DODO: A Mechanism Helping to Dynamically Construct Domain Ontologies for Services Integration

Chen Liu  
Software Division, Institute of Computing Technology, Chinese Academy of Sciences  
Graduate University of the Chinese Academy of Sciences  
0086-10-88455001-618  
liuchen@software.ict.ac.cn

Wanghu Chen  
Software Division, Institute of Computing Technology, Chinese Academy of Sciences  
Graduate University of the Chinese Academy of Sciences  
College of Mathematics and Information Science, Northwest Normal University, China  
0086-10-88455001-618  
cwhmail@vip.sina.com

Yanbo Han  
Software Division, Institute of Computing Technology, Chinese Academy of Sciences  
0086-10-88455012  
yhan@ict.ac.cn

ABSTRACT
A challenging problem in forming virtual organizations by integrating standalone networked services is their semantic heterogeneity. Virtual organizations correspond to logic domains that vary, and it is often impossible to pre-construct a global and stable ontology. This paper proposes a semi-automatic mechanism called DODO, with which one can dynamically build up ontologies for individual logic domains, and discusses its working principles. In the context of some practical applications, DODO has been tested and evaluated.

Categories and Subject Descriptors
D.2.12 Interoperability – data mapping;  
I.2.4 Knowledge Representation Formalisms and Methods – semantic networks;  
I.2.6 Learning – knowledge acquisition.

General Terms

Keywords
Virtual organization, Ontology construction, Ontology Matching, Knowledge Absorbing

1. INTRODUCTION
Aiming at realizing spontaneous resources sharing and business collaboration, virtual organizations (VO) \(^1\) can be established on demand by integrating standalone networked services in an open and dynamic environment. Along with the establishment of VO, a logic domain is demarcated. To enable resource sharing, knowledge about this domain should be firstly specified to reach semantic consensus. Domain ontologies are often exploited for knowledge specification. However, the characteristics of a VO plague the construction of domain ontologies. First, a VO is dynamic and inconstant. In a VO, resource entities are allowed to join and leave freely. Second, a resource entity is only responsible for its own knowledge management, which incurs the absence of a priori agreement about the ontology vocabulary and the language for knowledge specification. Consequently, domain experts could hardly pre-construct a global and stable ontology for a VO through traditional build-from-scratch approaches \(^2\). In addition, taking the costs of construction into account, build-by-integrating-others approaches \(^3\) are also impractical in our cases. Furthermore, these approaches do not support simultaneous integration of multiple sources as only two ontologies can be integrated at one time.

To enable the VO, constructing ontology dynamically and continuously is required. In this paper, we propose a semi-automatic mechanism called DODO (Dynamically Construct Domain Ontology). This mechanism doesn’t require that the ontology be constructed beforehand. On the contrary, it advocates consummating the ontology during the applications. The DODO mechanism avoids not only the shortcomings of inflexibility and hardcoded pre-construction provided by build-from-scratch approaches, but also the high construction and maintenance cost possessed by build-by-integrating-others approaches. Moreover, this mechanism can support simultaneous integration of multiple resource entities. The rest of this paper is organized as follows: In the section 2, we present related works. In the section 3, we simply introduce the rationales of a VO. Then, the rationales of DODO are discussed in the section 4. In the section 5 and 6, implementation technologies of DODO are explored in detail. In the section 7, the practical application and evaluation of DODO are discussed. Finally, we draw a conclusion and briefly state our future works.

2. Related work
The traditional approaches for ontology construction can be classified into following two categories: build-from-scratch approaches, in which domain experts analyze the application domain and construct ontology from scratch; build-by-integrating-others approaches, in which ontologies are constructed by integrating and modifying already existing...
ontologies. The build-from-scratch approaches construct ontologies beforehand and result in inflexible and hard-to-change ontologies. Build-by-integrating-others approaches are hard to implement and apt to lose original semantics heavily. Besides, it cannot support multiple entities exchanging resources simultaneously required by a VO.

2.1 Build-from-scratch approaches

Today, there is no unified methodology to constructing ontology yet. Each work group will construct ontology depending on its own methodology which is proposed in order to meet the needs of its own project. The representative methodologies and projects are demonstrated as follows: Enterprise Ontologies and skeletal methodology [3], TOVE ontology and evaluation methodology [6] as well as CHEMICAL and Methontology methodology [11].

