Semantic enterprise application integration standards

Nenad Anicic* and Zoran Marjanovic

Faculty of Organizational Sciences,
University of Belgrade, Jove Ilica 154,
Belgrade 11000, Serbia-Montenegro
E-mail: anicic.nenad@fon.bg.ac.yu E-mail: zormar@fon.bg.ac.yu
Fax: +381-11-461-221
*Corresponding author

Nenad Ivezic and Albert Jones

National Institute of Standards and Technology,
100 Bureau Drive,
Gaithersburg, MD 20899, USA
E-mail: nivezic@nist.gov E-mail: ajones@nist.gov
Fax: +1-301-975-4482

Abstract: This paper investigates the potential of the Semantic Web technologies to support a semantic-based Enterprise Application Integration (EAI) standards architecture. We give detailed information of the support that these technologies and the underlying Description Logics (DL) formalism provide for the integration task. Our main aim is to assess the potential impact of these emerging technologies on industrial interoperability efforts. In addition to that effect, we plan to use this advanced EAI standards architecture as an experimental framework in which the Semantic Web technologies are evaluated on realistic enterprise integration problems. We illustrate novel capabilities beyond the existing syntactic integration approaches when managing multiple enterprise ontologies derived from a common ontology.

Keywords: Enterprise Application Integration (EAI); semantic technologies; standards; e-business; Semantic Web; XML; OWL.

Reference to this paper should be made as follows: Anicic, N., Marjanovic, Z., Ivezic, N. and Jones, A. (XXXX) 'Semantic enterprise application integration standards', *Int. J. Manufacturing Technology and Management*, Vol. X, No. Y, pp.XXX–XXX.

Biographical notes: Nenad Anicic is a PhD candidate at the Faculty of Organizational Sciences, University of Belgrade, Serbia and Montenegro. He works as a Teaching Assistant at the Faculty of Organizational Sciences and his main research areas are databases and advanced information systems development.

Zoran Marjanovic, PhD, is an Associate Professor at the Faculty of Organizational Sciences, University of Belgrade, Serbia and Montenegro. He teaches database and information systems design courses. His main research areas are databases and advanced information systems development.

Copyright © XXXX Inderscience Enterprises Ltd.

Nenad Ivezic, PhD, is a Guest Researcher of the Enterprise Systems Group in the Manufacturing Systems Integration Division of the National Institute of Standards and Technology from the Oak Ridge National Laboratory. His main research areas are semantic software systems and interoperability.

Albert Jones, PhD, is the Leader of the Enterprise Systems Group in the Manufacturing Systems Integration Division of the National Institute of Standards and Technology. His main research interests are advanced enterprise systems, interoperability, next generation control, simulation and scheduling systems.

1 Introduction

Nowadays, the success of industry-wide enterprise integration efforts depends on the syntactic, XML-based Enterprise Application Integration (EAI) standards (Meadows and Seaburg, 2004; Open Applications Group (OAG), 2004; RosettaNet, 2004). Capabilities of these standards to support application integration efforts are significantly limited by the restricted reasoning capabilities of the syntactic formalisms (Decker et al., 2000). For example, business document content rules that are perfectly valid syntactically may state conflicting semantic requirements in an application integration context.

Semantic Web formalisms allow use of computational approaches to reason about formally expressed concepts and make inferences that are beyond the capabilities of the syntax-based approaches. The reasoning methods, such as satisfiability and consistency checking, may be performed to check whether two business document schemas are compatible and whether a business document instance belongs to a specific class of documents.

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 (DL) (McGuinness and Harmelen, 2004; Nardi and Brachman, 2003). As an example, it is possible to express constraints on 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 syntactic approaches, yet they are essential in the EAI tasks.

This paper investigates the potential of the Semantic Web technologies to support a semantic-based EAI standards architecture. We give detailed information on the support that the Semantic Web technologies and the underlying DL formalism provide for the EAI and validation tasks. Our main aim is to assess the potential impact of these emerging technologies on industrial interoperability efforts. In addition to that effect, our aim is to define a semantic-based EAI standards architecture for an experimental framework in which Semantic Web technologies are evaluated on realistic enterprise integration problems. While in this paper, we focus on showing formally how Semantic Web technologies support the EAI task, a previous publication illustrates usage of these technologies on an example application integration task and functionalities supported by the approach (Anicic and Ivezic, 2005).

The rest of this paper is structured as follows: Section 2 describes a specific integration context motivating our work, a traditional EAI standards architecture and the proposed Semantic Web-based EAI standards architecture. Section 3 gives an overview of the Semantic Web terminology and formalism relevant to our integration methodology. Section 4 gives illustrations of the developed integration methodology and a detailed description of the supporting DL framework. Section 5 includes a description of the related work. Finally, Section 6 provides concluding remarks.

2 A Semantic Web-based architecture for EAI standards

In this section, we introduce an EAI problem context that motivates this work. We then compare a traditional and a novel semantic-based EAI standards architecture.

2.1 An EAI problem context

By integration of enterprise applications, we denote exchange of *business document instances* 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. *Interoperable data exchange* is such an exchange of data that preserves intended meaning of that data.

The integration context we consider is one where two industrial consortia, such as Standards for Technology in Automotive Retail (STAR, 2004) and Automotive Industry Action Group (AIAG, 2004), base their interface models on the same 'horizontal' document standard – the OAGIS Business Object Documents (BODs) (OAG, 2004). BODs are specifications of general XML Schema components and general aggregations that form business document content models from these components. Each consortium independently uses the OAGIS BODs to customise their own document content models and define usage rules for the components (e.g. mandatory and conditional components).

Nowadays, the usage rules for the business document content models are captured outside the XML Schema using syntactic constructs (e.g. Schematron rules) (Jelliffe, 2004). A significant manual task is required to reconcile differences among constraints and rules of two or more standards. We seek a semantic-based approach to enable automated checking of compatibility among rules and constraints that are independently developed in two or more standards groups with a common terminology at their bases. Additionally, the semantic-based approach needs to support interoperable data exchange among applications from independent business contexts.

