A Framework for Developing Manufacturing Service Capability Information Model

Yunsu Lee and Yun Peng

Department of Computer Science and Electrical Engineering,
University of Maryland, Baltimore County,
1000 Hilltop Circle, Baltimore, MD 21250
{yunsu.lee,ypeng}@umbc.edu

Abstract. Rapid formation and optimization of manufacturing production networks (MPN) requires manufacturing service capability (MSC) information of each party be accessible, understandable, and processible by all others in the network. However, at the present time, MSC information is typically encoded according to local proprietary models, and thus is not interoperable. Related existing works are primarily for integration in "isolated automation" of pair-wise or small size networks and thus are not adequate to deal with the high degree of diversity, dynamics, and scales typical for a MPN. In this paper, we propose a model development framework which enables to evolve a reference model for MSC information based on the inputs from proprietary models. The developed reference model can serve as a unified semantic basis supporting interoperability of MSC information across these local proprietary models. Methodology for resolving structural and other semantic conflicts between deferent models in model development is also presented.

Keywords: manufacturing service capability, ontology development, pattern-based ontology transformation, canonicalization.

1 Introduction

Today, service capability information of manufacturers is typically represented according to some models developed by individual enterprises or communities. These local proprietary MSC information models are not interoperable because of their differences in service category, capability structure and values. As a result, manufacturers often have difficulty in quickly discover suppliers with required capabilities without a significant level of human involvement. A MSC information reference model that is semantically rich can help reconcile semantic difference among local proprietary models and increase access and precision to capability information. However, related existing works [1, 2, 3] are primarily for integration in isolated automation of pair-wise or small size networks with less semantic diversity and thus are not adequate to deal with the high degree of diversity, dynamics, and scales typical for a manufacturing production networks (MPN).

V. Prabhu, M. Taisch, and D. Kiritsis (Eds.): APMS 2013, Part I, IFIP AICT 414, pp. 325–333, 2013.

In this paper, we propose a framework that helps to develop such a MSC information reference model. This framework takes a transformational approach and is centered on the ability to evolve a reference model based on the inputs from proprietary models. And that ability is provisioned by the abilities to perform the semantic gap analysis which identifies the semantic differences between the input models and the reference model. The differences are then used to drive the evolution of the reference model. A challenge for semantic gap analysis in this framework is to deal with the structural conflicts between the input from the local models and the reference model. This is addressed by aligning the structural representations of the input with the set of modeling conventions used in the reference models known as ontology design patterns (ODPs).

The rest of the paper is structured as follows. In the next section, we describe the proposed model development framework. In Section 3, we discuss the possibility of semantic loss after ontology transformation. And, finally we describe related works before giving conclusion and future plans.

2 Model Development Framework

The proposed reference model development framework is outlined in Fig. 1. We assume that each proprietary model uses its own syntax such as relational databases, XML and XML schemas. In the first step (*Transformation*), these heterogeneous syntaxes are transformed into a common syntax (OWL in our framework). The output of this step is an input into the following *Canonicalization* step. Another input to the canonicalization step is the patterns library which contains ontology design patterns (ODPs) from the reference model. The canonicalization step resolves the structural conflict by aligning the structural representations of a proprietary model with a set of modeling conventions used in the ODPs. The output of the canonicalization step is called canonicalized proprietary model. In the next step (*Semantic Gap Analysis*), the semantic differences between the canonicalized proprietary model and the reference model are identified. The differences are then used to evolve the reference model. The changes in the reference model are then verified for consistency in the *Verification/Reasoning* step. Details of each of these steps and ODP are given next.