2.2 Build-by-integrating-others approaches

Apart from build-from-scratch approaches, build-by-integrating-others approaches, such as ontology integration, focus on how to construct the ontology by reusing existing ontologies. Ref [8] clarifies three different meanings of the word “integration”. The first meaning is reusing other available ontologies to build a new ontology. PhySys (The Physical Systems ontology) [9] is a typical representative of this kind of integration for reusing EngMath ontology [10]. The second meaning is merging different ontologies about the same subject into a single one that “unifies” all of them. The ONION [11] and the MOMIS [12] are the typical works about this kind of integration. The last meaning is integrating ontologies into applications. In this case, one wants to introduce into an application one or more ontologies that underlie and are shared among several software applications. [13]

3. Introductions to Virtual Organizations

As figure 1 shows, the VO is formed on the basis of SATOR’s registry centers. Different resource entities such as providers, dealers and manufacturers may publish their available resources information to the registry centers to facilitate VO to discover needed resources. Registry centers are geographically distributed.

Figure 1. The Rationale of a VO.

A VO is formed by dynamically and continuously integrating needed resources from different resource entities in the runtime. An ontology is indispensable for a VO to shield the semantic incompatibleness. However, it is intractable to construct such an ontology as the semantic descriptions of chosen resources are required to be dynamically integrated into the VO ontology seamlessly. This means the VO ontology cannot be static and should keep evolving.

4. The rationales of DODO Mechanism

Considering interchangeability, the basic ontology model DODO based on is compatible with OKBC [14] knowledge exchanging model. The definition of this model is:

Definition 1. An Ontology model is a tuple OntModel:= (C, P, I, R), where:
- C: The set of classes. Classes are collections of objects that have similar properties. Classes constitute a subclass–superclass hierarchy with multiple inheritances.
- P: The set of properties. Properties can be divided into attributes and relations. They are usually first-class objects. That is, a property can exist without being attached to a particular class. Besides, properties can be transitive or symmetric, or have inverses.
- I: The set of instances. Instances are individual members of classes. We use the term instance data to refer to instances and their property values.
- R: The set of restrictions. Each property has a set of restrictions on its values, such as cardinality and range.

Figure 2. The Rationales of DODO Mechanism.

As illustrated by figure 2, DODO mechanism provides a dynamic iterative process for constructing the ontology through learning knowledge from other resource entities in the running of the VO. DODO mechanism requires domain experts to pre-construct a basic semantic framework for the application domain first (see section 5). This semantic framework is called semantic skeleton. The construction of semantic skeleton is helpful to ensure learning correct knowledge and direct the construction of the ontology along the proper way. Then, in the running of the VO, new knowledge need to be captured, absorbed and verified. DODO mechanism provides many technologies such as matching, absorbing and verifying (see section 6) to address these requirements and flesh out the pre-constructed skeleton.

5. The semantic skeleton

The semantic skeleton is the starting point of the whole learning process, which aims at providing enough background knowledge to eliminate the possible ambiguities of learning. An example
illustrated in the figure 3 could demonstrate the necessity of it. Suppose two resource entities both provide the knowledge about “Person” resources. However, one is the “Person” who worked in the enterprise and the other is the “Person” who will login in a UNIX system. Obviously it is meaningless to learn knowledge from them both as they in fact don’t represent the same kind of resource. Therefore, to make clear what to learn and from which to learn, the background knowledge is required in advance. Nevertheless, how to define the background knowledge is a hard nut to crack, as it should provide enough information to eliminate the semantic ambiguities and doesn’t need to be complete. Back to the figure 3, let’s imagine that if the context can be predefined, then we will understand fully what the exchanging resources really are and what they can learn from others. In other words, if we can pre-construct the skeleton showed in the left side, we can know that our expected resources are “Person” in the enterprise information system and we should learn from the first “Person” resource.

Figure 3. The Necessity of the Semantic Skeleton.

The semantic skeleton is a basic semantic framework. It aims at eliminating the ambiguities of learning. We define it as follows:

**Definition 2.** An semantic skeleton is a tuple \( S := (C, P, Hc, Hp) \)

, where:

- **C**: The set of key concepts of a given application domain.
- **P**: The set of optional properties of a given application domain, which means domain experts can decide whether to predefine the properties and attach them to the concepts.
- **Hc**: The hierarchy of predefined key concepts.
- **Hp**: The hierarchy of predefined key properties.

Build-from-scratch approaches are best suited to construct such skeleton. We summarize our past experiences and propose the following steps of construction illustrated in the Figure 4.

Figure 4. The Steps to Construct Semantic Skeleton

(1) Identifying the Domain: Domain experts should identify the domain that the skeleton will be applied first. They must make clear why the skeleton is built, their intended uses, and a range of intended users.

