Distributed Ontology Development Environment for Multi-agent Systems

Motoyuki Takaai, Hideaki Takeda and Toyoaki Nishida
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-01, Japan
{motoyu-t,takeda,nishida}@is.aist-nara.ac.jp

Abstract

In this paper, we propose a new environment that supports distributed ontology development for multi-agent systems. In our distributed ontology development environment (Donden), users to program agents can build ontologies locally which can be associated to each other. Donden consists ontology browser which is provided for each user and ontology server which is provided for a group of users which want to share ontologies. Ontology browser helps user to edit ontology and to make links to other user's ontology by graphical user interface. Ontology server helps users to synthesize ontologies by computing similarity between concepts of ontologies. We also show another application of distributed ontology development which supports distributed engineering information base.

Introduction

In recent years, researchers are interested in problem solving and knowledge information processing by multi-agent systems. In multi-agent systems, a number of computer programs (so called agents) are connected to each other, so that larger and more complex problems can be solved.

The most important problem of multi-agent systems is how to use the concepts in different description and conceptualization together. When two agents with different descriptions of concepts try to communicate each other, difference of descriptions disturbs their communication.

Ontology(Patil *et al.* 1992) is an answer for this problem. An ontology is a specification of a conceptualization. Agents with a common ontology can share their knowledge and work cooperatively by common descriptions of concepts in the ontology. Generally, ontology is large and have many ways to describe concepts, so that ontology is difficult to build and to keep consistency.

In this paper, we discuss the way of building ontology and propose a distributed ontology development environment for ontology builders.

There are two ways of building ontology for multi-agent systems. The first approach is by bottom-up method, i.e., ontology is build along agent building. Ontology inconsistencies are removed at the same time or after building. The second is by top-down method, i.e., ontology is build before agent building.

KC-Kansai (Nishida & Takeda 1993) is a testbed of knowledge sharing and reuse for very large knowledge base. Ontology of KC-Kansai was built by the first approach. In PACT (Cutkosky *et al.* 1993) took the same approach. In this approach, users can build ontology which is convenient for their agents. But they must build the ontology cooperatively in order to avoid inconsistency in ontology.

Geographically distributed users are building ontologies in collaboration through WWW (Farquhar *et al.* 1995). It takes the second approach. The ontologies each of which is built separately from agent building may be consistent, but not convenient for agent builders because relations among them are not clear.

In this paper we discuss how to realize the bottom-up approach for ontology building and propose a distributed ontology development environment called Donden. Donden supports building ontologies as follows: 1) to edit ontology and to make links to other user's ontology by graphical user interface. 2) to synthesize ontologies by computing similarity between concepts of ontologies.

In section 2, we show Donden architecture. In section 3, we discuss how to reflect many aspects of a concept on ontology descriptions. In section 4, we show how Donden supports distributed ontology development. In section 5, we show experiments of the integration among ontologies. In section 6, we show another application of distributed ontology development which supports distributed engineering information base. Finally we summarize the paper in the last section.

Donden: a distributed ontology development environment

Figure 1 is the architecture of Donden. Donden consists of ontology browser which is provided for

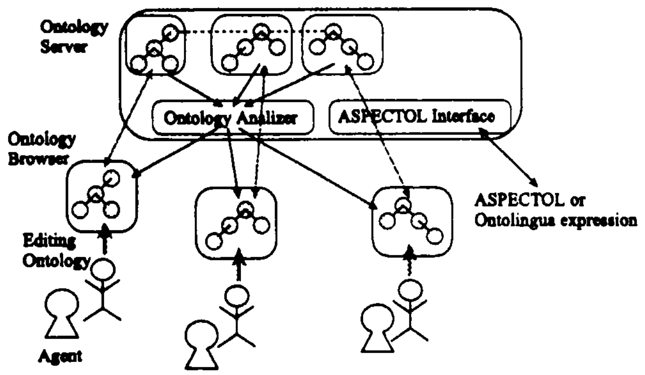


Figure 1: the architecture of Donden

each user and ontology server which is provided for a group of users who want to share ontologies. Ontology browser helps user to edit ontology and to make links to other user's ontology by graphical user interface. Ontology server manages multiple ontologies at the same time by using an ontology description language ASPECTOL. Ontology server helps users to synthesize ontologies by computing similarity between concepts of ontologies. Ontology server and ontology browser are realized as the KQML(Finin *et al.* 1992) agents.

