15th ICCRTS

“The Evolution of C2”

A Semantics-Based Approach to Schema Matching and Transformation in Network Centric Environments

Topic:
Networks and Networking

Authors:

Anton DeFrancesco
Securboration Inc.
1050 W. NASA Blvd.
Suite 157
Melbourne FL, 32901
(321) 409-5252 x21
adefrancesco@securboration.com

Bruce McQueary
Securboration Inc.
1050 W. NASA Blvd.
Suite 155
Melbourne FL, 32901
(321) 409-5252 x16
bmcqueary@securboration.com

1. Abstract

DoD and Air Force direction is clear regarding acquisition of new systems and direction for legacy systems: service oriented implementation and net-centric sharing of data are no longer optional. A key challenge with this direction, however, is the ability to align services in terms of their capabilities, inputs, outputs, and other constraints – what is referred to as *schema matching and transformation*. Without this capability key service-based initiatives such as Global Command and Control System - Integrated Imagery and Intelligence (GCCS-I3)[1], which is a standard set of tools and services to accessing imagery and intelligence, will not be realized to their maximum potential. The schema matching and transformation needed to interoperate with such services has been a manual, tedious, and error prone task. While there has been some automation, the majority is based on syntactical comparisons. This paper illustrates how semantic modeling concepts can be applied to facilitate schema matching and transformation. Our approach leverages an ontology-based model that encodes semantics of the domain and determines how well services schema match the domain. Consequently schema is compared based on semantics, or the underlying intent of information exchange within the intended domain, versus a syntax-based comparison.

2. Introduction

The word ‘schema’ is derived from a Greek word that means ‘shape’ or more generally ‘plan’[2]. With respect to the net-centric environment, schema relates to a systems ‘plan’ for information exchange and includes specification of capabilities, requirements, and data structures, along with other relevant details. Schemas (or schemata) can take various forms depending on the underlying systems. For example, Web Services Description Language[3] (WSDL) is used to specify schema that describe the set of network services a system provides, and XML Schema[4] definitions (commonly referred to as XSD) describe the related network data structures used by the services (i.e. input and output parameters and data types). Additionally, schema can take the form of meta-data tagging approaches, such as DoD Discovery Metadata Specification[5] (DDMS), or it may be based on other custom structures specified using custom XML tags.

The vision for schemata within net-centric operations is for them to support interoperability of independently developed systems. In theory integration engineers should be able to browse schemata from existing repositories and with minimal effort leverage services from other systems and components to determine their compatibility. For example this might include determining how services can be composed together such that the output of one service feeds the input to another service within the context of an operational process. The schemata can be compared to identify applicable network services and the data structures they consume/produce.

[1] Defense Update, Issue 2, 2006. GCCS-I3 Global Command and Control System – Integrated Imagery and Intelligence. http://defense-update.com/products/g/GCCS-I3.htm
[2] http://en.wikipedia.org/wiki/Schema
[3] Christensen, Curbera, Meredith, and Weerawarana, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl
[4] Fallside and Walmsley, XML Schema Part 0: Primer Second Edition, http://www.w3.org/TR/xmlschema-0/
[5] Department of Defense Discovery Metadata Specification Home Page. http://metadata.dod.mil/mdr/irs/DDMS/

However, in practice schemata do not perfectly match, and aligning them becomes a highly manual and tedious process. It is more typical that schemata may match only to a certain degree and transformations on data elements are required before services can be composed. Thus the schema alignment process requires iterating through matching schema elements and specifying transformations for those elements that do not match.

The matching process is a crucial step towards mitigating the complexity of data transformation. It takes two schemas and returns a mapping that identifies the correspondences. Much of the matching to date is done manually by domain experts that can be easily led astray by minor name and structure variations. Automated and semi-automated applications to support schema matching are generally based on analyzing syntactic information.

Syntactic information deals with the form in which information is assembled and conveyed. In schema matching, examples include specification of a piece of data as numeric or text or three characters long, and may include lists of allowable character strings as values to a data field. A domain user may recognize that such information has implications for meaning (for example, that within a data field labeled "location," a piece of data is three characters long and may tell an expert user which of several possible meanings of location applies). That implication, however, is indirect and depends on user's intimate familiarity with the context and the possibilities. Another user, from a different domain and/or with less familiarity, would be unable to draw the same conclusion. The specification that the piece of data is three characters long does not tell the user what that data means, only its form. This is the essence of syntactic information and it has a role in schema matching, but primarily as a 'first-pass' to deeper analysis that considers the schemata meaning within their domain(s).

This deeper analysis is possible because it focuses on semantic information, or the meaning of the schemata. Examples of semantic information include: documentation or meta data that the location information in an XML document or database refers to stable codes assigned by the Air Force and can be used for shipping; or that a schema element labeled "equipment" in fact refers to the equipment held by a particular organization and includes only equipment that is authorized, on-hand, and deployable; or specification that a particular piece of data labeled "funds" provides information about funds available in the current fiscal year. Each of these is primarily semantic in nature. Each captures some aspect of the meaning of the data, including how it relates to other information. Each makes explicit some piece of the context, the intentions and limitations, the relationships that enable users to understand that data correctly, to extract the information the data is supposed to represent. Each represents some piece of knowledge, beyond syntax or form, that is essential to correct interpretation of the meaning of the data, and so to the effective use of the information. This is the essence of semantic information[6].

3. A Semantic Based System

The key to disparate system integration is domain knowledge to enable an understanding of each integrating part. It is this cognition, formulated by axioms - an established rule or principle or a self-evident truth[7] - that allows system integrators the ability to identify entities and relations by semantics. By utilizing this domain knowledge, system integrators are able to distinguish

[6] Community of Interest (COI) Primer, United States Air Force, Version 3.0

[7] Merriam-Webster's Online Dictionary. http://merriam-webster.com/dictionary/AXIOM

between entities that have similar names but different meaning. (ie. A tanker has different meaning for the Army and Air Force.)

