Ontology-Based Dynamic Forms for Manufacturing Capability Information Collection

Yun Peng, Yan Kang

Ontology-Based Dynamic Forms for Manufacturing Capability Information Collection

Yun Peng and Yan Kang

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
{ypeng, kangyan1}@umbc.edu

Abstract. Building flexible manufacturing supply chains requires availability of interoperable and accurate manufacturing service capability (MSC) information. These requirements can be met by encoding the MSC information using shared domain ontologies. However, difficulty in understanding the syntax and semantics of the shared ontologies hinders the adoption of such ontology-based approach. In this paper, we propose an Ontology-based eXtensible Dynamic Form (OXDF) user interface architecture to assist non-expert users to collect MSC information and represent that information as instances of the shared domain ontology. To achieve this result, we introduce three key innovations: 1) intelligent ontology navigation that dynamically generates forms and form components from the relevant parts of the ontology; 2) intelligent search engine that helps finding relevant ontology entities; and 3) an update mechanism that allows users to define new terminologies to the shared domain ontology.

Key words: supply-chain, ontology, interoperation, semantic similarity

1 Introduction

The competitiveness of manufacturers is increasingly defined by the combined capabilities of the suppliers that make up the OEMs' supply chains [1]. Effective communication between suppliers' manufacturing capabilities and customers' requirements is essential for building a flexible network of suppliers in a supply chain. The quality of manufacturing service capability (MSC) information is one of the determining factors for the quality of manufacturing information exchange and it can be measured by its *accuracy* and *semantic interoperability*.

Today, MSC information is typically published either on the suppliers' web sites or registered at e-marketplace portals. MSC information published on web pages is typically not understandable by computer programs and thus lacks interoperability. Advanced techniques from information extraction and natural language processing may help obtain structured MSC information from web pages. However, due to inherent ambiguity of natural languages, this approach typically leads to a significant loss of accuracy. Some commercial e-marketplace portals require users to enter their MSC information in a uniform format according to the portal's proprietary MSC information models. This gives potential for interoperability to users within a portal, but not between portals. Besides, these proprietary models are typically insufficient to precisely capture the intended meaning of users' capability information.

All these problems can be addressed by adopting well-defined domain ontology that allows MSC information of individual suppliers to be described accurately with-

out ambiguity and understood correctly among all stakeholders sharing that ontology. However, manufacturing domain experts have difficulties adopting the ontology-based approach. This is because 1) the syntax and semantics of formal ontology languages are not familiar to users and have a steep learning curve; 2) the scope of a comprehensive domain ontology is typically very large in its size and complex in its structure; and 3) as a newly emerging technique, tools friendly for inexperienced users are not yet sufficiently mature and widely available. In this paper, we report our research in providing such tools. Specifically, we propose a new Ontology-based eXtensible Dynamic Form (OXDF) user interface architecture to assist non-expert users to collect MSC information and encode it using the vocabulary of the ontology. To achieve this result, we introduce three key innovations, including: 1) intelligent ontology navigation that dynamically generates forms and form components from the relevant parts of the ontology; 2) intelligent search engine that helps finding relevant ontology entities; and 3) an update mechanism that allows the user, if needed, to define new terminologies outside the ontology.

2 Manufacturing Capability Ontology

Although a great number of ontologies have been developed and published, only few of them are specific for the manufacturing or related domains. These few include for example GoodRelations[1], an upper ontology for e-commerce, and MSDL (Manufacturing Service Description Language) [2], a detailed ontology for machining services. Ontology segments used in many of our examples are taken from MSDL.

Since manufacturing is a huge and complex domain, it is inconceivable to define a single comprehensive and detailed ontology for the entire domain. Our vision is that the semantic model for the manufacturing domain is composed of a set of ontologies, including one or more upper domain ontologies, a number of detailed manufacturing subdomain ontologies and some widely accepted ontologies for general concepts such as temporal, spatial and geographical. With properly defined links and mappings, these ontologies are connected and form a single logic system that we will refer to as *Manufacturing Capability Ontology* (MCO) in the rest of this paper.

We also assume that MCO is written in OWL. OWL is a combination of RDF data model[2] and a dialect of description logic [3]. The primary advantage of RDF is that it can easily integrate data with different levels of structure, while the description logic and its well-defined formal semantics allow one to perform logical reasoning over the ontology and its instances. With these and other benefits, OWL has emerged as the de facto standard for defining ontologies and sharing them on the web.

