ONTOLOGY MODELING TOOL USING CONCEPT DICTIONARY AND INFERENCE

Yoichi Hiramatsu
Galaxy Express Corporation
1-18-16 Hamamatsucho, Minatoku,
Tokyo 105-0013, Japan

Keywords: Ontology Modeling, Editing Tool, Concept Dictionary, EDR, Lexical Dictionary, WordNet, Inference Algorithm, Web service, Enterprise System, Common Lisp, I-S-A semantic link, Operation Support System

Abstract: The usefulness of ontology is strongly dependent on a knowledge representation policy and its maintenance. The subject of knowledge representation and modeling tools has been one of the most exciting themes among ontology scientists. Some ontology editing tools originated and developed in the field of expert systems, and others were designed originally by ontology research groups. Key features of the newly implemented tool are: (a) reference to a concept dictionary (EDR, WordNet) to ascertain word semantics, and (b) use of an inference algorithm (MOP) provided by Schank et al. Satisfactory results were obtained in the application of ontology modeled by the present tool. We will discuss how our tool was constructed and describe applications using the tool to achieve solutions for enterprise integration. This work was developed as a part of the project entitled "Operation-support system for large-scale system using information technology" (Koide et al., 2003) for the Japanese Government IT Program, period 2002-2005.

1 INTRODUCTION

One of the design goals of the ontology modeling tool is to provide an interactive and graphical facility for constructing ontology representation files. A graphical user interface (GUI) should work as an advisory system wizard providing developer guidance during the construction of class hierarchies and relationships. This paper deals with one aspect of these goals: that is, implementation of a modeling tool (named "MOP Editor") connected to a concept dictionary and to an inference algorithm. Despite the existence of attractive tools to model ontologies, we have persisted in developing the necessary tool for our specific requirements.

The category of modeling tools described herein should gain considerable importance for the concretization of novel software architecture such as, for example, those which model and generate business-to-business (B2B) applications supporting decentralized and dynamic electronic agents (Alloui et al., 2003).

If we find a way to access an external source of reference to determine ontology, we can save time during the editing phase. We have decided to use an online concept dictionary (EDR, 2003) and a lexical dictionary (WordNet, 2003) to refer to for concepts of Japanese and English terms. And, if we devise a way to interactively test functionality of the "in-construction" ontology during the construction phase, the ontology quality is improved significantly in comparison with that obtained without interactive tests. Thus, we have chosen an inference engine known as MOP (Memory Organization Package) algorithm introduced by Schank et al. (Riesbeck et al., 1989; Schank et al., 1994) in order to execute ontology evaluation tests. MOP is a kind of frame-based knowledge formalism that holds class hierarchy, in which concepts and instances are represented by *mop* objects.

The effectiveness of MOP Editor and the constructed ontology were evaluated against objective design criteria using three kinds of Web services: message generation, synonym retrieval, and ontology sharing. Web services are distributed around the system and work collaboratively.

2 ONTOLOGY AND DOMAIN FEATURES

Ontology is a set of conceptual building blocks of

Hiramatsu Y. (2004).
ONTOLOGY MODELING TOOL USING CONCEPT DICTIONARY AND INFERENCE.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 225-230
DOI: 10.5220/0002603002250230

knowledge in a determined application area. It provides a domain of discourse for knowledge sharing among computers. Ontology enables a number of machines to share their knowledge: for example, for information retrieval systems. Concretely, ontology enumerates concepts, attributes of concepts, relationships among concepts, and constraints on these relationships, thereby defining the knowledge reference structure of the considered domain.

Our domain is *ground support equipments for rocket launching* at a rocket launch facility at Tanegashima Space Center in Japan. Entry terms of the domain are the names of the following objects: fluid-pump, air-compressor, pipe, junction, storage-device, electrical-cable, fuel-tank, oxygen-tank, valve, actuator, fuel, oxygen, vaporizer, sensor, pressure-controller, etc.