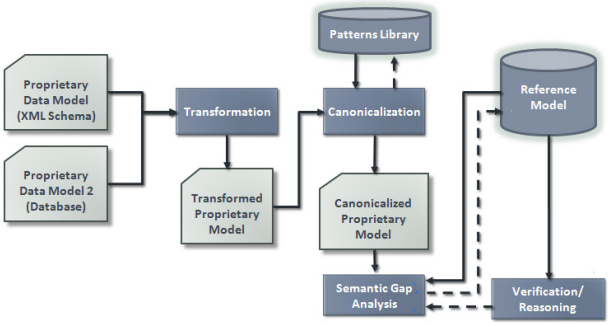

Fig. 1. The reference model evolution framework

2.1 Transformation

This step takes each proprietary model as input and converts them into common syntax of OWL. The output of this step is called a transformed proprietary model. Because the proprietary models take different conceptualization of the domain the result of the transformation still remains to be structurally and semantically different. This step can be largely automated when the proprietary model is well-structured (e.g., relational database and XML schema) as opposed to unstructured (e.g., text, HTML). In our work, we assume that the proprietary models are in relational databases. Currently, there are many tools to support RDB-to-OWL transformations. We have investigated D2RQ in particular and found that the D2RQ is capable of supporting the automatic transformation for the proposed framework [14, 15].

Fig. 2 below shows an example of the transformation from a relational database table into OWL. The *PartLength* table is converted into an *owl:Class* named *s:PartLength*. The *LengthValue* attribute is converted into *owl:DataProperty* named *s:PartLength_Value*. The record, which has the value `4` as its key, is converted into an *owl:NamedIndividual* named *s:PartLength_4*. Its *Value* attribute value *6cm – 48cm* is an *xsd:String* value of the *s:partLength_Value* data property.

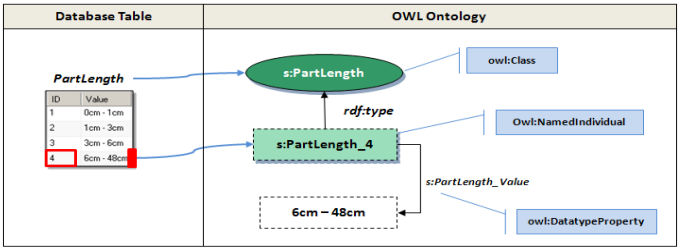

Fig. 2. RDB to OWL transformation example

2.2 Ontology Design Patterns

An ODP is a reusable successful solution to a recurrent semantic modeling problem, written in an ontology language such as OWL [8]. ODPs can be viewed as generic, small ontologies or ontology components with explicit documentation of design rationales and best reengineering practices. Pattern-based approach for ontology design has been gaining popularity recently because by reusing existing tested patterns as building blocks a domain ontology can be constructed quickly with high quality and less conceptualization divergence. A large amount of ODPs have been proposed in the ontology design community [9]. In this paper, we define a formal representation of ODPs as follows and show in Fig. 3 a simple exemplary ODP that captures the concept of *LengthCapability* with two *DatatypeProperty*, *hasMin* and *hasMax*.

- Definition 1: ODP is a 2-tuple {Sig, BE}
  - Sig is a non-empty set of Ontology Signature
  - BE is a non-empty set of binding expressions

- Definition 2: Ontology Signature is a 2-tuple {E, X}
  - E is a non-empty set of entity and literal parameters
  - X is a set of axioms
- Definition 3: Binding Expressions is a 2-tuple {P, C}
  - P is a non-empty set of parameters in the Signature
  - C is a non-empty set of concepts and values giving a specific meaning to the ODP signature

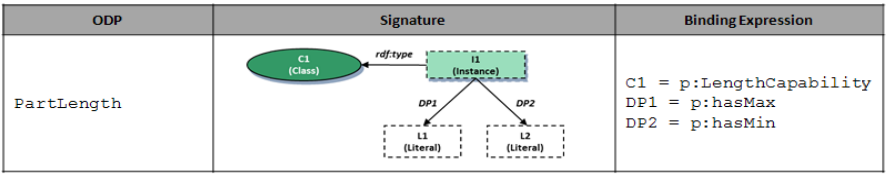

Fig. 3. Exemplary ODP

2.3 Canonicalization

The canonicalization is a methodology to resolve structural differences between two different ontologies. The canonicalization aligns the structural representations of a proprietary model with the set of ODPs used in the reference model. The canonicalization consists of semantic annotation and pattern-based ontology transformation.