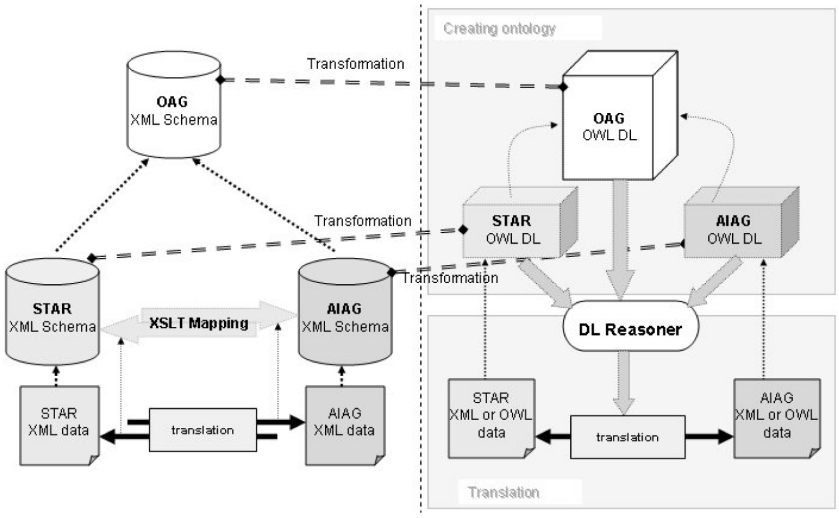
2.2 Traditional EAI standards architecture

The left portion of Figure 1 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:

- identify and resolve manually any semantic and syntactic differences for implementations of the STAR and AIAG XML Schema interface models
- create two Extensible Stylesheet Language Transformation (XSLT) sheet transformations from the source to the target XML Schema interface model and vice versa
- 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 style sheet transformations)
- validate the translation:
  - validate translated business documents with respect to the target XML Schema interface model (using syntactic approaches such as Schematron rules)
  - validate translation using an 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 the second step above.

Figure 1 Traditional and Semantic Web-based EAI standards architectures



Applying the two translations in sequence (using different mechanisms) and comparing the final source business document to the initial source business document is problematic when using only 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 another example, an equivalent time period can be specified either by a start date with:

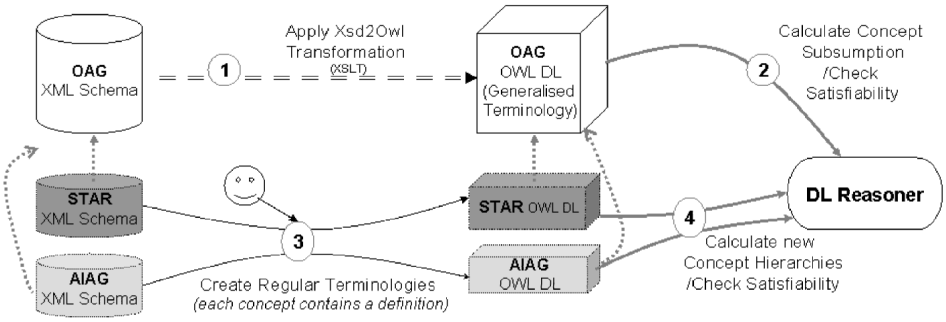
1 an end date or

2 a duration of time period.

2.3 A Semantic Web-based EAI standards architecture

The right portion of Figure 2 shows the proposed Semantic Web-based EAI standards architecture with the Description Logics version of the OWL Web Ontology Language (OWL-DL) employed to formally define business document content models (McGuinness and Harmelen, 2004). This, in turn, enables us to readily use automated reasoning methods provided by DL reasoners (e.g. Racer; Haarslev and Moller, 2001). 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 base terminology.

Figure 2 Ontology creation: design time view of the semantic integration method



As in the 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:

- perform model-based equivalence analysis of STAR and AIAG schemas. The following steps are involved:
  - 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 semantic models using an automated inference tool
- 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 with respect to the merged (source-STAR + target-AIAG) ontology
  - classify the source OWL individual into the target ontology (AIAG) and perform validation and serialisation.

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

3 A Semantic Web terminology and formalism overview

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 the DL 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).

The use of the DL 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 $SHOIN(D_n)$ an expressive DL (Horrocks et al., 2003), with an ontology equivalent to a DL knowledge base. An OWL essential feature is that it uses a DL style model theory to formalise the meaning of the language. To define formal semantics of OWL DL as a DL model, we consider the semantics of concepts in terms of an interpretation $I = (\Delta^I, o^I)$ that consists of a domain of interpretation (nonempty set) Δ^I and an interpretation function o^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.

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

- *Calculating a concept satisfiability* determines whether the concept description is not contradictory with the rest of an ontology. A concept is satisfiable if it has a model for a concept that is nonempty; otherwise the concept is unsatisfiable.
- *Checking consistency of an individual* (with respect to this concept description) means determining whether the individual is an instance of a concept. A DL knowledge base usually consists of a set of terminological axioms and a set of assertions. An individual is an instance of a concept if and only if it satisfies all constraints specified for the concept definition.

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

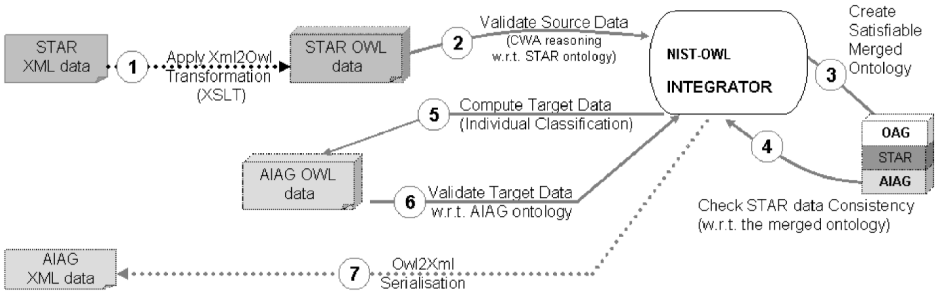
- *subsumption computation* determines whether a concept description is more general than another one
- *individual classification* determines the most-specific concept for the particular individual.

4 Semantic Web-based integration methodology

In this section, we give detailed information on the two phases of the proposed Semantic Web-based integration methodology. Figure 2 illustrates the Ontology Creation phase where we determine if interoperable data exchange among different adopted XML Schemas (e.g. STAR and AIAG schemas) is possible. This phase occurs at design time.