Applying these domain concepts in the form of axioms to an automated system involves many facets of Natural Language Processing (NLP), Graph Theory, and the use of ontology. Each of these technologies provides a role in the overall processing including language interpretation, modeling, and reasoning. Together, they augment each other allowing a system to reach a new level of semantic meaning. This notion of semantic matching is currently being implemented in the Semantic Matching and Transformation System (SMTS), which is a system for the parsing and interpretation of service level schemata and the application of that information towards service identification, matching, and transformation. Its purpose is to alleviate the arduous task of sifting through large, complex schemata looking for entities that are semantically similar. The SMTS ingests these schemata, parses through them, and then correlates the information against a semantic model. The data is then matched up based on the level of correlations between the entities and presented to the user. The challenges faced by system integrators for schema alignment in an operational environment are illustrated in Figure 1. Too often it is assumed that producers and consumers in a net centric environment can automatically exchange information. As depicted in Figure 1, this is typically not the case. Although services may be conceptually compatible, in order to actually exchange information, their schema must be aligned to allow semantically similar entities to be either directly reused or transformed into syntactically compatible format. An example of two semantically similar but syntactically different schemas is demonstrated in Table 1 and Table 2.

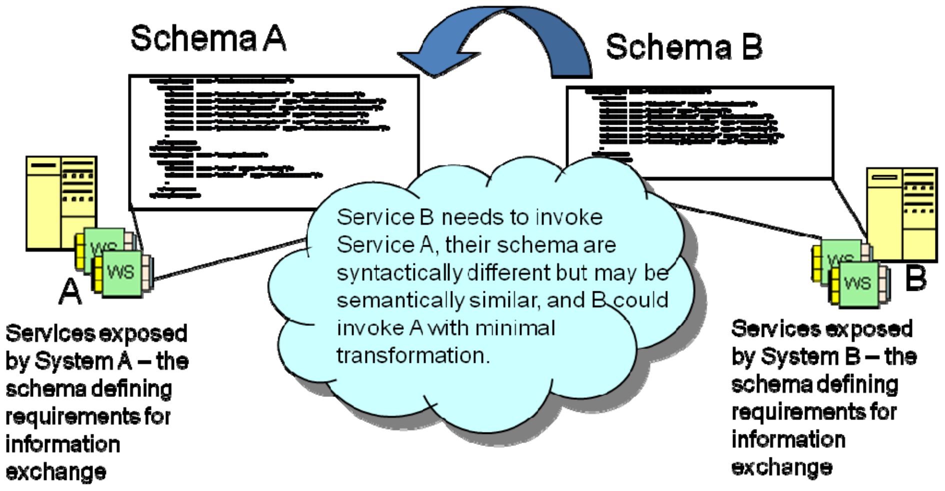

Schema A
Schema B
Service B needs to invoke Service A, their schema are syntactically different but may be semantically similar, and B could invoke A with minimal transformation.
WS
WS
A
B
Services exposed by System A – the schema defining requirements for information exchange
Services exposed by System B – the schema defining requirements for information exchange

Figure 1 - Problem Scenario

The following two schemata will be used throughout the document as an example of how the semantic matching engine breaks down XML Schema into component parts and analyzes the information. The first example located in Table 1, is for a PatientRelocation record which contains the primary information for patient transportation and is based off of a "Patient Movement Record" used within transfer applications and services at USTRANSCOM.

Table 1 - Example Patient Relocation Schema Definition

```
<complexType name="PatientRelocation">
  <sequence>
    <element name="identifier" type="RelocationID"/>
    <element name=”patient” type=”Patient”/>
    <element name=”patient-status” type=”PatientStatus”/>
    <element name="origination-facility" type="Facility"/>
    <element name="destination-facility" type="Facility"/>
    <element name="authorizing-physician" type="Physician"/>
    <element name="receiving-physician" type="Physician"/>
    …
  </sequence>
</complexType>
```

The second schema, given in Table 2 is for PatientTransferData. It defines the basic information required to initiate a patient transfer from one hospital to another.

Table 2 - Example Patient Transfer Data

```
<complexType name="PatientTransferData">
  <sequence>
    <element name="transferringPatient" type="PatientData"/>
    <element name="initiatingDoctor" type="MedicalPersonnelData"/>
    <element name="receivingDoctor" type="MedicalPersonnelData"/>
    <element name="originatingHospital" type="HospitalData"/>
    <element name="destinationHospital" type="HospitalData"/>
    <element name=”patientCondition” type=”PatientCondidtionData”/>
    …
  </sequence>
</complexType>
<complexType name=”HospitalData”>
  <sequence>
    <element name=”name” type=”string”/>
    <element name=”address” type=”AddressData”/>
    …
  </sequence>
</complexType>
```

For the sake of simplicity, the schema have had their namespaces stripped out and only contain a subset of the information within the actual schemata.

3.1. Semantic Analysis

The use of a vanilla syntactic or statistical parser with the above definitions would yield little in results. No information is seemingly shared between the schemas; there are virtually no keywords or syntactic commonality that may be used as a basis. This is not to say that syntactic or statistical parsing is not beneficial, but alone they cannot provide the power and benefits required.

Consider Table 1 above; it contains elements that attempt to use self-described names. The element *destination-facility* is an example of an entity that provides a wealth of semantic information. The use of the word *destination* has implications of some object or entity traveling to a goal. *Facility* is a bit vaguer and implies some building or vehicle used for a purpose. This allows the reader to assess that some entity is traveling to another location which may either be a building or vehicle. To a human reader, this assessment can be done quickly and semi-accurately given reasonable domain knowledge. The difficult task is to impart that assessment ability to a machine, which lacks the foundation of knowledge that humans build up over time.

To address this inadequacy, an iterative approach is taken that analyzes individual words and their relations. It does this by examining the data and breaking down the information, decomposing it into its component parts. It then attempts to identify the data based on some known body of knowledge, or Domain Model, and establishing relationships the data may have with surrounding elements. This controlling algorithm is a cyclic one that requires the system to continuously refine itself to reach a conclusive state, as shown by Figure 2.