(2) Knowledge Capture: Identify the key concepts of application domain and attach the key attributes and relations to each concept.

(3) Knowledge Reusing: Judge whether existed ontologies can be reused. If no one can be reused, then construct new knowledge and add it to the skeleton to be built.

(4) Knowledge Encoding: Record and store generated knowledge.

6. Flesching out the semantic skeleton

6.1 Matching the ontology

**6.1.1 Matching features**

(1) Name Affinity

Name affinity means to quantify the level of matching two concepts based on their names. Referred to a global dictionary WordNet [18], DODO defines four kinds of lexical relations between names and a coefficient is assigned to each relation. As an example, in the figure 5, the name “Author” in the semantic skeleton and the name “Writer” in the resource entity A are the synonym and their name similarity is 1.

<table>
<thead>
<tr>
<th>Matching feature</th>
<th>Coefficient</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN</td>
<td>1</td>
<td>Synonym-of</td>
</tr>
<tr>
<td>BT/NT</td>
<td>0.8</td>
<td>Broader/Narrower Terms</td>
</tr>
<tr>
<td>RT</td>
<td>0.6</td>
<td>Related Terms</td>
</tr>
<tr>
<td>URT</td>
<td>0</td>
<td>UnRelated Terms</td>
</tr>
</tbody>
</table>

Table 1. The lexical relations defined in DODO

(2) Structure affinity

Structure affinity means to quantify the level of matching two concepts based on their attributes, which is proportional to the number of similar attributes. Two attributes are similar only when their names are similar and their domains are compatible. Here, two domains are compatible means they are same or one domain is the specialization of the other domain. As an example, in the figure 5, there are two similar attributes between the resource
“Author” in the semantic skeleton and the resource “Writer” in the resource entity A, therefore their structure affinity is 2/3. (3) Context affinity

Context affinity means to quantify the level of matching two concepts based on their contexts. Here, the context of a concept is the set of concepts which are directly adjacent to the given concept. The affinity between two adjacent concepts is evaluated considering their names and structural affinity.

6.1.2 Matching results

We define a global affinity as the linear combination of the previous affinities to assess the level of matching two concepts comprehensively. Exploiting the global affinity, the number of concepts matching a target concept depends on the level of closeness we want to impose on them. In DODO, the aim is to find only the concept(s) considered equivalent. Therefore, the level of closeness is represented as an equivalence threshold \( t_E \). Referred to the Equivalence Matching Strategy proposed by the resource entity RO, we denote \( M_C \) as the set of concepts of RO absorb are illustrated in the Table 2.

\[
M_C = \{ C' \in RO | GA(C, C') \geq t_E \}
\]

According to the definition 3, we can find the matching concepts and capture knowledge to be absorbed during the process of exchanging resources among resource entities. The captured knowledge can be divided into the following categories:

(1) New knowledge: It means, in the target ontology, matching knowledge cannot be found. This category of knowledge comprises the new concepts, new attributes and new relations.

(2) Updated knowledge: It means, in the target ontology, matching knowledge already exists. However, matching knowledge is a little different with the source knowledge. Therefore, it needs to update the target knowledge.

6.2 Absorbing the knowledge

6.2.1 The abilities to absorb

This phase aims at absorbing captured knowledge to impel the pre-constructed semantic skeleton to consummate. The abilities to absorb are illustrated in the Table 2.

<table>
<thead>
<tr>
<th>Table 2. The abilities to absorb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
</tr>
<tr>
<td>Derive new concept from existed concepts</td>
</tr>
<tr>
<td>Add a completely new concept</td>
</tr>
<tr>
<td>Change the name of an existed concept</td>
</tr>
<tr>
<td>Add a new attribute for a concept</td>
</tr>
<tr>
<td>Change the name of an existed attribute</td>
</tr>
<tr>
<td>Add a new relation among concepts</td>
</tr>
<tr>
<td>Change the name of an existed relation</td>
</tr>
</tbody>
</table>

6.2.2 The principle to absorb

In the running of VO, one resource entity may interact with many entities simultaneously. Consequently knowledge about even the same kind of resource may have multiple editions. As an example, considering describing the “age” attribute of concept “Person”, one resource entity will describe the “age” attribute as an “integer” but another resource entity will describe it as a “string”. Therefore, when adding “age” attribute to the concept “Person” in the semantic skeleton, we need to decide which type this attribute should have.