Figure 3 illustrates the Data Translation phase where we reason about concrete XML instances (based on the adopted XML Schemas) to determine the possibility for interoperable data exchange among different adopted XML Schemas using these XML instances. This phase occurs at run time. Next, we discuss both of these phases in detail.

Figure 3 Data translation: run time view of the semantic integration method



4.1 Apply Xsd2Owl transformation

An automated transformation was devised for the OAG XML Schema representation to obtain an OAG OWL-based generalised 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 consideration design rules for the OAG components and document specifications. Figure 4 shows some of these rules.

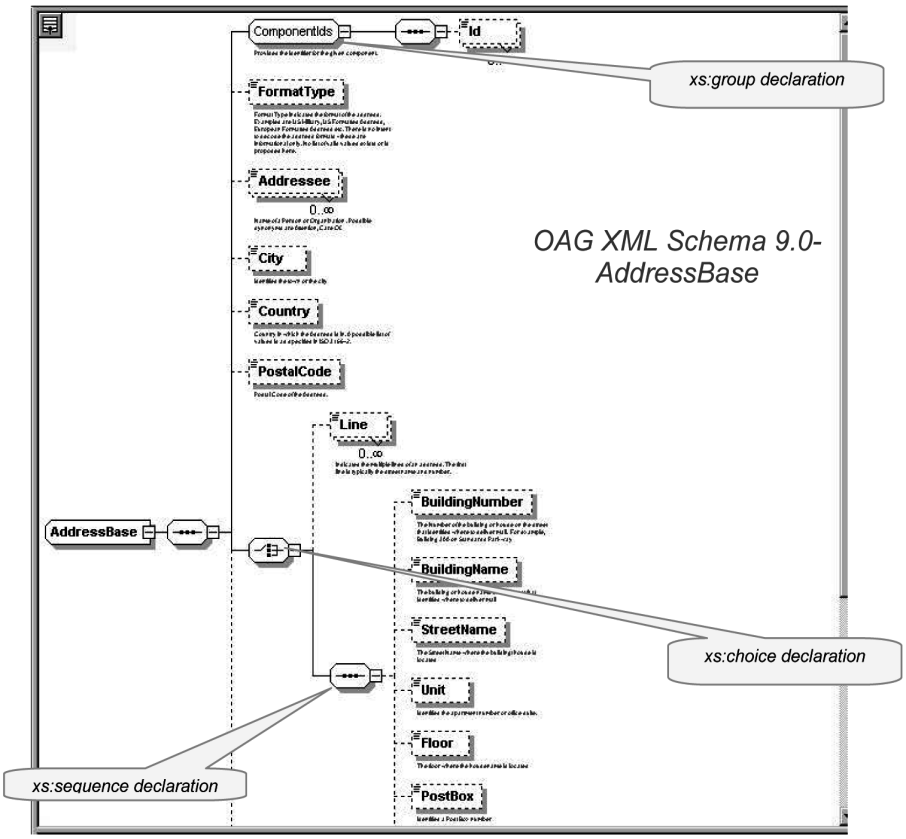
Figure 4 Example transformation rules from OAG XML Schema into OAG OWL-DL generalised ontology

- **The global** (root) **schema *element*, *complexType*** *and* ***simpleType*** **declaration** are mapped to OWL class.
  - "simpleType" OWL class is assigned functional datatype property with name composed of component's name and '*Value*'.
- **Simple types defined as enumeration** are mapped to OWL *enumerated class,*
- Every **inner element declaration** is mapped into ObjectProperty.
- **Attributes** are mapped to functional property. (The type of property (object or datatype) depends on definition of attribute).
- Every **model group** is mapped into a logical constraint for the particular OWL class. *Hierarchy of properties has been created to capture relationship between model groups but also to enable easy definition of constraints.*

An illustration of application of the transformation from XML Schema to OWL (Xsd2Owl) transformation rules are shown in Figures 5–6. Figure 5 shows a rendering of

the XML Schema for the OAG aggregate component `AddressBase`. (The rest of this paper uses this component to illustrate the methodology.) The `AddressBase` component describes all possible elements that an OAG `Address` instance (that uses `AddressBase`) may have. For that reason, all elements are optional in the complexType definition of the `AddressBase`.

Figure 5 A rendering of the XML Schema for the OAG `AddressBase` component

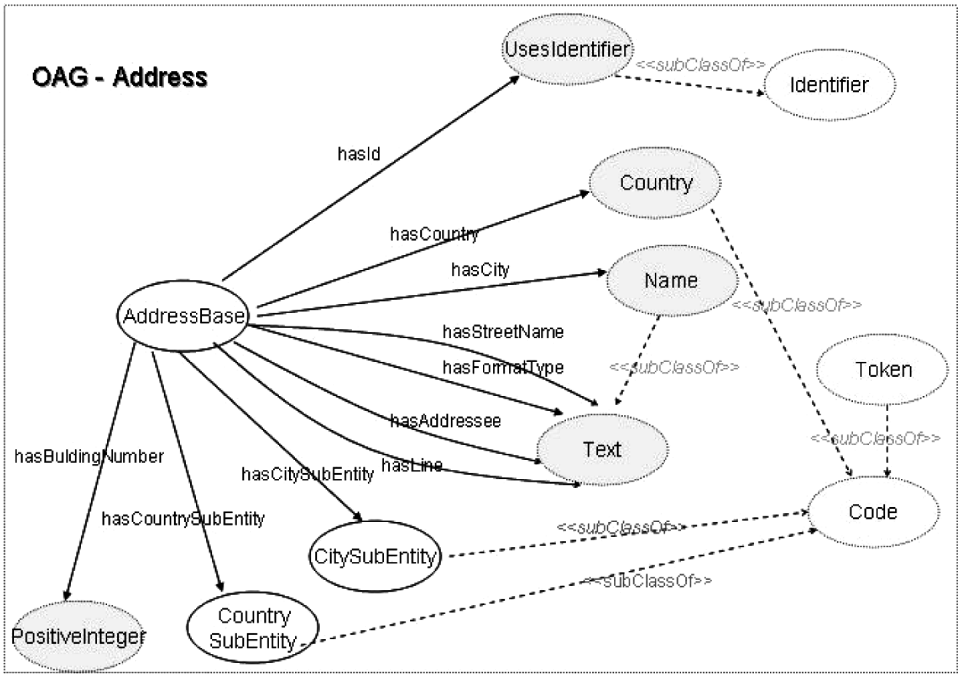


In the `AddressBase` schema definition, the component `ComponentIds` is a named model group definition. We can also see choice between 'unstructured' `Line` and the 'structured Line' including `StreetName`, `BuldingNumber` and `Floor`. We capture this constraint in the resulting OWL description by using a hierarchy of properties.