Frame ontology of Ontolingua

Ontolingua is an ontology description language which was made by KSE (Knowledge Sharing Effort) of DARPA. With Ontolingua, we can write declarative frame expression. We can define classes, relations, functions, and instances by using Ontolingua primitives as define-class, define-relation, define-function, and define-instance.

Aspect

Generally, a concept has different description. To share the concept among systems, we must use a common description for the concept. But different systems usually have different descriptions for a concept because of differences of their purpose and a point of view. It may disturb agents to understand each other. Aspect is a framework which can manage such different descriptions of concepts. Ontology is a set of the concept units called aspect. Different aspects for a concept are different expression for it. And translation rules between aspects make agents to share knowledge.

Figure 2 and Figure 3 are the example expression of the rail and taxi fare system. The description after `:def` is a necessary condition for the instance (shown by `?fare`) into the class. In Figure 2, the instance is defined as a thing having one adult-fare slot (the type is `Basic:money`) and one child-fare slot (the type is also `Basic:money`).

Figure 4 shows a example of the translation rule from taxi fare to rail fare. The `taxi:fare` slot of `?taxi-fare` is translated to the `rail:adult-fare` slot of `?rail-fare` and the `rail:child-fare` slot of `?rail-fare`.

```
(define-class rail:fare (?fare)
 :def
  (and
   (has-one ?fare rail:adult-fare)
   (value-type ?fare rail:adult-fare Basic:money)
   (has-one ?fare rail:child-fare)
   (value-type ?fare rail:child-fare Basic:money)))
```

Figure 2: the fare class of rail way

```
(define-class taxi:fare (?fare)
 :def
  (and
   (has-one ?fare taxi:fare)
   (value-type ?fare taxi:fare Basic:money)))
```

Figure 3: the fare class of taxi

Editing ontology by Donden

Donden provides Ontology Browser for each user and Ontology Server for a group of users who want to share ontologies.

Ontology Server has following functions.

1. Management of users
2. Management of ontology for each ontology builder
3. Support of the integration of concepts for ontology builders.
4. Translation between expression inside Ontology Server and Ontolingua.

Ontology Browser exchanges information of ontologies with Ontology Server and realizes the following functions.

1. To indicate class and class-instance relations by using graph and make possible to edit ontology visually.
2. To show and take the ontology of other ontology builder in.

We implemented the prototype of Ontology Browser and Ontology Server by using scheme interpreter, STk.

```
(define-translation category-of-fare
 (=> (taxi:fare ?taxi-fare)
     (rail:fare ?rail-fare))
 ((-> (taxi:fare ?taxi-fare ?fare)
      (and (rail:adult-fare ?rail-fare ?fare)
           (rail:child-fare ?rail-fare ?fare)))))
```

Figure 4: a translation rule from a taxi fare to a rail fare

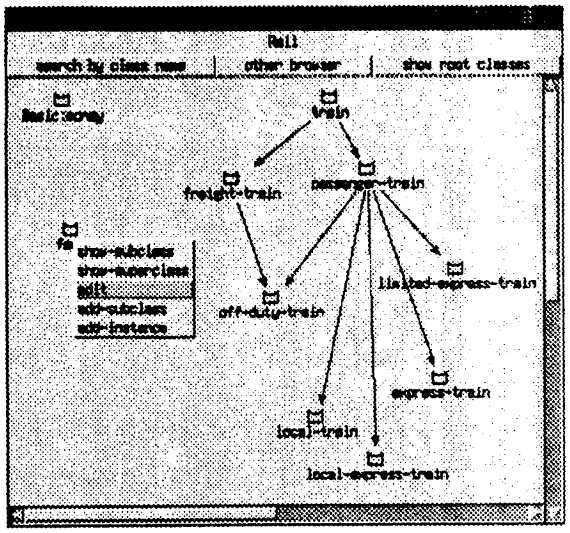
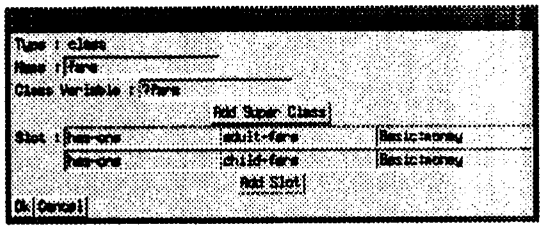


Figure 5: Ontology Browser



Type : class
Name : fare
Class Variable : ?fare
Slot : has-one adult-fare Basic:money
has-one child-fare Basic:money

Figure 6: Editing class by Ontology Browser

Figure 5 shows that a user of a rail road agent makes ontology by using Ontology Browser. In this figure, there are class hierarchy about train, class about train, class about the money which is brought in from Basic ontology builder. Edit is chosen in the pop-up menu of fare class in this screen.