3 Overview of OXDF Architecture

The Ontology-based eXtensible Dynamic Form (OXDF) user interface architecture is driven by the Manufacturing Capability Ontology (MCO). The architecture presents

[1] http://www.heppnetz.de/projects/goodrelations/
[2] http://www.w3.org/TR/rdf-primer/

an ontology in a *form format* because forms are widely used in commercial e-marketplaces and familiar to and easy to fill by manufacturing domain users. Each ontology class is rendered as an *atomic form*, consisting of the class name and a set of its defined properties called *form components*. As shown in the diagram in Figure 1 below, OXDF starts with a base form (step 1), which is one of the top-level classes selected by users, depending on what kind of information they want to enter. For entering basic information about a company, the base form will present either a *supplier* class or *customer* class; for capability information a *service* class will be presented.

An OXDF form assists users to collect MSC information by navigating through the capability hierarchy of MCO to reach a class that is closest to that particular capability (step 2). In this way, a sequence of atomic forms for the classes traversed during the navigation is presented. Users can enter capability information using the form components (properties) in these atomic forms. When information about that capability is entered, an instance of that capability class is created (step 3) and stored in the instance repository (step 4). Users may fail to find proper atomic form or form component for a particular content. OXDF helps with this situation with its ontology search engine which returns a list of classes or properties that are semantically close to the user's intended content (step 2.1). If nothing is returned or none of the returned entities is satisfactory, users can create a new atomic form or form component (step 2.2), and in effect define their own classes and properties outside of the given MCO. Newly created entities will be stored in another repository (step 5).

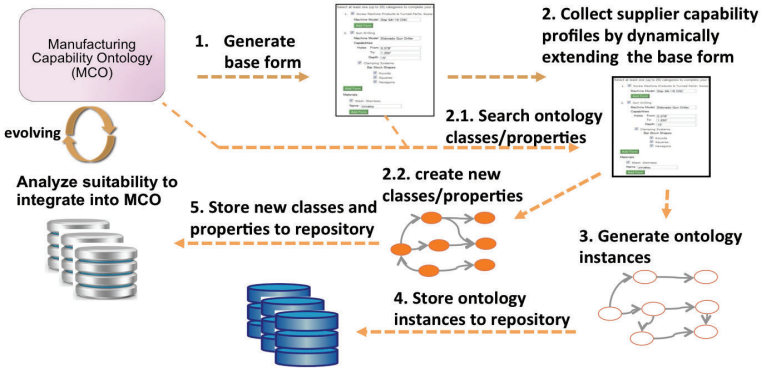

Fig 1. OXDF overview

4 Ontology Navigation

Related classes in MCO are either organized in a class hierarchy or defined as domain/range of an object property. Thus, OXDF helps users navigate MCO in two ways: *vertically* navigating the class hierarchy by following the subclass links and *horizontally* navigating by following object property links.

4.1 Navigating Class Hierarchy:

OXDF renders the class hierarchy as a column tree, as shown in Figure 2 below. Each column of the column tree contains classes at the same level. The leftmost column contains the base form for the chosen top-level class. When the user selects any class

in a column, a new column appears on its right containing all subclasses of that class. The column tree is thus expanded until the user finds a class in the right-most column that is closest to her needs. After it is done for one capability, the process starts again for another capability.

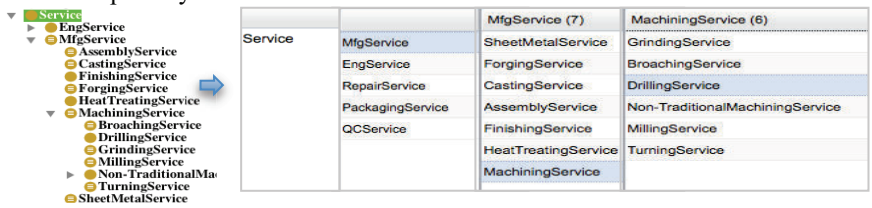

Fig 2. Column Tree (right) for a segment of MCO class hierarchy (left)

4.2 Navigating Object Properties of Classes:

When a class is selected during the traversal of the class hierarchy, the atomic form of that class is created. The components in this atomic form are class properties. Users fill capability information into these components. There are two types of properties in OWL: *datatype property* and *object property*. For datatype property, user only needs to input value of the given primitive type. The object property is more involving. It links the current class (as its domain) to another class (as its range). Completing an object property means to create an instance of the range class. Therefore, when users select an object property in the atomic form, an atomic form for the range class is presented.