Figure 6 shows a graphical representation of the OWL-DL model obtained through the transformation of the `AddressBase` XML Schema. This figure shows all property restrictions for the OWL class `AddressBase` including strictly specified ranges that are extensions of other concept descriptions (e.g. `Identifier` and `Code` classes).

Using the DL formalism, let us define terminology T as a set of axioms. Then, an interpretation I satisfies T iff I satisfies each element of T. The generalised terminology is such a T where all axioms are defined as inclusions. An interpretation I satisfies an inclusion $C \sqsubseteq D$ if $C^I \subseteq D^I$.

Figure 6 A graphical representation of the transformed AddressBase OWL structure



Consider a terminology T_1 that contains all atomic concepts that specify aggregate component `AddressBase`. The generalised terminology T_1 contains basic concepts such as `UsesIdentifier`, `Text`, `Name`, `Country`, `SequenceText`, and the following complex axiom description:

```
AddressBase ⊑ (∀ hasId.UsesIdentifier)
⊓ (∀ hasFormatType.Text) ⊓ (≤1 hasFormatType)
⊓ (∀ hasAdressee.Text) ⊓ (∀ hasCity.Name)
⊓ (≤1 hasCity) ⊓ (∀ hasCountry.Country)
⊓ (≤1 hasCountry) ⊓ (∀ hasPostalCode.Code)
⊓ (≤1 hasPostalCode)
⊓ (∀ hasLine.SequenceText)
⊓ (∀ hasBuldingNumber.Text)
⊓ (∀ hasBuldingName.Text)
⊓ (∀ hasStreetName.Text) ⊓ (∀ hasUnit.Text)
⊓ (∀ hasFloor.Text) ⊓ (∀ hasPostBox.Text)
⊓ (¬(≥1 hasLine) ⊔  ¬(≥1 sequence63736952))
⊓ (¬ (≥1 sequence63736952)
    ⊔ ((≤1 hasBulidingNumber)
       ⊓ (≤1 hasBuildingName)
       ⊓ (≤1 hasStreetName) ⊓ (≤1 hasUnit)
       ⊓ (≤1 hasFloor) ⊓ (≤1 hasPostBox))).
```

The generalised terminology T_1 also consists of a set of role axioms (a role hierarchy):

```
hasId ⊑  componentIds
hasLine ⊑  choice49723144
sequence63736952 ⊑ choice49723144
hasBuildingNumber ⊑ sequence63736952
hasBuildingName ⊑ sequence63736952
hasStreetName ⊑ sequence63736952
hasUnit ⊑ sequence63736952
hasFloor ⊑ sequence63736952
hasPostBox ⊑ sequence63736952
```

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

4.2 Calculate concept subsumption and check satisfiability

A DL reasoner may calculate concept subsumption and check whether any concept description is contradictory in the resulting ontology. For example, an unsatisfiable concept 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).

Probably the most popular approach to calculate subsumption in DL is called tableau-based algorithm (Schmidt-Schauß and Smolka, 1991). The algorithm instead of directly testing subsumption of concept descriptions $C \sqsubseteq D$, reduces to checking unsatisfiability of axiom $C \sqcap \neg D$. 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. We say that a generalised ontology T is satisfiable if every concept in T is satisfiable.

4.3 Create regular or normalised terminologies

Once we have a satisfiable generalised terminology, every individual application integrator may independently use the terminology to specify additional constraints and definitions for the data entities used in a particular business context.

If a terminology T is a set of inclusion axioms that also define a set of atomic concepts (i.e. the meaning of the name symbols is completely determined), then T' is a normalised terminology of T iff all concepts in T' are defined as extensions of model T in which all concept specifications[1] agree with the atomic concept and role axioms in T. We say that T' is normalised terminology of T if T' and T share the same model I. The regular terminology is a normalised terminology T' that contains only concept definitions that agree with the atomic concepts and roles in T.

Terminological axioms that represent defined concepts are given in a form called equality (≡). With such axioms, we associate the left-hand side concept name `Address` to the description on right-hand side `AddressBase` with two cardinality constraints. An interpretation I satisfies $C \equiv D$ if $C^I = D^I$, that is, two sets of axioms are equivalent if they have the same model.

Here, Address is defined as AddressBase with mandatory properties hasCity and hasCountry:

$$\texttt{Address} \equiv \texttt{AddressBase} \sqcap (\geq 1\ \texttt{hasCity}) \sqcap (\geq 1\ \texttt{hasCountry})$$

4.4 Check satisfiability of the normalised terminologies

For a created normalised terminology, a DL reasoner will calculate a new subsumption hierarchy. (All OAG concept descriptions (axioms) are imported into the new ontology). In the new hierarchy, if all concepts are satisfiable (i.e. non-contradictory), then this terminology can be used for application integration.

Let us show an example how a DL reasoner can find a contradictory concept. As we said before, the model group concepts (e.g. choice, all and sequence) are mapped into property hierarchies. Suppose that a logical constraint is specified for the OAG AddressBase component to state an 'exclusive or' option between an unstructured (free) text address line and a structured line (that contains hasStreetName, hasLine, 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 in Section 4.1) and hasLine using the OAG AddressBase defined above, a reasoner will find that the concept is unsatisfiable. That is, no individual of the specified class exists such that it satisfies all the necessary class conditions.

Formally, let T be a generalised terminology that contains all OAG concept descriptions and T' be a regular terminology of T; then T' is satisfiable if all concepts in T' are satisfiable with respect to all axioms in T and T'.