- Semantic Annotation

The semantic annotation is to identify correspondences between ODPs of the reference model (called the target) and the ontology artifacts of the transformed proprietary model (called the source). The semantic annotation process starts with establishing terminological links between entities and literals in the source and those in the ODPs of the target by matching their meaning or semantics. Many approaches have been proposed for determining semantic similarity between entities of two ontologies [7, 8]. The similarity can be measured purely based on lexical information in the labels of two ontology artifacts. And, the structural information can be considered as well. For instance, *s:PartLength* is linked to *p:LengthCapability*. With these terminological links and their binding expressions, the ODPs that are related to these terms are retrieved as shown in Fig. 4. The resulting correspondence indicates which ODP of the target should be applied to which set of the source artifacts.

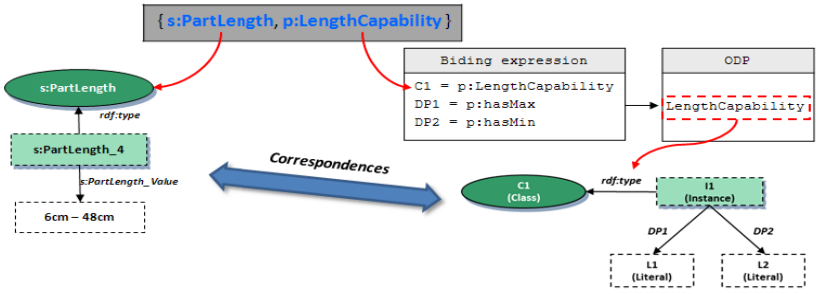

Fig. 4. Correspondence between the transformed proprietary model and ODP

- Pattern-based Ontology Transformation

The pattern-based ontology transformation first identifies sub-structures of the transformed proprietary model that is semantically close to a target ODP. Then, the patterns of the identified sub-structures are identified and they are called source ontology patterns and represented by the formal representation given in Section 2.2. In the next step, the pattern transformation rules are generated. A pattern transformation rule specifies relations between parameters in the source and target ODPs. These relations describe how the source ontology pattern should be transformed to the corresponding target ontology pattern. For instance, let's assume that the target ontology pattern has two data properties including *hasMin* and *hasMax* and the source ontology pattern has only one data property that represents the part length capability min and max values with a single literal value such like *6cm – 48cm*. To deal with this situation, a literal value pattern is defined with the string regular expression, *([0-9]+)cm - ([0-9]+)cm*. The first group in the regular expression corresponds to the minimum part length value and the second group corresponds to the maximum part length value. Fig. 5 below illustrates the pattern transformation rule for this situation.