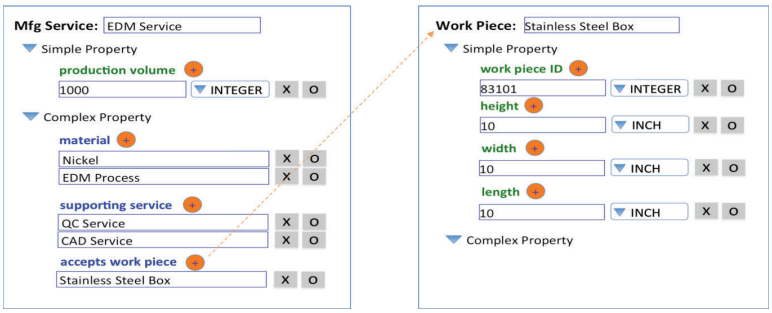

Fig 3. atomic form for "MfgService" class

Figure 3 shows an example of collecting information for a capability called "EDM Service". This service is determined as belonging to the class "MfgService" during navigation and the atomic form for this class is presented (left) with two lists of components: *Simple Property* list for datatype properties and *Complex Property* list for object properties. In this example, EDM Service has one datatype property "hasProductionVolume" and several object properties such as "hasMaterial", "hasSupportingService", and "acceptsWorkpiece". These names are converted to natural language-like labels in the form as shown on the left in Figure 3. For each datatype property, a space is given for the user to fill the value. For each object property, the name of its range class is presented together with the property name. When an object prop-

erty is selected, the atomic form for the range class is presented (right), users can then fill in the properties in that form with more detailed information.

5 Form Extension

When users fail to find an appropriate form (class) or components (properties) to fill in a particular content when navigating MCO, they can extend the forms by searching and reusing entities existing in the ontology or creating and adding new ones if she fails to find anything suitable from the search. Specifically, suppose the user stops at class *C* in the navigation process. If she cannot find a proper subclass of *C* in the column tree for her capability information, then a new class can be created and added as a subclass of *C*. If she finds *C* is a suitable class but does not have components she needs to describe the attributes of the intended capability information, then one or more components can be added into the form for class *C*, thus creating new properties for *C*. These two kinds of form extension are discussed next. The search engine itself will be presented in the next section.

5.1 Form extension by creating new classes

Users can add a class at any level in the class hierarchy. The added class may be either a new class or an existing class that has already been defined in the MCO. Reuse of existing terms is highly desirable since it reduces the need for ontology revision and expansion. When a suitable class, say *D*, is found by search, the user can choose to add the class *D* to the current location (to become a subclass of *C* and displayed in the column tree) or she can navigate to the location in the hierarchy where class *D* is found. For example, when the user needs a "Boring" class as a subclass of "HoleMaking", she can first search the ontology to see if such a class already exists. The search engine returns a list of terms defined in MCO that are close (by semantic similarity) to "Boring", as shown on the left of Figure 4. This list indicates that there is a "Boring" class in the ontology and it has superclass "Turning". After making it a subclass of "HoleMaking", "Boring" class inherits all the properties of "HoleMaking" in addition to all properties it already has as a subclass of "Turning".

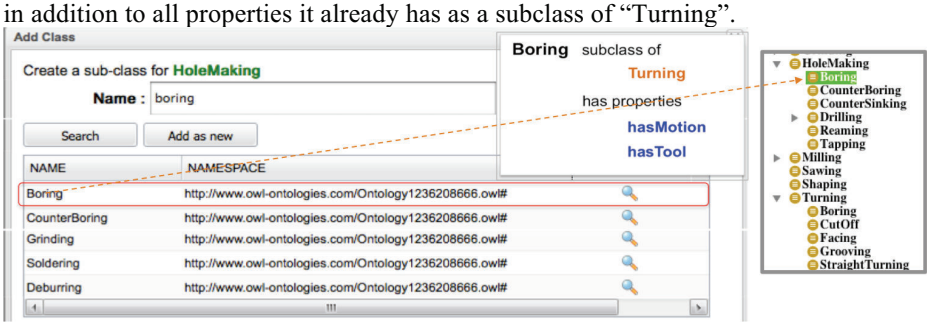

Fig 4. Search for "Boring" class

Sometimes, due to the lack of familiarity with the ontology, the user may navigate along a wrong path in the class hierarchy. For example, the user may intend to enter