For instance, let us define a regular terminology T' with a single axiom (refer to the AddressBase description in Section 5.1):

$$\texttt{Address} \equiv \texttt{AddressBase} \sqcap (\geq 1\ \texttt{hasCity}) \sqcap (\geq 1\ \texttt{hasCountry}) \sqcap (\geq 1\ \texttt{hasLine}) \sqcap (\geq 1\ \texttt{hasStreetName})$$

To check the regular terminology T', we need to check satisfiability for every axiom in T'. As every equality C ≡ D can be specified as two inclusion axioms (C ⊑ D and D ⊑ C), checking satisfiability can be transformed to checking subsumptions of the two axioms. If any of these subsumptions does not hold, the concept is unsatisfiable.

From the definition of Address we can see that for every individual that is an instance of Address we must have a filler for each of the roles hasStreetName and hasLine. On the other hand, the AddressBase description is defined so that if we have hasLine we cannot have hasStreetName or any other roles that belong to sequence63736952 (this fact is obviously defined in the role hierarchy). Then, we can simplify the subsumption problem to check the following:

$$(\geq 1\ \texttt{hasLine}) \sqcap (\geq 1\ \texttt{hasStreetName}) \Rightarrow \neg(\geq 1\ \texttt{hasLine}) \sqcup \neg(\geq 1\ \texttt{hasStreetName})$$

To do this, check whether

$$\textit{Descr0} = (\geq 1\ \texttt{hasLine}) \sqcap (\geq 1\ \texttt{hasStreetName}) \sqcap \neg(\neg(\geq 1\ \texttt{hasLine}) \sqcup \neg(\geq 1\ \texttt{hasStreetName}))$$

is unsatisfiable, as discussed in Section 4.2.

First, we push negation into expression using de Morgan's rules, and we obtain:

$$Descr1 = (\geq 1\ hasLine) \sqcap (\geq 1\ hasStreetName) \sqcap (\geq 1\ hasLine) \sqcap (\geq 1\ hasStreetName)$$

After simplification ($P \sqcap P = P$) we get:

$$Descr2 = (\geq 1\ hasLine) \sqcap (\geq 1\ hasStreetName) \quad (1)$$

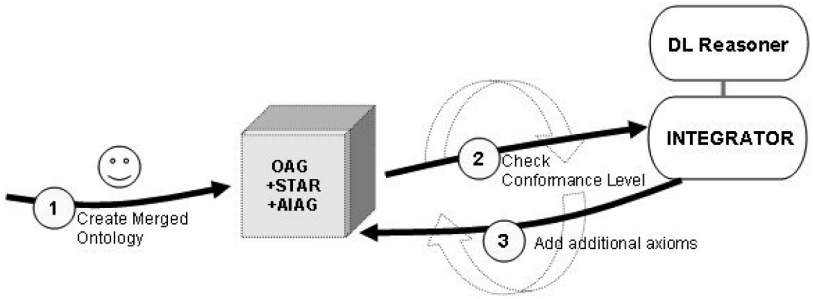
Next, we try to find a finite interpretation I such that $Descr2 \neq \emptyset$. The system will generate an individual and state that it belongs to $Descr2$. From (1) we can infer that there must be an individual that has fillers for both roles $hasLine$ and $hasStreetName$. At this point, we have not found a clash and we have a model – an interpretation that satisfies the constraint. In other words, an instance of $Address$ does not belong to $AddressBase$, subsumption relationship does not hold, and the given concept is unsatisfiable.

4.4.1 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. As both ontologies use the same generalised terminology, a new subsumption hierarchy will be calculated and new relationships may emerge among concepts. A DL reasoner is used to check satisfiability of each concept in the merged ontology. If there are no contradictory concepts, then we can say that two interface models may support interoperable data exchange. Figure 7 shows this testing step.

Figure 7 Testing for necessary integration conditions



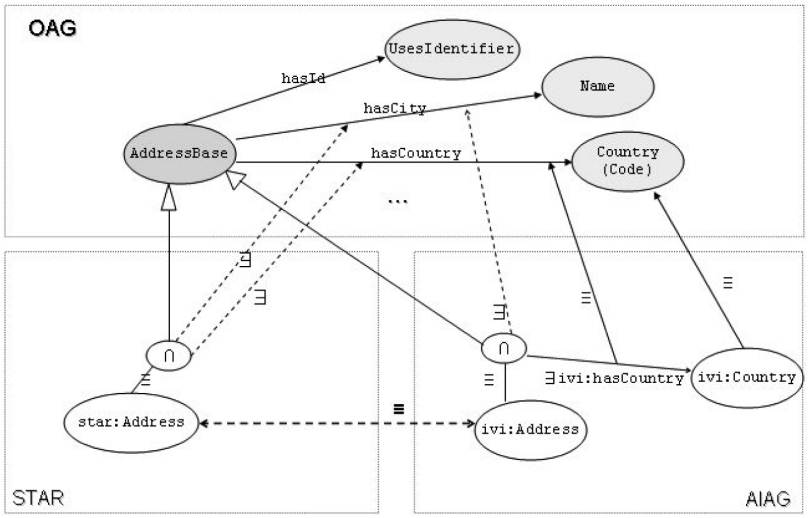
A reasoner can calculate relationships such as *subClassOf* or *equivalent*. When the subsumption or equivalency relationship cannot be calculated (i.e. when *subClassOf* or *equivalent* relationships do not hold for two concepts), an individual may still be classified to belong to either one or both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

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 Figure 7. New axioms might change subsumption hierarchy, produce new relationships and may increase compatibility between two ontologies.

An example of equivalence is illustrated in Figure 8. On the top we have OAG `AddressBase` description. On the lower left side we can see a definition of `star:Address` using only the OAG provided property and concept descriptions. The `star:Address` is given as a concept equivalent to a conjunction of axioms: subclass of `AddressBase` and cardinality at least 1 for properties `hasCity` and `hasCountry`.

Figure 8 Example of equivalence between a STAR and an IV&I concept



The right portion of the figure shows the AIAG Inventory Visibility and Interoperability (IV&I) project's definition of the `ivi:Address` that introduces a new property `ivi:hasCountry` with range class `ivi:Country`. We have also defined a relationship between the new concept and the property introduced in `ivi:Address` to the corresponding OAG concept and property. While these two address definitions might look different at first, they are equivalent.