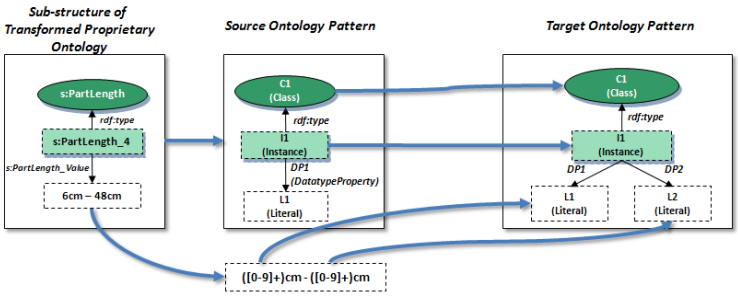

Fig. 5. Pattern transformation rule generation

Then, the transformation rules are executed on the transformed proprietary model and it is called pattern transformation. The pattern transformation is divided into two sub-processes, pattern instances detection and transformation rule application. The pattern instances detection process applies the source ontology pattern to find all pattern instances in the transformed proprietary model using the SPARQL. The SPARQL query generated from the source ontology pattern is shown in Fig. 6. It retrieves all the pattern instances which conforms the source ontology pattern. A pattern instance is a set of the transformed proprietary model's entities and literals that use the pattern.

```
PREFIX rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX s:  <http://www.nist.gov/el/sid/msnm/PortalB.owl#>
SELECT distinct *
    WHERE {
        ?I1 rdf:type s:PartLength .
        ?I1 s:PartLength_value ?L1 .    }
```

Fig. 6. SPARQL query generated from the source ontology pattern

The transformation rule application process applies the transformation rule on the retrieved entities and literals in the transformed proprietary model. The output entities and literals provide all the necessary elements to establish the set of axioms in the target ontology pattern. The result of the pattern transformation is called canonicalized proprietary model which is the final output. The canonicalized proprietary model is expected to be structurally aligned with the structure of the reference model.

2.4 Semantic Gap Analysis

The semantic gaps between the canonicalized proprietary model and the reference model can be identified by mapping between those two different models. The mapping can be done manually and/or semi-automatically. Works in ontology matching in the past decade are summarized and analyzed in [7]. These works have been largely focused on achieving full ontology mapping or alignment, and, as indicated by the authors, left several open issues, particularly the issues of matching across entity types (i.e., to match across structural conflicts). However, in our framework, the structural conflicts are already resolved through the canonicalization. Therefore, we expect that those existing ontology matching algorithms would be suitable to this mapping task. The identified semantic gaps such as newly found concepts, relations, and axioms are documented and used for evolving the reference model.

2.5 Verification/Reasoning

Semantic inconsistency errors can often been seen when mapping and merging different ontologies. Thus, ensuring that ontologies are consistent is an important part of ontology development. Therefore, if the reference model is evolved based on the semantic gap analysis, the reference model should be verified for guaranteeing the consistency of the evolved model. The verification/reasoning step checks and verifies consistency across the proprietary model, proprietary data and the reference model. This includes translation checking, consistency checking, redundancy checking, etc. If inconsistency is found in this step, the semantic gap that causes this inconsistency should be re-analyzed and the changes should be reconsidered, and the verification/reasoning and semantic gap analysis steps shall be executed in iterations until there is no inconsistency.

3 Discussion

In this section, we discuss the possible semantic loss in canonicalization. The canonicalization is a type of ontology transformation. A key requirement for ontology transformation is that while syntactical changes are being made to data structures, the semantic meaning of that data should not be changed. Although all of the data transformed from original structure to canonical form is syntactically correct, it may be semantically incorrect and results in information loss. Thus, it is essential to consider the semantic effects of syntactic changes to correctly perform canonicalization.

In [4], the authors sketched a set of possible ontology change operations and discussed the effects of these changes with respect to the instance data preservation. The effects of the ontological changes can be classified as information-preserving, transformable, and information loss. The ontological changes with information loss should be very carefully handled while performing canonicalization. If one entity exists only in the proprietary model and does not exist in the reference model, we need to investigate whether the entity is meaningful and should be considered as a new concept or not. In the case of the former, the entity should be kept and additional information should be annotated so that it would be listed up in the gap analysis step. And, in the case of the latter, the entity should be excluded from the transformation rule and as a result it would be removed after canonicalization.

4 Related Work

In this section, we briefly review existing works that are relevant to ontology construction. The key ontology engineering activities in ontology construction are summarized in [10], which also stressed the need for guidance on ontology reuse. Guidance for building ontologies either from scratch or reusing other ontologies can be found in [11]. After establishing the ontology, an important issue is that ontology tends to change and evolve over time due to changes in the domain, changes in conceptualization, or changes in the explicit specification [12]. Works in managing ontology change and evolution are well-summarized in [13].