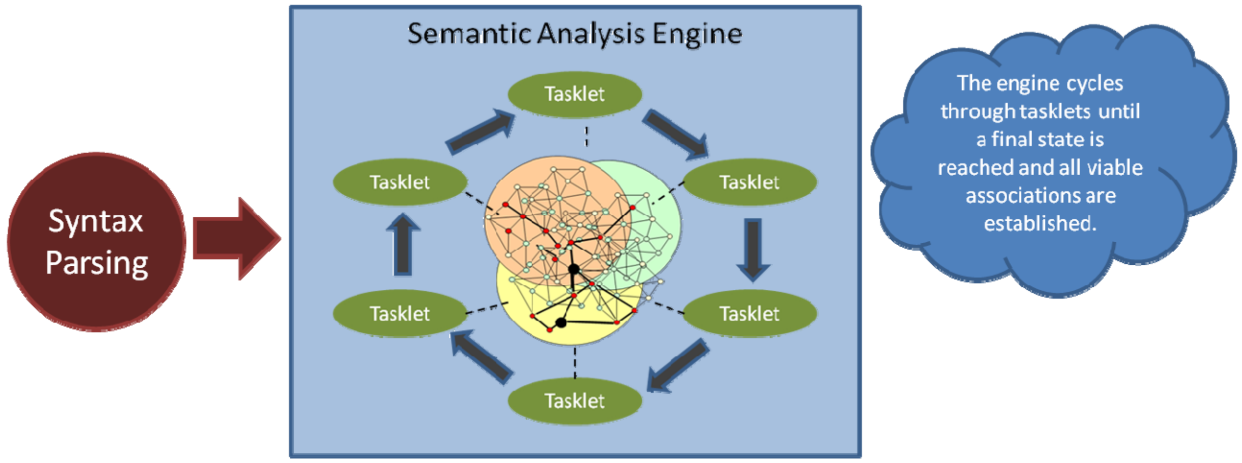
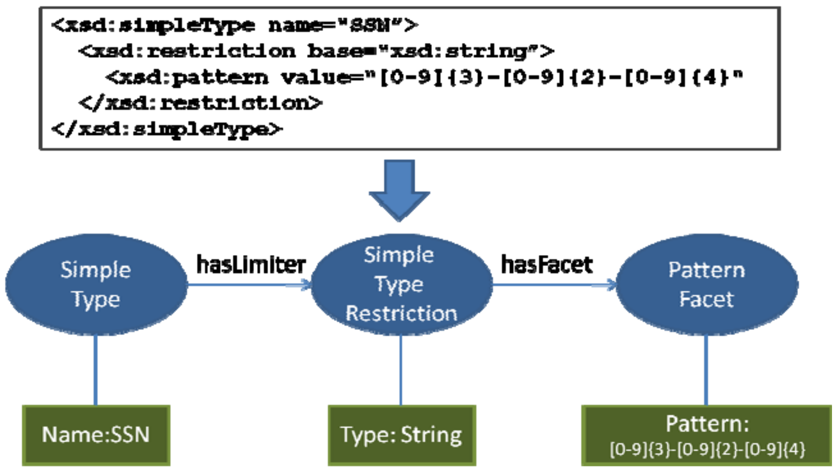

Syntax Parsing
Semantic Analysis Engine
Tasklet
Tasklet
Tasklet
Tasklet
Tasklet
Tasklet
The engine cycles through tasklets until a final state is reached and all viable associations are established.

Figure 2 - Word Analysis Breakdown

3.1.1. Syntax Parsing

The cycle begins with the syntactic parsing of the schema data where information is extracted and placed within a directed graph adhering to the explicit hierarchy defined by the schema. The syntactic parsing is used to elicit information based on the schema structure and does not require the complexities of a full Natural Language Parser (NLP) engine. The extracted knowledge is retained within a complex structure that maintains all schematic relationships and retains all schema information such as type, limiters and facets. Limiters can be restrictions on a schematic type and a facet is a further constraint placed on the limiter. An example of the level of knowledge extraction is demonstrated in Figure 3.



```
<xsd:simpleType name="SSN">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9]{3}-[0-9]{2}-[0-9]{4}"
  </xsd:restriction>
</xsd:simpleType>
```

Figure 3 - Example XML Schema to dataset element

As demonstrated from the example, the system extracts out the schema type named "SSN" and maintains an association between it and the restriction that was placed upon the type. The restriction itself also maintains an association with the facet that places further limitations on the restriction, thus defining that a SSN must have a format of XXX-XX-XXXX where X is a number between zero and nine.

3.1.2. Semantic Analysis Engine

The schematic information, having been broken down into a dataset suitable for processing, is then passed to the Semantic Analysis component where it undergoes a cycle of processing where the datasets are further processed by correlating identifications and associations. At the end of each cycle, an algorithm measures of the scope of changes that were made in the last iteration and determines whether the system requires additional cycles. If the system does require an additional pass, the cycle begins anew. The schematic data is processed by small, independent algorithms called tasklets. Each tasklet reads in the schematic data, executes its algorithm on the data, updates the state of the data (if required), and returns a value indicating the scope of changes that occurred within the algorithm.

When the SMTS is evaluating a specific schema entity or domain concept, it has what is referred to as a current target. The current target is a value within the schema entity or domain concept that is undergoing evaluation as the SMTS evaluates that particular entry. An example would be *destinationHospital* , the Semantic Analysis Engine would first set the current target to the type associated with *destinationHospital*, *Hospital*. During the next cycle, the system might choose the name *destinationHospital* as the target. Different iterations can use a variation of the name or type, including the use of synonyms, hypernyms (discussed below), and partials of the words, such as *destination* and *hospital*. A hypernym, as defined by WordNet, is a word that is more generic than a given word.[8] For example, a point is a hypernym of geographic point.