Formally, let us define a regular terminology $T_{\texttt{star}}$ as an extension of the generalised terminology $T_{\texttt{oag}}$. The terminology $T_{\texttt{oag}}$ represents a generalised OAG ontology, which contains `AddressBase` components as defined above. The name of concept is defined as an URI reference.

Let us define $T_{\texttt{star}}$ terminology as following:

```
star:Address ≡ oag:AddressBase  ⊓ (≥1 oag:hasCity)
               ⊓ (≥1 oag:hasCountry)
```

Let us define another regular terminology $T_{\texttt{ivi}}$ (also an extension of the generalised terminology $T_{\texttt{oag}}$) which contains these axioms:

```
ivi:Address ≡ oag:AddressBase ⊓ (≥1 oag:hasCity)
              ⊓ (≥1 ivi:hasCountry)
ivi:hasCountry ≡ oag:hasCountry
```

To check compatibility between the two terminologies $T_{\texttt{star}}$ and $T_{\texttt{ivi}}$ is to check whether they share the same model (i.e. every interpretation I which is model of $T_{\texttt{star}}$ is a model of $T_{\texttt{ivi}}$ and vice versa). If two ontologies have a same model, then we can say they are compatible. The compatibility level depends on inferred relationships among the concepts in terminologies.

To check compatibility between two terminologies we define a merged ontology T' that contains all axioms from the two terminologies. In our example, based on the definition of the `Address` concept using subsumption checking we can conclude:

$$\texttt{star:Address} \equiv \texttt{ivi:Address}$$

Let us change the $T_{\texttt{ivi}}$ terminology by removing the role `ivi:hasCountry`. Then, definition of `ivi:Address` contains `ivi:hasCountry` and $T_{\texttt{ivi}}$ contains only one axiom:

$$\texttt{ivi:Address} \equiv \texttt{oag:AddressBase} \sqcap (\geq 1\ \texttt{oag:hasCity})$$

Checking subsumption between the Address axioms (in the newly merged terminology T′), we can conclude that `star:Address` is subsumed by `ivi:Address`, while the opposite subsumption does not hold:

$$\texttt{star:Address} \sqsubseteq \texttt{ivi:Address}$$

In this case, the `ivi:Address` definition is more general than `star:Address` that allows us to conclude that a unidirectional translation is possible. In other words, it may only be possible to translate from a specific STAR message into a general IVI message while an opposite translation is undefined. The definition of the IVI address does not exclude usage of optional properties, which means that there may exist an individual, which is an instance of `ivi:Address` with property `oag:hasCountry` (or `ivi:hasCountry`), and that individual can be classified as `star:Address` (i.e. an individual will be successfully translated into a STAR individual). This is to say that some business document instances, because of their usage of *optional* properties, may still be translated in the opposite direction.

On the basis of the equivalence testing capability illustrated above (such as between addresses), we may recursively determine equivalence between complex components (i.e. classes) and whole business document schemas such as OAG BODs.

4.5 Transforming source data into OWL individuals

We transform XML Schema instances into OWL-DL individuals to conform with the assumptions used in the ontological reasoning (i.e. satisfiability checking). The translation rules include the following:

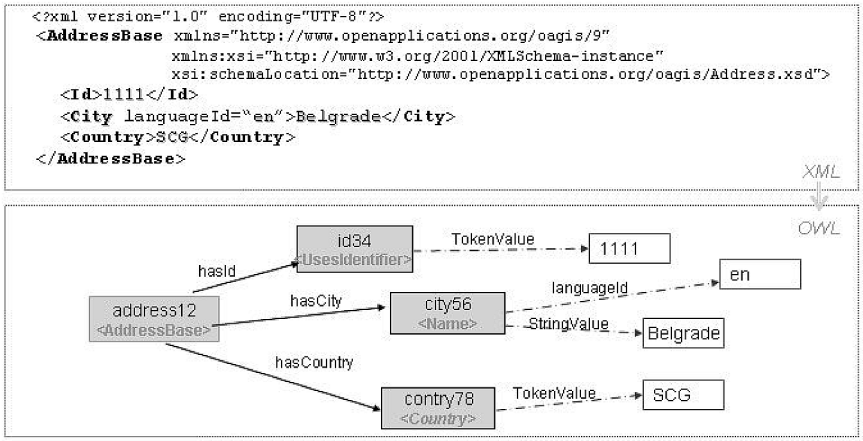
- for every element (including root element), create an OWL individual of the corresponding type.
- parent–child relationships are translated to class-property relationships: every child element is a value of the respective property of its parent class.
- the text content of an element/attribute is mapped into datatype property with an RDF (i.e. Resource Description Format)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. Two individuals are content equivalent if they have identical content (property values).

According to the previously defined rules, an AddressBase XML Schema instance is transformed in the following way, as shown in Figure 9:

- For each `AddressBase` element (i.e. `Id`, `City` and `Country`), we create an OWL individual with generated ID and a corresponding type that was defined previously in the Xsd2Owl transformation.
- For each `AddressBase` element (i.e. `Id`, `City` and `Country`) we define respective properties (i.e. `hasId`, `hasCity` and `hasCountry`) with corresponding values being the previously (step 1) created OWL individuals.
- The content of each element is related to a corresponding datatype property (e.g. the Id data is related to `TokenValue` with RDF literal "1111" which represents the value for that particular property).

Figure 9 An example transformation of XML data into OWL individuals



A DL knowledge base $\Sigma(T, A)$ contains 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` = (Δ^I, o^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 $(a^I, b^I) \in R^I$.