Abstract concepts that correspond to the entry terms are arranged in such a way as to compose a class hierarchy in which superclass and class are connected by an I-S-A semantic link. For example, "pressure-controller is-a-kind-of controller". Ontology can be accurately represented by the class hierarchy of abstract concepts plus actual instances. Instances are tagged names of real objects, such as sensor-AA, controller-101, controller-102, and so on. Instances are attached to ending points of the leaf of the class hierarchy. In general, the number of instances is estimated to be one order greater than the number of classes. In our present work, we have selected a quantity of over a hundred for classes and over a thousand for instances.

3 CURRENT ONTOLOGY MODELING TECNIQUES

Free software tools and commercial products for ontology modeling have evolved during the last few years. Some examples are Ontolingua, Protégé2000, OntoEdit and Hozo. These tools accept entry terms and create an I-S-A semantic link for classes, generating text-type files of class hierarchy. Most of them are equipped with plug-in features for importing other ontologies written in XML, RDF, DAML, OWL, etc.

Ontolingua (Farquhar et al., 1996; Ontolingua, 2003) is a set of tools and services that supports the process of achieving consensus on common shared ontologies by geographically distributed groups. Ontolingua makes use of the world-wide web to enable wide access and provide developers with the ability to publish, browse, create, and edit ontologies stored on an ontology server.

On the other hand, the Protégé system (Gennari et al., 2002; Noy et al., 2001; Noy and Musen, 2003) is an environment for building knowledge-based systems, mainly in the field of domain ontology. Protégé runs on a variety of platforms, supports customized user interface extensions, incorporates the Open Knowledge Base Connectivity (OKBC) knowledge model, and interacts with storage formats such as relational database, XML and RDF.

Similarly, OntoEdit (Sure et al., 2002; OntoEdit, 2003; Maier et al., 2003) is another ontology editing environment that supports the development and maintenance of ontologies by graphical means. OntoEdit is built on top of an internal ontology model, enabling therefore as much neutral modeling as possible for concepts, relations and constraints. It is equipped with a GUI menu entry (in which developers can choose namespace) and a back-end inference engine.

Another interesting tool that we have investigated is Hozo (Kozaki et al., 2002; Kozaki et al., 2000). This has different features than Ontolingua, Protégé or OntoEdit. Hozo is based on an ontology theory of role-concept in which the tool can distinguish concepts dependent on particular contexts from the so-called basic concept, and can manage the correspondence between a wholeness concept and a relatioship concept.

The newly implemented MOP Editor differs from the modeling tools described above in the following points: (a) it is coded in Common Lisp (CL) and Common Lisp Object System (CLOS), thereby allowing capability for dynamic maintenance, (b) it makes reference to the concept dictionary to help developers to build as general an ontological model as possible, and (c) it supports the inference engine that will be used to check the appropriateness of the built ontology.

4 IMPLEMENTED MODELING TOOL

4.1 Configuration of Tool

Figure 1 is a process view of the MOP Editor. The GUI facility was implemented using the Integrated Development Environment (IDE-CG) of the CL. The MOP algorithm was developed using a set of functions of CLOS (Koide and Kitamura, 2002). Although there are a lot of respectable programming languages available, we think that CL is, in a sense, the mother of all languages and is highly efficient. The main reason for this statement is that CL includes a complete theory of computation by treating code and data within a single and uniform system.

Link characterization among concepts is based on the I-S-A (*is-a-kind-of*) semantic link, as adopted in EDR and WordNet.

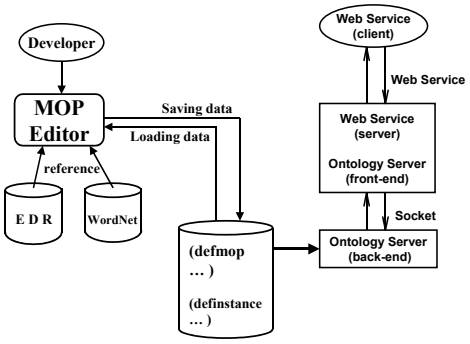

Figure 1: Process view of MOP Editor

4.2 Class Hierarchy

The class hierarchy mentioned here is a *classification hierarchy*. Concepts are connected by an *is-a-kind-of* relationship. This relationship can be seen as the following: if X is a kind of Y, then X is a specialization of Y, while Y is a generalization of X. Therefore, our ontological representation is based on the *classification hierarchy*.