The cycle begins with the identification tasklet that attempts to directly match the current target within the schematic entity with an associated semantic concept within the Domain Model. If a match is found then a link is established between the schemata structure and the Domain Model

[8] Miller, G., Princeton University, http://wordnet.princeton.edu/

concepts. This link is assigned variable strength, depending on the iteration. If no matches are found for a given schematic node, it is passed to the next tasklet in the current cycle. As the cycle continues through multiple passes, the identification tasklet will weaken the strength of the link the schematic structure may form with the Domain Model concepts. This is due to the fact that each cycle will cause the system to evaluate a more generic instance of the schematic information thus rendering the data less valuable.

One of the more powerful aspects of semantics processing is the ability to relate data and reason across those relationships. The second tasklet analyzes the schematic information and the Domain Model looking for matching or similar associations.

At this point, it is advantageous to discuss the Domain Model. The Domain Model is an ontological model that is represented in the machine readable Web Ontology Language (OWL[9]). Domain axioms are encoded into the model by using ontology constructs such as entities and their relationships. Axioms capture information surrounding the domain and detail the artifacts, processes, and relationships needed to communicate the concepts for the domain space. Figure 4 demonstrates an example of a Domain Model and some of the relationships that exist. The model depicted is a subset of the actual Domain Model for the SMTS example. The SMTS requires the use of a more directed Domain Model. The domain space is required to be limited in size and scope to focus the semantic analysis towards a specific domain.

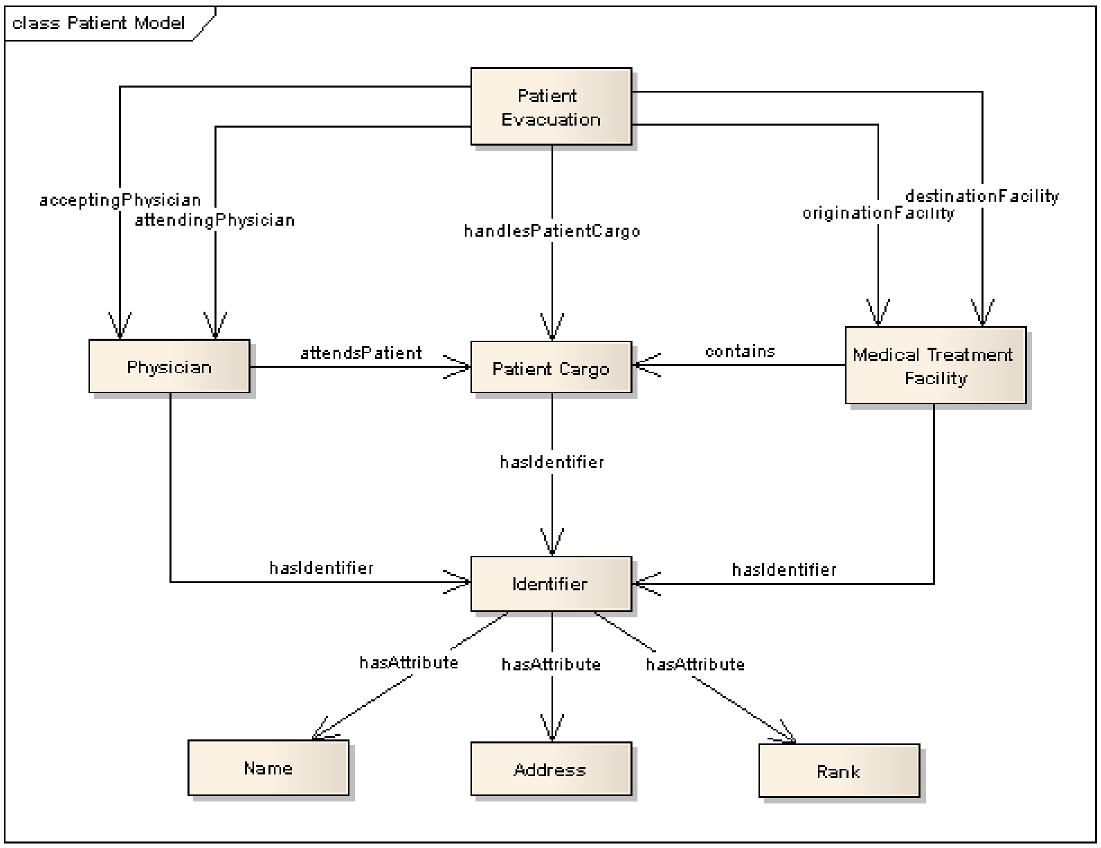


Figure 4 - Graphical representation for a subset for the Patient Movement Requirement Domain Model

[9] World Wide Web Consortium (W3C). Web Ontology Language(OWL), http://www.w3.org/TR/owl-ref/

The figure above describes a set of relationships between the domain entities. An example is Patient Evacuation has two links to Physician because during an evacuation, there would be an attending physician at the site where the patient was picked up, and a physician accepting the patient at the new site. Both of these concepts are modeled.

The OWL reasoning component was chosen because of its performance, expressivity, and support for defining metadata beyond the current definitions and implementations. Non-ontological metadata implementations provide no formal mechanisms to capture the relationships, and interactions of the objects they are attempting to describe throughout the domain space. OWL provides formal mechanisms to capture all forms of 'traditional' metadata and extends that capability through the use of ontological properties to capture complex domain relationships. Enabling the user to define relationships in the form of OWL properties, combined with OWL defined property characteristics, provides a powerful mechanism for enhanced reasoning about a property[10].

For example, consider the OWL transitive property characteristic that states for each property if P is a property of X and Y; and P is a property of Y and Z; then P is a property of X and Z:

$$\forall\chi\forall\gamma\forall\alpha(P(\chi,\gamma)\wedge P(\gamma,\alpha)\rightarrow P(\chi,\alpha))$$

An example of the rule above can apply to different scenarios, an example would be has Ancestor. If X has an ancestor Y and Y has an ancestor Z, then by inference we know that X has an ancestor Z.
The Semantic Web Rule Language[11] (SWRL) is a rule set that is integrated with OWL that enables more complex expressions and capabilities than base OWL constructs. The following is an example system rule that is implemented in SWRL; the rule pertains to Figure 4 above.

$$\forall\chi\forall\gamma\forall\alpha(\mathrm{hasIdentifier}(\chi,\gamma)\wedge\mathrm{hasAttribute}(\gamma,\alpha)\rightarrow\mathrm{hasAttribute}(\chi,\alpha))$$

The statement says that if X has an identifier Y and Y has an attribute A, then X has an attribute A. This type of statement can become important because individuals creating schema tend to not to create flat structures. For example, one schema creator may not put the string "name" under a person, but instead have a structure that contains a person's first name, last name, middle initial, prefix and suffix. This type of structure is common in schematic development and SWRL helps alleviate such issues prevalent in this type of schematic hierarchy.