Conjunction assertions about an individual forms a description 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:

```
oag:AddressBase(adress12)
oag:hasId(address12, id23)
oag:hasCity(address12, city56)
oag:hasCountry(address12, country78)
oag:TokenValue(id34, "1111")
oag:languageId(city56, "en")
oag:StringValue(city56, "Belgrade")
oag:TokenValue(country78, "SCG")
```

4.6 Validating source data

This validation step checks transformation result with respect to both the concept definition and other semantic constraints, which may be defined in the corresponding ontology. 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, however, a document being exchanged must contain all required information and to compute that an instance has all mandatory properties it is necessary to validate instance with 'local Closed World Assumption' (CWA). The semantics of OWL currently provide the standard logical model of an OWA. To illustrate checking in OWA let us consider a set of assertions A defined as the following:

```
star:Address(address1)
oag:hasCity(address1, city56)
oag:hasCity(address1, city89)
```

We have an individual `address1` with an explicit assertion that it is an instance of the `star:Address`. This individual has two `hasCity` role fillers. 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 an explicit statement that two individuals are different or equal), but is very important when we perform individual checking. The interpretation function $\cdot^{I}$ is extended in such way that for any two individuals $a, b \in A$, $a \neq b$ if $a^{I} \neq b^{I}$. Using UNA, the 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. On the basis of the instance checking in OWA, one can conclude that this individual is consistent. However, when an individual is not complete, as is the case, we can still recognise concept membership. If we know that an individual `address2` is `star:Address`, adding information to the model cannot cause it to become false.

From the definition of the `star:Address` class, however, we can see that `hasCity` and `hasCountry` are mandatory properties of that class. In this case, we 'close the world' by creating a temporary class description (i.e. a query) based on the particular individual. We include 'close' operator, which takes an individual and a role and 'closes' the role on the individual, by first counting the known fillers for the role on the individual and then asserting number restriction on the individual (the class definition). We also assert the values on the present properties as value restrictions of that temporary class. This definition will be an Auxiliary Most-Specific (AMS) concept definition for a particular individual. An inference rule is needed to look at roles that are closed on individuals and check to see if all their fillers satisfy a value restriction.

$$\texttt{T\_address2} \equiv \texttt{star:Address} \sqcap (\leq 0\ \texttt{oag:hasCity}) \sqcap (\leq 0\ \texttt{oag:hasCountry})$$

To check whether *a* is an 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.

```
T_address2 ⊑ star:Address
```

As this subsumption does not hold, we can conclude that the individual `address2` is inconsistent with respect to the concept `star:Address` in closed world reasoning, that is, the individual `address2` is not a valid instance of `star:Address`.

4.7 Create satisfiable merged ontology

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). 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).

The actual process of ontology merging is the same discussed in the Testing Integration Capabilities portion of Section 4.4. (If this testing resulted in additional mapping axioms, then this entire additional axiom set will be included in the merged ontology.) It is also possible to create the merged ontology at design time. In that case, the merged ontology will be referenced and, for the performance reason, may be reduced only to include the sufficient set of concepts that is necessary for the next step of the data transformation. We check satisfiability for every concept of the merged ontology.

4.8 Check source data consistency for the merged ontology

As 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 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 about consistency checking, see Section 4.6).

4.9 Compute target data

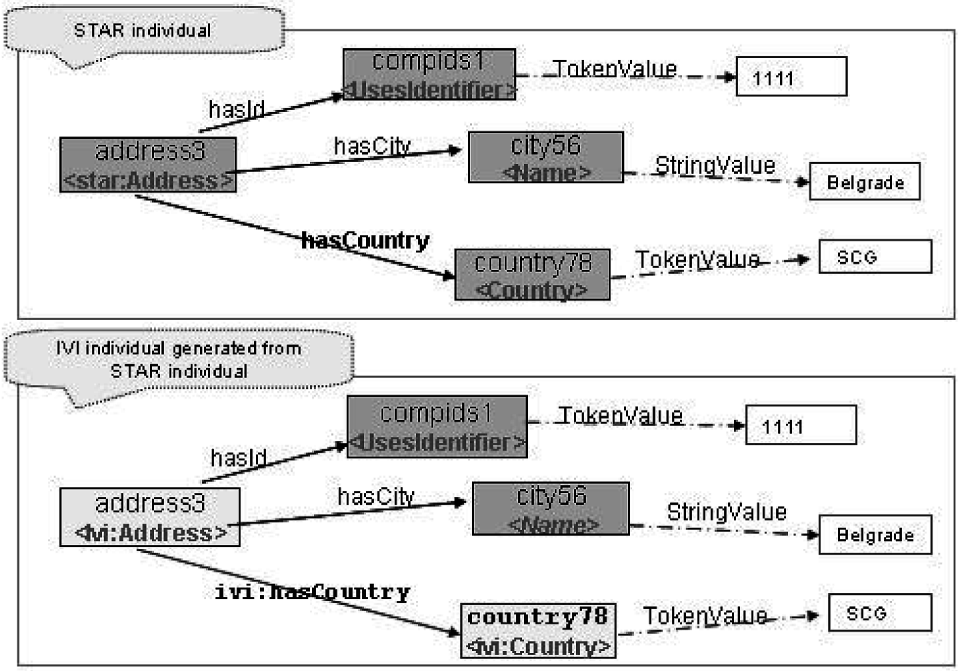
To compute target data, we use the merged ontology to calculate a new concept subsumption hierarchy. We use the individual classification capability of a DL reasoner to compute target data (i.e. individuals). The individual classification allows us to find the most-specific concept for every individual in the target ontology as well as other concepts that the particular individual belongs to.

Let us define a set of assertions A_{star} that describe an individual `address3`, an instance of the `star:Address` concept. The individual has property fillers for `oag:hasId`, `oag:hasCity` and `oag:hasCountry`:

```
star:Address(address3)
oag:hasId(address1, compids1)
oag:hasCountry(address1, country78)
oag:hasCity(address1, city56)
oag:TokenValue(compids1, "1111")
oag:StringValue(city56, "Belgrade")
oag:TokenValue(country78, "SCG")
```

For the `address3` individual, we define an AMS concept `Msc_address3` (we formally define operator $\in$: `address3` $\in$ `star:Address` $\sqcap$ `Msc_address3`). If the `address3` satisfies necessary and sufficient conditions for `star:Address` (and subsumption between `Msc_address3` and `star:Address` holds), then `address3` will be classified as a valid instance of `star:Address` (Figure 10).

Figure 10 Compute target data – individual classification



Using the merged ontology T', we check satisfiability between the AMS concepts and other concepts 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):

```
star:Address(address3) and  star:Address ≡ ivi:Address
⇒  ivi:Address(address3)
```

If an individual is consistent, then it can be classified as an instance of a new concept or a set of concepts in the merged ontology T'. Our target ontology T_{ivi} introduces a new property `ivi:hasCountry` that is equivalent to `oag:hasCountry`. As a range of that property, a new `ivi:Country` class is defined in the T_{ivi} ontology. As a consequence of these relationships between concepts and properties, the newly classified T_{ivi} instance has obtained a locally defined property `ivi:hasCountry` and the corresponding range (that has also obtained a locally defined class for its range value).