4.3 Data Organization

MOP Editor organizes the developer's data in a form of project unit (file named *.mprj). Each project contains all the information that developers have typed in, that is; the ontology file (named *.mont) and the instances file (named *.mins). The ontology file and instance file are composed of Lisp S-expressions for *mop* objects and *mop* instances, respectively. The following are examples of ontology files and instance files:

<ontology file>
(defmop *mop-object* (*superclass* *superclass* …)
(*role1* *filler1*)
(*role2* *filler2*)
(*role3* *filler3*)
(*role4* *filler4*)
…)

<instance file>
(definstance *mop-instance* (*superclass*)
(*has-a* *mop*)
(*part-of* *mop*)
…)

The schematic shown in Figure 2 clarifies the input and output files of MOP Editor. Projects file *.mprj can be loaded dynamically by developers during ontology maintenance services, without any necessity for system shutdown.

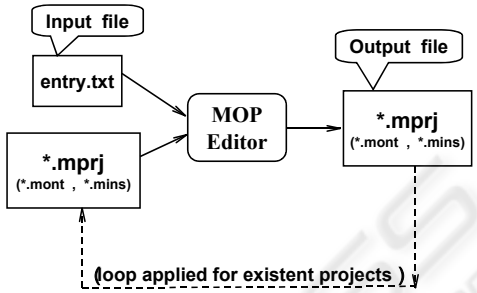

Figure 2: Schematic of Input/Output

4.4 Utilization of Concept Dictionary

MOP Editor utilizes two kinds of online dictionaries as a reference source to build the top, domain and task ontologies. The flowchart in Figure 3 shows how developers use the parser software and dictionaries to add new concepts. Developers can switch between the two dictionaries according to their needs when looking for new concepts.

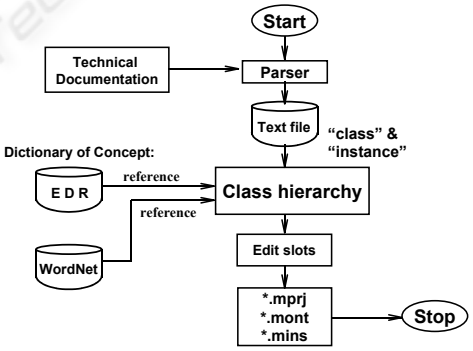

Figure 3: Flowchart

EDR

EDR is a machine tractable dictionary of Japanese words used in the research field of natural language processing. The words resource is organized in the form of data records. The main feature of EDR is that its sub-dictionaries are interconnected via concepts using concept ID tags.

In this work, we have used three sub-dictionaries of EDR: (a) a Japanese Word Dictionary, (b) a Concept Classification Dictionary, and (c) a Headconcept Dictionary. The role of the Japanese Word Dictionary is to describe the correspondence

between Japanese words and the concepts represented by these words, and also to provide grammatical information for the word when used with a given meaning. Each record is composed of the following fields: record number, headword, invariable portion of the headword and pair of adjacent attributes, Kana notation, pronunciation, syntactic category (noun, verb and particle), syntactic tree, conjugational information, surface case information, aspect information, word function information, concept ID, English headconcept, Japanese headconcept, English concept explanation, Japanese concept explanation, usage, frequency, and management information.

Thus, the purpose of the Concept Classification Dictionary and the Headconcept Dictionary is to provide concepts for the Japanese Word Dictionary. The first contains a classification of concepts having a super-sub relation, and the data record is composed of the following fields: record number, concept ID of the superconcept, concept ID of the subconcept, and management information. The second one gives a description of each concept in words, and the record is composed of the following fields: record number, concept ID, English headconcept, Japanese headconcept, English concept explanation, Japanese concept explanation, and management information. Both dictionaries contain about 400,000 concepts.

MOP Editor uses a concept ID to connect concepts among these sub-dictionaries. The ontology developer would display the superclass and class of concepts when he is looking for a relationship that can better represent the subsumption relationship (or I-S-A semantic link) in the treated domain.