information about her lathe machine but ends up at the form for "Machining" which as a class for processes does not have any subclass for lathe machine. Search of the ontology finds that "MachineTool" has "Lathe" as its subclass. At this point, user might notice that she has navigated the MCO along a wrong path, and it makes more sense to keep "Lathe" as a subclass of "MachineTool", since by definition Lathe is a class of physical resources but "Machining" is a class of processes. OXDF can redirect users to the right location in MCO where "Lathe" is defined. Then, users can enter the machine information there.

When the users fail to find the desirable class in the search results, they can choose to simply create a new class *E* and add it as a subclass of *C*. *E* will appear in the column tree and inherit all the properties of *C*. Additional properties can be added to *E* as described next.

5.2 Form extension by creating new properties

Similarly, when users choose to extend the form for a class with additional properties, they can search the ontology for possible reuse or create new properties. Users can create both object properties and datatype properties. For creating a datatype property, they only need to provide name and value type of this property. For object property, they can optionally create the range of this property. Since the range of an object property is itself a class, it can be provided, again, by either searching the ontology for existing class or creating a new one.

User-created classes and properties, together with their instances can be used as a material basis for potential ontology revision and expansion. For example, if the number of instances of one or a group of similar user-created classes (or properties) reaches a threshold, then a closer analysis is called for to determine if it is appropriate to create a new class and add it into the current MCO.

6 The Search Engine

The input to OXDF search engine is a word or phrase users use to describe the entity (class or property) they want to find in the ontology. The search engine will try to identify all existing ontology entities that have high *semantic similarities* with the user input. Simple keyword search often fails for this task due to the inherent ambiguity of natural languages. OXDF search engine effectively deals with these difficulties by analyzing the synonyms of the entities and utilizing the ontology's subsumption hierarchies of classes and properties.

Let e_1 denote the user search input, e_2 an ontology entity whose similarity to e_1 is to be determined, and $H(e_2)$ the subsumption hierarchy of e_2 whose details are given in Subsection 6.2. The overall similarity between e_1 and e_2 is given as

$$w_1 \cdot Sim_L(e_1, e_2) + w_2 \cdot Sim_H(e_1, H(e_2)) \quad (1)$$

where $Sim_L(e_1, e_2)$ computes the similarity between labels of e_1 and e_2 and $Sim_H(e_1, H(e_2))$ computes the similarity between e_1 and the subsumption hierarchy of e_2. w_1 and w_2 are weights for the two similarities, and are set to 0.5 by default.

6.1 Label similarity computation

Part of the semantics of an entity lies in the words used to label that entity. Label similarity computation attempts to measure semantic similarity between words in two entities e_1 and e_2. We use a synonym-based algorithm proposed by Kang [4] to calculate $Sim_L(e_1, e_2)$. This algorithm first draws two sets of synonyms of words that compose e_1 and e_2 respectively from WordNet [5]. $Sim_L(e_1, e_2)$ is computed based on pairwise n-gram similarity measures between the words from the two sets of synonyms.

6.2 Structural similarity computation

As stated earlier, OXDF search engine takes subsumption hierarchy of ontology entity (i.e., e_2) into consideration when computing similarity between e_1 and e_2. Without considering such contextual information, search engine might miss important entities that are semantically related to the user's input but with lower label similarity.