From an individual assertion, we may deduce information about other individuals using the notation of propagation. If it is known that `a` is an instance of `ivi:Address`, based on equality between roles `ivi:hasCountry` and `oag:hasCountry` and range restriction on `ivi:hasCountry`, one can deduce that `country78` is also an instance of `ivi:Country` class.

```
ivi:Address(address3)
  and  ivi:hasCountry(address3, country78)
  and  (∀ivi:hasCountry.ivi:Country)
⇒ ivi:Country(country78)
```

4.10 Validate target data

Similar to the discussion of validating a source individual, it is necessary to check that a target individual is a valid instance under both OWA and CWA assumptions. The individual consistency checking in OWA is already done with respect to the merged ontology. The OWL individuals classified in the AIAG concept hierarchy have to be checked for sufficiency with respect to the 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), is translatable into a target OWL data and allowing interoperable data exchange.

As discussed previously, finding a new individual consistent is still not sufficient for interoperability in B2B context. For instance, if it is calculated that `address3` ∈ `ivi:Address` and target ontology T_{ivi} have these additional axioms:

```
ivi:Address ⊑ ∃oag:hasLine.Text
ivi:Msc_address3 ⊑ ivi:Address ⊓ (≦0 oag:hasLine),
```

then the AMS concept `ivi:Msc_address3` will contain a constraint on `hasLine` equal to zero. This, however, is contradictory to the introduced axiom ‘at least 1 `hasLine` with range `Text`’.

An AMS concept in the target ontology may be different from the corresponding (i.e. the same individual) AMS concept in the source ontology.

4.11 Serialising target data

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

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

5 Related work

A previous effort investigated use of Semantic Web technologies (e.g. the DAML+OIL merger of the US DARPA Agent Markup Language and the Ontology Inference Layer (OIL)) in support of semantic constraints definitions and management for RosettaNet (Trastour et al., 2003). In this paper, an approach for mapping from XML Schema to DAML+OIL is 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 our focus on evaluation and validation of integration results in the EAI standards domain.

An initial exploration of OWL as a model-based language for integrating XML data sources was reported in Lehti and Fankhauser (2004) with OWL introduced as a top layer of heterogeneous XML data sources. The focus is on an OWL query language 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 of XML schemas through specification of concepts and semantic relationships among them (Boukottaya et al., 2004). The work introduces a transformation framework that encompasses the whole XML document transformation process, from modelling and semantic matching to transformation script generation. Conceptual modelling is used to automate the transformation algorithm. While this work 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 customised.

6 Conclusion

In this paper, we have described a Semantic Web-based integration architecture and methodology to serve as a blueprint to assess the Semantic Web technologies for EAI standards. This novel integration methodology is described through an integration scenario and validation steps that are performed both at design time and run time. During design time, the methodology supports development of generalised and normalised 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 results show interesting capabilities such as the ability to perform an individual equivalence test that is content based. Our immediate future work will focus on experimental assessment of the initial ideas for Semantic Web-based EAI standards. We expect to identify the key technical issues for the proposed approach and through experimentation show whether these issues be addressed using the proposed approach.

7 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

Anicic, N. and Ivezic, N. (2005) 'Semantic Web technologies for enterprise application integration', *ComSIS International Journal*, Vol. 2, No. 1, Available at: http://www.comsis.fon.bg.ac.yu/ComSIS/Vol2No1/RegularPapers/AnicicIvezic.htm.

Automotive Industry Action Group (AIAG) (2004) Available at: http://www.aiag.org/.

Boukottaya, A., Vanoirbeek, C., Paganelli, F. and Khaled, A.O. (2004) 'Automating XML document transformations: a conceptual modeling based approach', *The First Asia-Pacific Conference on Conceptual Modeling*, New Zealand: Dunedin.

Decker, S., et al. (2000) 'The Semantic Web – on the roles of XML and RDF', *IEEE Internet Computing*, Vol. 4, No. 5, pp.63–74.

Haarslev, V. and Moller, R. (2001) 'Description of the RACER system and its applications', *Proceedings of the International Workshop on Description Logics.*

Horrocks, I., Patel-Schneider, P.F. and Harmelen, F. (2003) 'From SHIQ and RDF to OWL: the making of a web ontology language', *Journal of Web Semantics*, Vol. 1, pp.7–26.

Jelliffe, R. (2004) *Schematron – Pattern-Based Schema Language*, Available at: http://www.ascc.net/xml/resource/schematron/schematron.html.

Lehti, P. and Fankhauser, P. (2004) 'XML data integration with OWL: experiences and challenges', *Applications and the Internet*, *Proceedings 2004 Internat*, *Symposium*, pp.160–167.

McGuinness, D.L. and Harmelen, F. (Eds) (2004) *OWL Web Ontology Language Overview*, Available at: http://www.w3c.org/TR/owl-features/.

Meadows, B. and Seaburg, L. (Eds). (2004) *Universal Business Language 1.0*, Available at: http://docs.oasis-open.org/ubl/cd-UBL-1.0/.

Nardi, D. and Brachman, R.J. (2003) 'An introduction to description logics', in F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider (Eds). *Description Logic Handbook*, Cambridge University Press, pp.1–39.

Open Applications Group (OAG) (2004) Available at: http://www.openapplications.org/.

RosettaNet (2004) Available at: http://www.rosettanet.org.

Schmidt-Schauß, M. and Smolka, G. (1991) 'Attributive concept descriptions with complements', *Artificial Intelligance*, Vol. 48, No. 1, pp.1–26.

Standards for Technology in Automotive Retail (STAR) (2004) Available at: http://www.starstandard.org/.

Trastour, D., Preist, C. and Coleman, D. (2003) 'Using Semantic Web technology to enhance current business-to-business integration approaches', *Seventh IEEE International Enterprise Distributed Object Computing Conference*, *EDOC 2003*, Brisbane, Australia.

Note

[1]A specification may be either description or definition.