Canonicalization has been studied in several works. [5, 6] provide workable methods and tools including key enablers. They provide well defined XML schema for the pattern transformation definition (including pattern definitions and transformation rules). For pattern instances detection engine, PATOMAT provides the functionality to generate SPARQL query from the pattern transformation definitions and its pattern transformation engine uses OPPL application interface for pattern transformation. PATOMAT also has the TPEditor component which is an editor of source and target ontology patterns and associated transformation rules.

5 Conclusion and Future Works

Our work is motivated by the need to improve precision and interoperability of manufacturing services models to enable sharing precise information models of suppliers' manufacturing services in manufacturing production networks. In order to effectively develop such manufacturing services models, we propose a model development framework which enables a reference model to evolve based on the inputs from proprietary or other existing standard models. The differences between the input models and the reference model identified by the semantic gap analysis are used to evolve the reference model. The reference model is then verified for ensuring its consistency.

In this framework, we propose a canonicalization methodology to align the structural representations of a proprietary model with the set of modeling conventions (ODPs) used in the reference model. The benefit of canonicalization is the reduction

of the mapping complexity by reducing the number of entities and structural complexity in the manufacturing service models and the number of mappings in semantic gap analysis.

As of our future work, we are working on analyzing requirements for manufacturing services capability to create a basic information model which will be a basis to derive representation patterns for manufacturing services capability. Based on the basic information model, we will create a library of representation patterns for the manufacturing services capability. We will also be conducting more in depth researches on core components of the model development framework. Finally, we will develop processes and tools to create a reference model using representation patterns for the manufacturing services capability.

References

1. W3C Semantic Web Activity, http://www.w3.org/2001/sw/
2. Kalfogloul, Y., Schorlemmer, M.: Ontology Mapping: The State of The Art. The Knowledge Engineering Review (2003)
3. Kim, J.: A Semantic Analysis of XML Schema Matching for B2B Systems Integration. PhD dissertation, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County (2011)
4. Natalya, F.N., Michel, K.: Ontology Evolution: Not the same as Schema Evolution. Knowledge and Information Systems 6(4), 428–440 (2004)
5. Šváb-Zamazal, O., Svátek, V., Scharffe, F., David, J.: Detection and Transformation of Ontology Patterns. In: Fred, A., Dietz, J.L.G., Liu, K., Filipe, J. (eds.) IC3K 2009. CCIS, vol. 128, pp. 210–223. Springer, Heidelberg (2011)
6. Svab-Zamazal, O., Svatek, V.: OWL Matching Patterns Backed by Naming and Ontology Patterns. In: Znalosti (ed.) 10th Czecho-Slovak Knowledge Technology Conference, StaraLesna, Slovakia (2011)
7. Pavel, S., Jérôme, E.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engneering X(X), 1–20 (2012)
8. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 262–276. Springer, Heidelberg (2005)
9. Presutti, V., et al.: NeOn Project Delivery - D2.5.1. A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies, http://www.neon-project.org/web-content/images/Publications/neon_2008_d2.5.1.pdf
10. Jones, D., Bench-Capon, T., Visser, P.: Methodologies for Ontology Development. In: Proc. of the IT&KNOWS Conference of the 15th IFIP World Computer Congress (1998)
11. Staab, S., Schnurr, H.P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1), 26–34 (2001)
12. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Smi 2002-0926, University of Stanford, Stanford Medical Informatics, USA (2002)
13. Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. The Knowledge Engineering Review Journal 23(2), 117–152 (2008)

14. Bizer, C.: D2R MAP: A Database to RDF Mapping Language. In: Proceedings of the 12th International World Wide Web Conference, Budapest, Hungary (2003)
15. Bizer, C., Seaborne, A.: D2RQ–treating non-RDF databases as virtual RDF graphs. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) Proceedings of 3rd International Semantic Web Conference, Hiroshima, Japan. LNCS, vol. 3298. Springer, Heidelberg (2004)