The ontological reasoning tasklet begins by translating the schematic graph into a form where it is placed within a temporary instance of the Domain Model. The current target for a schematic entity is used to determine placement of entities within the Domain Model. Once placement is finished, the inference engine is run across the Domain Model to see if any additional information can be obtained. The resulting inferences are then analyzed to determine completeness of domain concepts. If a schematic entity matches a predefined percentage of associations with a Domain Model concept, then the relation between the schematic entity and the Domain Model concept is strengthened. If the percentage of associations is small but greater

[10] Herman, Ivan, 2007, Web Ontology Language(OWL), http://www.w3.org/2004/OWL/
[11] Horrocks, Patel-Schneidr, Boley, Tabet, Grosof, and Dean, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, Semantic Web Rule Language (SWRL), http://www.w3.org/Submission/SWRL/

than zero, the system will then mark the associations and attempt to find synonyms or hypernyms that may meet the criteria when the identification phase is run again - the relationship between the schematic entity and the Domain Model concept is unchanged. If no associations are present, the system will weaken the relationship between the schematic entity and the Domain Model concept.

Another type of tasklet is searches the lexical database, WordNet[12], for variations of the original entity name to place as the current target. This allows the system to make associations on more general instances of the entity name and potentially expand the number associations. This is done by using the hypernym capability of WordNet to lookup root meaning of words to try to find a more generic term.

The example given in Figure 5 demonstrates a portion of the hypernym hierarchy provided by WordNet. A hypernym tree provides more specific definitions the further out on the tree the word sits, the closer the word is to the base, the more general it is. For the purposes of comparing entities, once a certain threshold is reached within the hypernym tree, the definitions become too general to provide meaningful data to the system so they are bypassed.

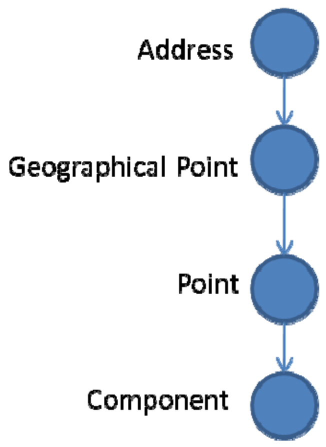


Figure 5 - Example hypernym tree for address

The drawback to using WordNet and hypernyms is that it is easy to become too general in the hypernym hierarchy and create a lot of low-value links between the schematic data and the Domain Model.

Another tasklet provides stemming on the current target and attempts to find additional forms that may have meaning within the system. This is done by using stemming to find the root of the entity. *Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form*[13]. This process allows the system to take into account plurality of entities and name derivatives that may not exist in the lexical database. An example is *connected* which has the root *connect*.

[12] Miller, G., Princeton University, http://wordnet.princeton.edu/
[13] http://en.wikipedia.org/wiki/Stemming

Capabilities are an entities inherent trait or ability. They define what an concept is, what it is able to accomplish, and what its characteristics may be. An example is a *Physician* having the capabilities DiagnosticCapability and TherapeuticCapability. These two capabilities define two abilities that a physician is capable of. Other concepts may also share the same capabilities, thus demonstrating a similarity between them. Each concept defined within the Domain Model has one or more capabilities assigned to it. The capability tasklet examines the associations and once a medium strength or higher association is made between the Domain Model and a schema model, the capabilities associated with that concept in the Domain Model are propagated to the schema entity.

The last tasklet of note is the translation tasklet. When WSDL and schema are read into the system and processed, all service methods are retained in the Semantic Repository. When a service method is encountered that contains the words “translate” or “transform” plus it takes in one entity and returns a different entity, the system will mark that method as a translation method. During the execution of this tasklet the system will take into consideration the translation methods and make associations based on the systems knowledge that a translation or transformation is possible. Users have the ability to override the marking of a translation method in cases where the method does not actually provide that capability.

The scope of changes that occurred within the system is recorded at the end of each cycle. If the amount of change is less than a system specified designator, the iteration through the phases will end. The SMTS then prunes associations based on the degree of strength between the schematic structures and concepts within the Domain Model. Depending on the focus of the Domain Model and generalities used within the schemas, the system can generate a large number of weak relations that may need removal.

At this point, the SMTS has finished semantic analysis upon the schema and created a set of relationships between the schemata and the Domain Model. It has also recorded that information in the Semantic Repository which is a database of schemata and its relations with the Domain Model.

