes are mapped to OWL functional properties. The type of property depends on the definition of an attribute (e.g., user-defined or predefined).
9. XML Schema model groups: all, choice, and sequence are mapped to hierarchies of properties. Some of constraints are mapped to class description constraints.
10. All other information about elements (e.g, annotation, document) are mapped to annotation properties.

Details of the transformation rules will be described in a future publication. It is important, however, to note that this transformation provides a basis for introducing additional application domain semantics.

3. RESULTS

To date, we have investigated issues in determining individual and concept equivalences [6]. To determine if two business documents are semantically equal, we have developed a testing tool that creates a temporary concept definition for every individual. That definition 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 equivalence among the corresponding individuals. Then, for every pair of equivalent concepts, the tool asserts a sameAs mapping between the two individuals. This means that the tool creates an assertion that explicitly defines equality between the two individuals. That new assertion helps the reasoner to calculate any new equivalence. The process is iterative and ends when no new concept equivalence is identified.

Also, 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 constraints defined for a particular concept. 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. Based on this result, we plan to investigate ontology

design patterns, which may avoid the "concept equivalence-individual inconsistency" type of translation problem.

We successfully transformed a version of OAG components from the OAG XML Schema standard representation into an OWL representation. Figure 3 shows a graphical rendering of a part of the transformed OAG hierarchy using Protégé OWLViz plug-in.

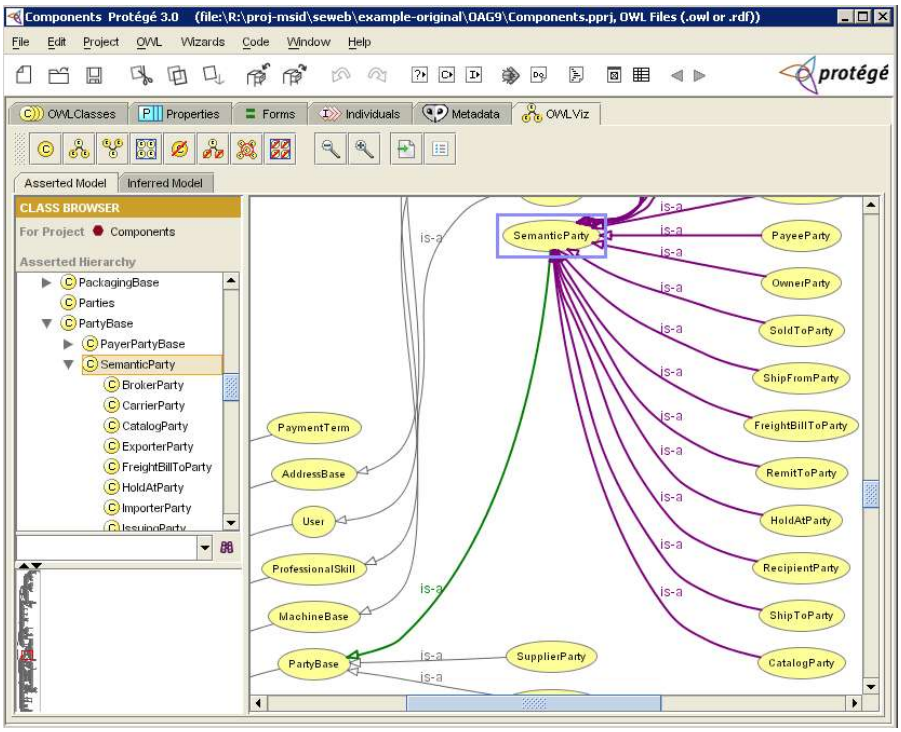

Figure 3: A Graphical Rendering of an OWL Representation of OAG Components

4. RELATED WORK

A previous effort investigated use of Semantic Web technologies (e.g., DAML+OIL) in support of semantic constraint definitions and management for RosettaNet [10]. That effort suggested an approach for mapping from XML Schema to DAML+OIL. It used RosettaNet XML Schema design decisions, which are different from OAG and, consequently, the mapping rules are slightly different. That approach is similar to ours, but our focus is on evaluation and validation of integration results in EAI standards domain

The on-going work on semantic annotation presents an important, complementary effort to ours [11]. This work attempts to expose the actual semantics of applications by making them explicit in the application interface. This work relies on the notion of a reference ontology, which, in our work, is provided by the common generalized ontology. In Step 3 of the Ontology Creation, the application integrators, which creates regular ontologies based on the satisfiable generalized one, would use semantic annotation tools to create effective normalized and/or regular ontologies.

5. CONCLUSIONS

We described a novel Enterprise Application Integration (EAI) architecture based on Semantic Enterprise Application Integration (SEAI) standards. We showed how such an architecture can be supported by automated reasoning tools and a transformation approach from the current XML Schema-based EAI standards to the new OWL-based SEAI standards. Based on our initial experimental assessment and showed how an existing industrial EAI standard, the OAGIS specification, may be transformed into an OWL-based representation. Our future work will focus on experimental assessment of the SEAI standards approach in collaboration with industrial consortia and standards organizations Our long-term vision is a reusable, coherent, and model-driven EAI architecture enabled by a collection of fundamental enablers: SEAI standards, automated validation approaches, application interface model definition and testing tools, and interoperability mediators.

6. 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.

7. REFERENCES

[1] Moad J., "Going Once, Going Twice," http://www.managingautomation.com/maonline/magazine/read.jspx?id=3276814, April, 2005

[2] Gallaher M., et al., "Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry," http://www.bfrl.nist.gov/oae/publications/gcrs/04867.pdf, April, 2005.

[3] David Brailer, “Interoperability: The Key To The Future Health Care Systems,” http://content.healthaffairs.org/cgi/content/full/hlthaff.w5.19/DC1, April, 2005.
[4] Open Application Groups, , http://www.openapplications.org April, 2005.
[5] RosettaNet, http://www.rosettanet.org, April, 2005.
[6] Anicic N., Ivezic N., and Jones A., “An Architecture for Semantic Enterprise Application Integration Standards,”Pre-proceedings of INTEROP-ESA’05, Available at http://interop-sa05.unige.ch/INTEROP/Proceedings/InteropESAScientific/OneFile/InteropESAproceedings.pdf.
[7] Anicic N., Marjanovic Z., Ivezic N., Jones A., “Semantic Enterprise Application Integration Standards,” Submitted to the International Journal of Manufacturing Technology and Management.
[8] Standards for Technology in Automotive Retail (STAR), http://www.starstandard.org, April 2005.
[9] Automotive Industry Action Group (AIAG), http://www.aiag.org., April 2005.
[10] 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).
[11] Missikoff M., Schiappelli F., and Taglino F., “A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications,” Proceedings of the 2nd International Semantic Web Conference, 2005, Available at http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-82/SI_paper_13.pdf.