In the end, these contradictions will be eliminated by the domain experts. Nevertheless, domain experts are not omnipotent. If only depending on artificial analysis, it is very difficult to make clear all interdependent requirements and find the best way. Hence DODO will take on these burdensome analysis tasks. But if depending on machine to recommend, we need a principle to measure which knowledge are better and should be absorbed in the first place. Herein DODO proposes a consensus-driven principle for absorbing, which means that if a piece of knowledge is provided by most of resource entities, then it should be recommended in high priority.

There are two different roles related to the resource entities’ semantic environment, in which semantics providers offer their own knowledge and semantic users utilize the knowledge. However, the consensus is defined as the knowledge accepted by most of these two kinds of roles. Consensus can be divided into the following category:

(1) Consensus among semantics providers: This category of consensus is defined as the knowledge which provided by the most of providers. However, it is obvious that the description capabilities of providers are different. Domain experts have the deepest insight about the application domain, so their description capability should be highest. Accordingly DODO will attach a coefficient to every semantic provider to weigh its description capability. This description capability will be continuously adjusted according to the feedback of users.

(2) Consensus among semantics providers and users: The application of knowledge will be regarded as the important factor to adjust corresponding provider’s description capability. This means the more a piece of provided knowledge is used the higher the description capability of corresponding provider.

6.3 Verifying the constructed ontology

After absorbing the captured knowledge, we need to verify the correctness of constructed ontology. Referred from build-from-scratch approaches, we propose the following verified criteria: consistency, completeness and non-redundancy.

(1) Consistency

Consistency means no contradictory conclusions can be drawn from valid input definitions. That is, the constructed ontology should avoid the following errors:

1) Class inconsistency errors: For example, a class is defined as a specialization or generalization of itself.

2) Instance inconsistency errors: For example, an instance belongs to more than one class and these classes haven’t direct or indirect inheritance relations.

3) Semantic inconsistency errors: For example, a concept is defined as a subclass of a concept to which it does not really belong.
constructed ontology should avoid the following errors:

1) Incomplete concepts errors: Generally, an error of this type is made whenever concepts are classified without accounting for them all, that is, concepts existing in the domain are overlooked.

2) Incomplete attributes errors: An error of this type means descriptions of concepts are not enough detailed. Some important attributes are omitted.

3) Incomplete relations errors: An error of this type means some important relations among concepts are overlooked when fleshing out the skeleton.

(3) Non-redundancy

Redundancy is a type of error that occurs when redefining expressions that were already explicitly defined or that can be inferred using other definitions. Non-redundancy means that the constructed ontology should avoid the following errors:

1) Redundancy of class definition: It occurs when there are two or more classes in the ontology with the same formal definition, that is, the only difference between them is the name.

2) Redundancy of relation definition: It occurs when there is more than one explicit definition of any of the hierarchical relations. For example, if we define the class dog as a subclass of pet, and pet as a subclass of animal, when dog is also defined as a subclass of animal.

3) Redundancy of instance definition: It occurs when there are two or more instances in the ontology with the same formal definition, that is, the only difference between them is the name.

(2) Completeness

Completeness means all that is supposed to be in the ontology is explicitly represented in it or can be inferred using other definitions. That is, the constructed ontology should avoid the following errors:

For all ontology constructing approaches, the most important thing is to ensure that the constructed ontology is correct. In the above-mentioned “evaluating ontology” step, DODO mechanism referred many traditional approaches and enumerate possible errors occurred in the construction process. In its implementation is SATOR, a semi-automatic tool is provided to help domain experts localize and solve these problems.

(3) Cost

The high cost of build-by-integrating-others approaches makes them fail to carry the burden of constructing ontology for the VO. Compared to these approaches, DODO alleviate the cost because in fact it is a learning method and doesn’t aim at integrate two different ontologies. Through matching technologies, DODO will discover and only learn knowledge about focused resources. Therefore the cost of DODO is much lower than build-by-integrating-others approaches.

8. Conclusion

This paper proposes a mechanism for dynamically constructing ontology for a VO. This mechanism doesn’t require that the construction of ontology should be finished before applications. It stresses the consummation of ontology during the applications. This mechanism makes the constructed ontology flexible and evolvable, which can enable the resource sharing in the running of VO.

In the future, our works mainly focus on how to resolve the contradictions semi-automatically. Today, in the implementation of DODO mechanism, the solving of contradictions is mainly dependant on domain experts and machines only recommend the plans. To further relieve the burden of domain experts, we plan to analyze each kind of contradiction in detail and let machine finish some of the tasks. Domain experts just need to verify the final results.

9. REFERENCES