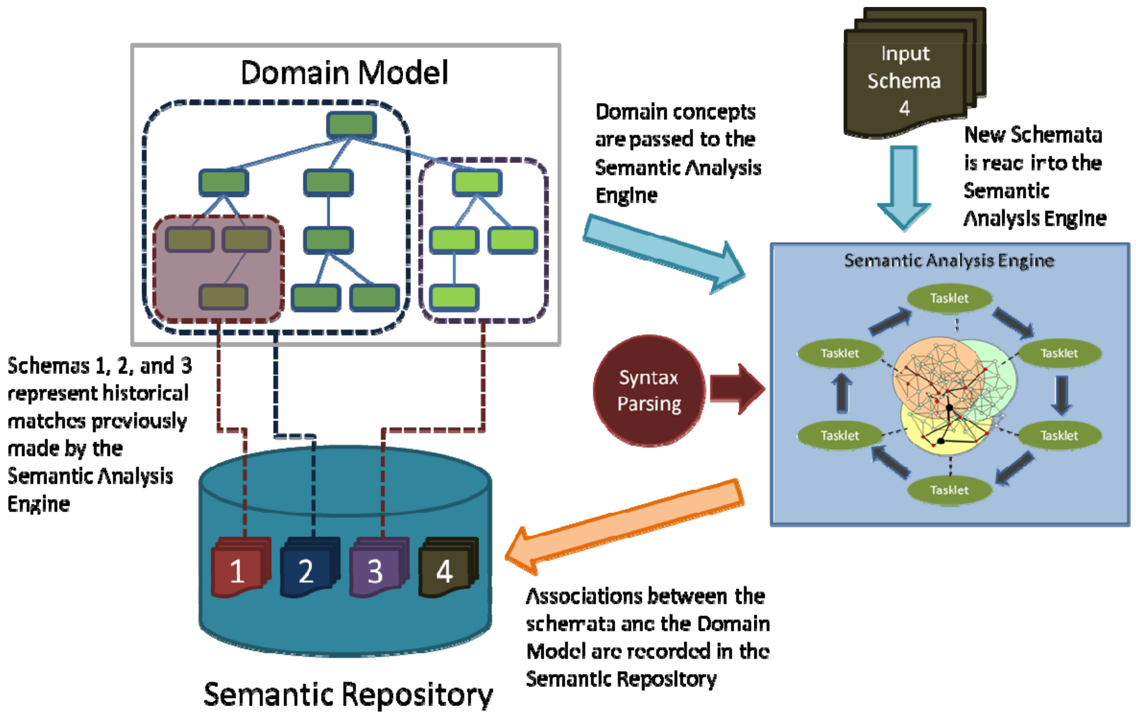

Domain Model
Input Schema 4
Domain concepts are passed to the Semantic Analysis Engine
New Schemata is reac irto the Semantic Analysis Engine
Semantic Analysis Engine
Tasklet
Syntax Parsing
Schemas 1, 2, and 3 represent historical matches previously made by the Semantic Analysis Engine
1
2
3
4
Associations between the schemata anc the Domain Model are recorded in the Semantic Repository
Semantic Repository

Figure 6 - Process of inputting the Domain Model and XML Schema to find associations

3.2. Schema Matching

After a schema is ingested by the Semantic Analysis Engine, its metadata is available for comparison against other schemata. The primary metadata used for the comparison are semantic concepts are the capabilities. During the capability tasklet, capabilities are assigned to the schematic entities. Once processing has finished, the SMTS performs one last comparison of the schema and the Domain Model. This time only capabilities are compared which enables the system to associate elements that contain semantically similar characteristics. This method produces results that will utilize the original elements plus any additional entities that hold the same capabilities.

The list of capabilities for a given schema entity is then matched against capabilities from other schema entities. The SMTS then uses a configurable rule set to determine if the two entities are semantically similar. It is important to note that capability matches imply semantic similarity within the schema entities; however it does not imply compatibility. Because the Domain Model is specific to a particular domain space, it is possible that two schema entities that match in one domain may not match in another.

3.3. Transformation

The transformation component of SMTS builds upon the metadata obtained through the Semantic Analysis Engine by matching up schematic structure with domain concepts and providing suggestions into how they might be associated. This is done by using the capabilities provided through the Schema Matching and providing associations between entities based on those capabilities. If an individual requests information pertaining to a specific service parameter, the SMTS will return a list of potential services that use semantically similar structures ordered by the number of capabilities and their strength. It is possible for a transformation to have more than one step, but that is rare given that each transformation may cause loss of data. Each additional transformation past the first one lowers the overall strength of a particular candidate.

3.4. Architecture

The architecture of SMTS provides services to all the primary components discussed in the document plus the presentation services and Ingestor services. The presentation services provide the user interface and the Ingestor Service provides parsing and protocol support for WSDL, Schema, OWL, and Universal Description Discovery and Integration (UDDI). These components plus the Syntactic Parser, Semantic Analysis Engine, and the Matching and Transformation Service are all represented by Web Service.

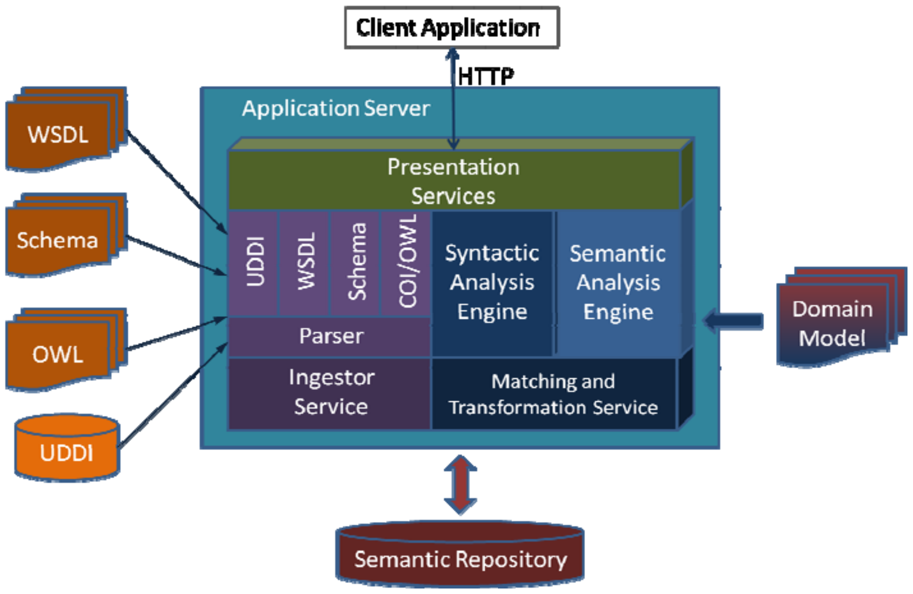


Figure 7 - SMTS Architecture

3.5. Example

DOD Directive number 6010.22 dated January 21, 2003 addresses the subject of National Disaster Medical System (NDMS). USTRANSCOM plays a role in the joint Federal, State and local aid organization for a coordinated medical response, patient movement and definitive inpatient care in time of war, U.S national emergency or major U.S. domestic disaster. They serve as the exclusive heavy lift provider for military cargo. One such “cargo” is wounded soldiers.