$H(e_2)$, the structural information used by the search engine, is the set of entities from the subsumption hierarchy of e_2, including all ancestors and descendants of e_2 but not e_2 itself. Let t denote an entity in $H(e_2)$. Then the relatedness between t and e_2, is denoted as $Rel(t, e_2)$. OXDF search engine uses Wu-Palmer similarity [6], a normalized distance measure to calculate $Rel(t, e_2)$:

$$Rel(t, e_2) = \frac{2 \cdot depth(LCA(t, e_2))}{SL(t, e_2) + 2 \cdot depth(LCA(t, e_2))} \tag{3}$$

where $SL(t, e_2)$ is the length of the shortest path between t and e_2 , and $LCA(t, e_2)$ is the lowest common ancestor of t and e_2 . Then, $Sim_H(e_1, H(e_2))$, which measures the similarity between e_1 and the subsumption hierarchy of e_2, is computed as follow:

$$Sim_H(e_1, H(e_2)) = \arg\max_{t \in H(e_2)} (Sim_L(e_1, t) \cdot Rel(t, e_2)) \tag{4}$$

Take the similarity computation between a user-provided property "shapability" and a property "hasColdFormability" defined in MSDL as an example. The subsumption hierarchy of "hasColdFormability" contains two properties: "manufacturingProperty" and "hasShapability". Table 1 shows label similarity and relatedness among "shapability" property and properties in hierarchy of "hasColdFormability"

Table 1. *Rel and* Sim_L between properties

properties in H("*hasColdFormability*")	Sim_L: label similarity with "*shapability*"	*Rel*: relatedness with "*hasColdFormability*"	$Sim_L \cdot Rel$
manfacturingProperty	0.12	0.50	0.06
hasShapability	1.00	0.80	**0.80**

The highest score among all the $Sim_L \cdot Rel$ values is 0.80. Thus, Sim_H("shapability", H("hasColdFormability")) is 0.80. Applied Kang's approach [4], Sim_L("shapability", "hasColdFormability") is 0.34. By applying (1), the overall similarity between "shap-

ability" and "hasColdFormability" is 0.57, which is much higher than 0.34 when only label similarity used.

7 Related Work

OXDF is developed based on our earlier work called XDF [4] which helps users navigating XML schema and generating XML instances for MSC information. The major shift from XDF to OXDF is that we adopt OWL ontology as the underlying formal model instead of XML schemas. The main motivation for this change is that XML schemas have relatively limited formal semantics and thus information encoded in XML instances suffers semantic ambiguity, making information less accurate and interoperability harder to be achieved.

Existing ontology engineering tools also support various ontology manipulations. However, these tools are primarily designed for professional ontology engineers. OXDF, on the other hand, aims at helping domain personnel who typically are knowledgeable of the manufacturing domain but lack experience in the syntax and semantics of the logic system of OWL ontologies. A number of tools or plugins tailored for non-expert users have also been reported in the literature [7]. Their main purpose, however, is to help users to access the ontology and obtain relevant entities by various navigational and query methods, not to collect the information from the users and organize the collected information as ontology instances as OXDF does. Also, they do not support dynamic creation of new classes and properties.

8 Further Work

We will investigate alternative technical approaches for search and ontology navigation. Search may become more efficient by focusing on the relevant parts of the ontology rather than searching the entire ontology as we have in the current architecture. This is extremely important because a comprehensive MCO would be very big and most of the classes/properties are not relevant when collecting information about a particular manufacturing capability. We are also interested in extend our OXDF to help semi-automatically extract capability information from manufacturers' web sites. With the guidance of the MCO, information extraction may become more accurate and the results can be used to tentatively populate the dynamic form before involving the human users.

References

1. Sturgeon, T.J., 2002, "Modular Production Networks: A New American Model of Industrial Organization," *Industrial and Corporate Change*, 11(3), 451 – 496.
2. Ameri F. and Dutta, D., 2006, "An Upper Ontology for Manufacturing Service Description," *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Philadelphia, 10 – 13.
3. Markus K., František S., Ian H., "A Description Logic Primer," In CoRR abs/1201.4089. arxiv.org 2012.

4. Kang, Y., Kim, J., and Peng, Y., 2011, "Extensible Dynamic Form Approach for Supplier Discovery,", *Proceedings of Information Reuse and Integration, 2011 IEEE International Conference*, Las Vegas, NV, August, 2011, 83-87.
5. WordNet, http://wordnet.princeton.edu/
6. Rada, R., Mili, H., Bicknell, E., and Blettner, M., 1989, "Development and Application of a Metric on Semantic Nets," *IEEE Transaction on Systems, Man, and Cybernetics*, 19 (1), 17-30.
7. Franconi, E., Guagliardo, P., and Trevisan, M., 2010, "An Intelligent Query Interface Based on Ontology Navigation," *Workshop on Visual Interfaces to the Social and Semantic Web*.