Figure 5 shows the scene editing the fare actually. Here, we can edit class name, slot and so on.

Ontology Server translates the class of fare into Ontolingua in Figure 2.

Support of ontology integration

Donden supports ontology integration by collecting similar expression of concepts and showing them to the ontology builder. We employ hierarchical cluster analyses as the method of collecting similar classes and multidimensional scaling (TypeIV quantification method) as the method of arranging the classes on a plane.

Method of calculating similarity of concepts

Mainly there are following relations between classes of similar expression.

Table 1: Clusters of a concept for travel

(1)	timepoint:universal-time-spec timepoint:long-time-spec timepoint:calendar-date timepoint:calendar-year
(2)	hotelguide:hotel guesthouse:guesthouse hotel-with-building-information:hotel business-hotel:hotel generic-hotel:hotel
(3)	overnight-with-two-meals:hotel-charge overnight-with-breakfast:hotel-charge overnight-without-meals:hotel-charge lowest-highest-room:hotel-charge
(4)	move-with-traffic:transfer move-with-place:destination

1. same concepts of different expressions
2. similar expressions of different concepts (e.g. super - subclass relation and so on)

The example of Figure 2 and Figure 3 have the same name fare, but since the are not showing the same concepts, these examples belong to (2). To count the similarity of classes we use the following things:

1. the name of class
2. the name of super class and sub class
3. the name of slot and the type of slot
4. the instance that belongs to the class
5. the relations and the functions which use the class
6. the document of the class

Donden calculates the similarity of classes by using this information and shows the possibility of concept integration to ontology builder.

We performed an experiment for integration of class expression. In this case the calculation of the similarity is a sum of the name similarity of each slot combination. The other types of information are omitted. The name similarity is

- 1.0 if they are same name.
- 0.5 if one name includes the other.
- 0.0 else case.

Collecting similar concepts by hierarchical cluster analyses

Table 1 is the result of the experience of hierarchical cluster analyses with the method of calculating similarity. The ontology for this experiment is a set of aspects for travel concepts e.g. time, hotel and sightseeing place. It is written in Ontolingua (Iino 1995; Gruber) . This table shows higher four clusters. The cluster (1) is all about time, and the cluster (2) is all about hotel. We can collect similar concepts from the similarity of slot expression.

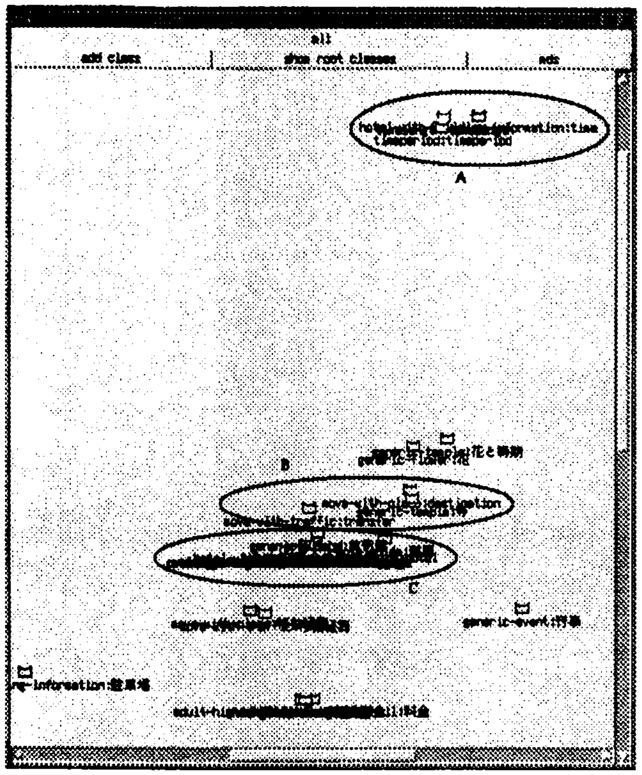


Figure 7: Classes arranged by multidimensional scaling

Arranging the concepts by multidimensional scaling

Figure 7 is the result of the experience of multidimensional scaling from the same ontology and the same method of calculating similarity with hierarchical cluster analyses. We can find some groups from it. The groups are classified three types as follows:

1. the group of same or similar concepts. For example, the four classes enclosed oval A on Figure 7 are all about time.
2. the group including different concepts which have same property concept. For example, the three classes enclosed oval B on Figure 7 are "transfer", "destination" and "temple" which have the transportation slot.
3. the others. Sometimes, we can find some groups in the group by zooming in. For example, the twelve classes enclosed oval C on Figure 7 consists three groups (five classes about hotel, six classes about hotel-charge and one class about museum.

Sharing engineering knowledge by distributed ontology development environment

We adopt the distributed ontology development environment to share engineering knowledge among engineers.

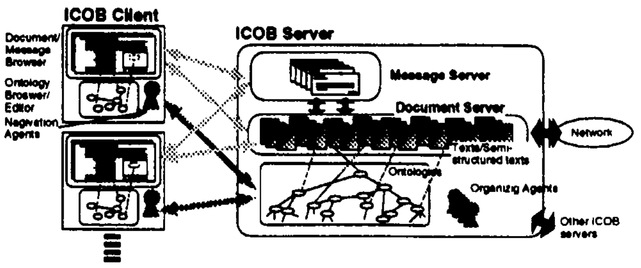


Figure 8: Architecture for Intelligent Corporate Base

In this environment, ontologies are developed by bottom-up method along development of engineering knowledge and role the bridge between multiple users.

Figure 8 shows an architecture for engineering knowledge and communication bases called ICoB (Intelligent Corporate Base) which is based on distributed ontology development environment.

There are servers which contains shared documents and communication messages, and clients each of which an engineer uses. Users can retrieve or submit documents or communication messages by using shared and private ontologies. The ICoB servers organize documents and messages by using ontologies which consist of shared and users' ontologies. At the same time, they can extend and their ontologies by referring and comparing shared and other private ontologies. The latter process corresponds organization of information we discussed in the previous section. ICoB clients and servers can have some facilities to assist users' information organization.

Conclusion and further work

By managing ontology based on Aspect, Donden supports distributed ontology development. In this approach, geographically distributed users can build ontology which is convenient for their agents, and can reuse other agent systems with different ontologies.

In the current implementation, only two integration methods are supported. Both methods mainly concern with semantics of ontologies. We are planning to provide various kinds of integration methods for examples, the syntactic approach to calculate similarity shown in (Nishida, Koujitani, & Takeda 1995) can work with our semantic approach together.

References

Cutkosky, M. R.; Engelmore, R. S.; Fikes, R. E.; Genesereth, M. R.; Gruber, T. R.; Mark, W. S.; Tenenbaum, J. M.; and Weber, J. C. 1993. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer* January 1993:28–38.

Farquhar, A.; Fikes, R.; Pratt, W.; and Rice., J. 1995. Collarborative ontology construction for information integration. Technical Report KSL 95-63, Knowledge Systems Laboratory, Stanford University.

Finin, T.; Weber, J.; Wiederhold, G.; Genesereth, M.; Fritzson, R.; McKay, D.; McGuire, J.; Pelavin, P.; Shapiro, S.; and Beck, C. 1992. Specification of the KQML agent-communication language. Technical Report EIT TR 92-04, Enterprise Integration Technologies. (Updated July 1993).

Gruber, T. Sharable ontology library.

Iino, K. 1995. Knowledge sharing with multiple ontologies. Master's Thesis.

Nishida, T., and Takeda, H. 1993. Towards the knowledgeable community. In *Proceedings of International Conference on Building and Sharing of Very Large-Scale Knowledge bases '93*, 157–166. Tokyo: Japan Information Processing Development Center.

Nishida, T.; Koujitani, K.; and Takeda, H. 1995. A plain indexing method for organizing conceptually promiscuous data. In *1995 AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, 103–109.

Patil, R. S.; Fikes, R. E.; Patel-Schneider, P. F.; McKay, D.; Finin, T.; Gruber, T. R.; and Neches, R. 1992. The DARPA knowledge sharing effort: Progress report. In Rich, C.; Nebel, B.; and Swartout, W., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*. Morgan Kaufmann.