The example offered here is one associated with the submission of the XML Document that represents a “Patient Movement (PM) Record” (AF IMT 3899) and seeking out corresponding semantically grounded XSD, calculating the likelihood of semantic matches, and applying transformation, and laying the foundation for further automated workflow support. The example XSD that is being compared is from the integration of a hospital patient transfer.

The Patient Movement Requirement (PMR) schemata has been scaled down given for the purpose of the document. The two schema segments that will be analyzed during the example are found in

Table 1 - Example Patient Relocation Schema Definition

```
<complexType name="PatientRelocation">
  <sequence>
    <element name="identifier" type="RelocationID"/>
    <element name=”patient” type=”Patient”/>
    <element name=”patient-status” type=”PatientStatus”/>
    <element name="origination-facility" type="Facility"/>
    <element name="destination-facility" type="Facility"/>
    <element name="authorizing-physician" type="Physician"/>
    <element name="receiving-physician" type="Physician"/>
    …
  </sequence>
</complexType>
```

and Table 1 above. The Domain Model is located in Figure 4.

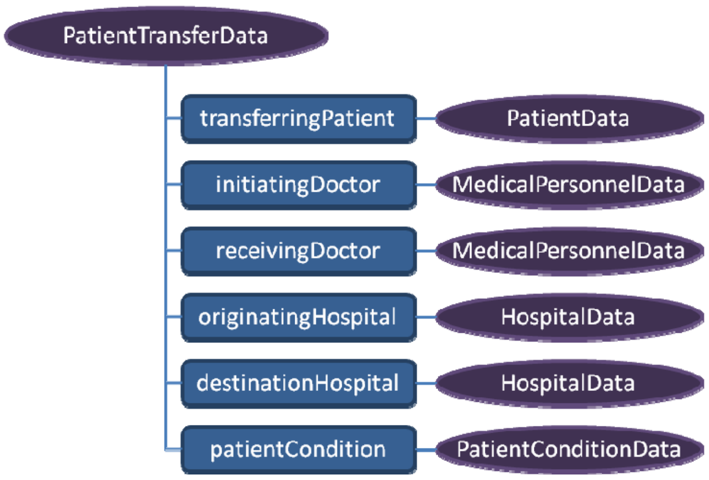


Figure 8 - Graphical Representation of Table 2

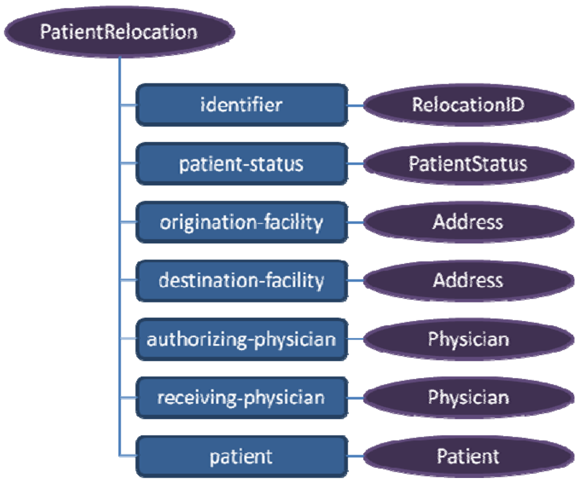


Figure 9 - Graphical Representation of Table 1

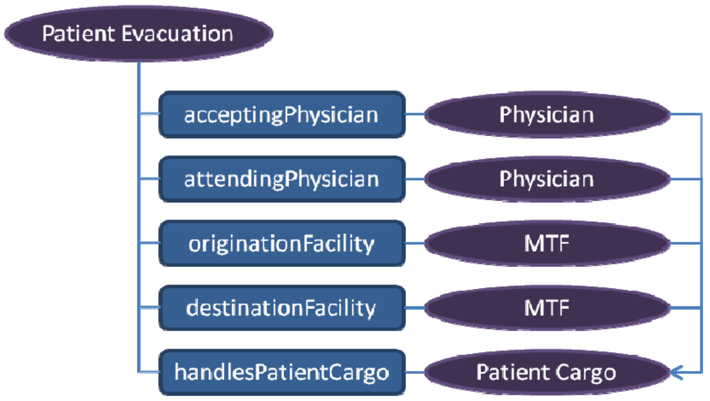


Figure 10 - Graphical Representation of Domain Model

The system begins by extracting the data from the XML Schema and placing the schema entities into a directed graph starting at the root XML node. One at a time, the graphs are fed through the Semantic Analysis Engine to retrieve their results. The following graphs will demonstrate the relations that are formed between the schemas and the Domain Model.

Figure 11 below illustrates the associations created between the schemas and the Domain Model during the first iteration. The primary associations created during the first iteration were based on partial name matches, which is a fairly weak association. The SMTS also made matches by using synonyms of the current target.

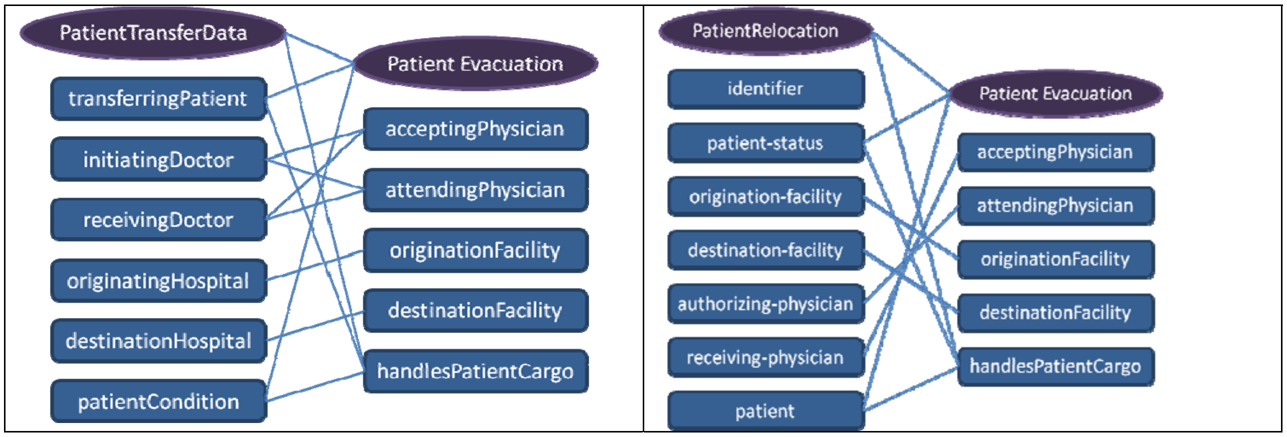


Figure 11 - Matching after 1st iteration

Subsequent iterations will cause the system to iterate through the type names for synonym and hypernym checking. Entities that have associations will be placed into the Domain Model for comparison and reasoning. These additional checks are using to strengthen current associations and create new ones.

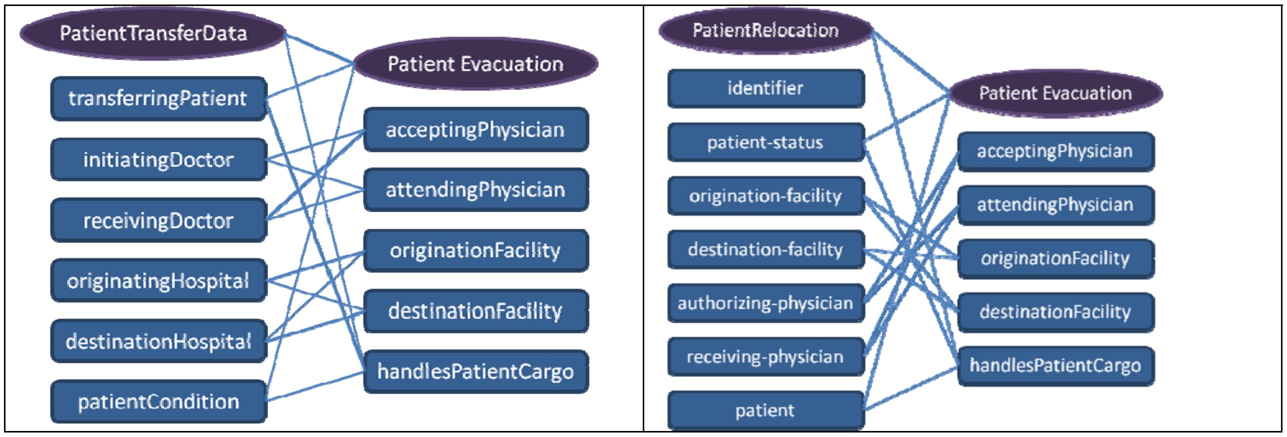


Figure 12 - Matching after final iteration

Figure 12 above shows the final state of the two comparisons. In both cases, the schematic entities did compare with the Domain Model concept *Patient Evacuation.* This means that all capabilities that were found within the Patient Evacuation were propagated to *PatientRelocation* and *PatientTransferData*.

When the SMTS finally compares the PatientTransferData and the PatientRelocation, it will see that they share the same core capabilities and will mark them as semantically similar. This in turn will highlight them as similar and if there is a translation path from the one entity to the other, the system will show that path to the user.

4. Results

The overall results for semantics based schema matching have been promising. The initial research focused on the matching or aligning schema associated with the Patient Movement Requirement (PMR). The results provided about a 60% matching of relevant schematic data which means a quantitative value was derived based on the cycles taken for the semantic analysis process to reach steady state compared against a manual investigation of the results. The matching of schematic entities was conducted using sample schemas based on USTRANSCOM Patient Movement Record services. The Domain Model was created based on USTRANSCOMS service and data models. Both primary schemata contained data sets of around one hundred data types and ten services. The Domain Model contained seventy-four domain entities. The processing time for this size data was negligible. Due to the scope of the initial research, all verification of data was done manually.

Additional testing was conducted with the use of 3rd party websites such as Amazon, Google, and Yahoo. Each of these entities provides web service interfaces for external developers to integrate into their systems. The schema matching process was able to parse through these services and generate correlations. Due to the lack of a solid Domain Model, the correlation data was limited, but the system was capable of scaling to handle service data sets that contained approximately five megabytes of data.

5. Conclusions

The manual application of schema matching and transformation is an extremely complex and difficult task that is also costly and time consuming. As more systems migrate from stove-pipe architectures to net-centric ones, the number of schemata that describes these systems will proliferate. This places an enormous burden on integration teams as the number of permutations explode and level of effort required for additional integration grows.

The solution to this is an automated method for schema matching and transformation. The Semantic Matching and Transformation Service provide the basis for alleviating much of the burden of system integration by identifying similar information within the schemas, as they pertain to the integrating domain. The schemata is parsed and analyzed by a cycling series of algorithms and ontological constructs designed to elicit and infer information from the schemata based on names, types, and structure. It is through this analysis that integrators will benefit by having large sets of domain specific information identified and potential transformation paths laid out for them.

Moving forward, the research direction for the STMS will focus on refining and tuning the tasklets to greater accuracy and slowly broadening the size and scope of the domain models. The STMS will also require additional tools for result verification and automated testing.

6. References

AFSPC Community of Interest (COI) Playbook, Version 1.4 April 2009

Design Patterns – Elements of Reusable Object-Oriented Software" (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) Addison-Wesley Publishing Company, 1995

XML Schema, World Wide Web Consortium (W3C), http://www.w3.org/XML/Schema
DOD-CIO, 2003, Department of Defense Net-Centric Data Strategy, http://cio-nii.defense.gov/docs/Net-Centric-Data-Strategy-2003-05-092.pdf

McQueary and DeFrancesco, 2009, Semantic Ontology-Assisted Matching System (SOAMS) – Air Force Research Labs Phase I Final Technical Report. Contract No. FA8750-09-C-0073