WordNet

WordNet is an online lexical resource of English language used by linguistic scientists. WordNet consists of lexicographer files that organize nouns, verbs, adjectives and adverbs into groups of synonyms called synsets (synonym sets) and relationships between this synset and other synsets.

Nouns and verbs are organized into a hierarchy based on the hypernymy/hyponymy and holonymy/meronymy relationships between synsets. Here, hypernymy and hyponymy mean, respectively, superordinate and subordinate relations. For example, Y is a hypernym of X, if X is a (kind of) Y. Holonym is the name of the whole of which the meronym names a part. For example, Y is a holonym of X, if X is a part of Y. WordNet shows how each word is linked to others, as for example when the developer types in the word *valve***,** he gets not only the synonyms definition, but the hypernyms (a *valve* is a kind of what?), meronyms (what are the parts of a *valve*?), and more.

4.5 Behavior of Inference Engine

MOP is a frame-based formalism based on the Schank's theory about how human memory is organized, and attempts to remind computers the way people are reminded.

In MOP algorithm, we use a global symbol named *mop* to represent a CLOS object. The *mop* object can have multiple superclasses and a set of attributes represented by slots, such as (role1 filler1), (role2 filler2), (role3 filler3), ... The role is a slot-name and the filler is a slot-value. The filler takes values of the type string, integer, double-float and own *mop* object. The class hierarchy is a representation of abstract concepts using *mop* objects. When developers create a new *mop* object, MOP Editor evaluates this *mop* object in order to avoid eventual definition errors of the *mop* object. MOP Editor still checks whether the attributes represented by the pair of role and filler are consistent inside the ontology file and instances file.

A new *mop* object is created if all the fillers have been approved through this error checking evaluation test. Note that if the filler already exists, then the previous defined filler is used for the evaluation test. On the other hand, if the filler does not exist, then the evaluation test uses the filler of the present *mop* object. Therefore, MOP algorithm differs from the frame-based system in the following points: (a) the newly created filler behaves as a *mop* object, (b) the filler is also structured hierarchically, (c) *mop* hierarchy is flexible, i.e., the knowledge representation becomes more and more detailed every time new *mop* objects are added to the class hierarchy. MOP uses *slots* to place the new instance at an appropriate location inside the class hierarchy. Figure 4 shows a view of the implemented MOP Editor.

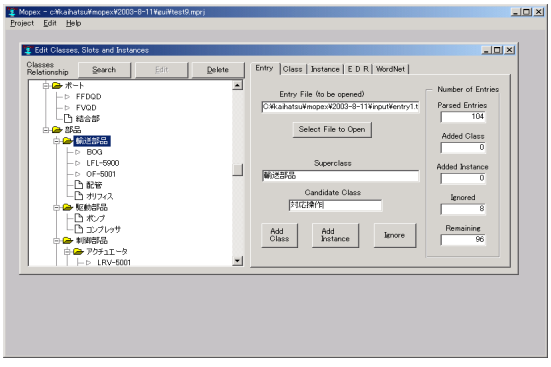

Figure 4: A view of MOP Editor

5 EVALUATION OF TOOL AND ONTOLOGY

Functionality of MOP Editor and appropriateness of the constructed ontology were evaluated using three kinds of Web services: message generation service, synonym retrieval service, and ontology sharing service. Web service is a software system designed to support interoperable machine-to-machine interaction in a distributed and collaborative environment, equipped with an interface described in a machine-processable format under the SOAP protocol.

5.1 Message Generation Service

Message generation is one of the simplest Web services. The Web client of a message generation service queries for generation of messages comprehensive to humans by sending a set of keywords in direction to the ontology server. The ontology server uses an adequate template and sends a reply message to the client. The role of the ontology is to provide a reference base of information retrieval to compose grammatically correct messages that better fit the client queries.

For example, for a given set of keywords such as: part = *controller*, device = *LNG drain control valve* and event = *abnormal closing*, the generated message would be "*Because of malfunction of the controller at LNG drain control valve, abnormal closing was detected*".

This message is sent back to the Web client. As shown in Figure 5a and Figure 5b, we confirmed that our ontology is capable of generating messages comprehensive to humans for several contextual situations in the range of computer simulation.

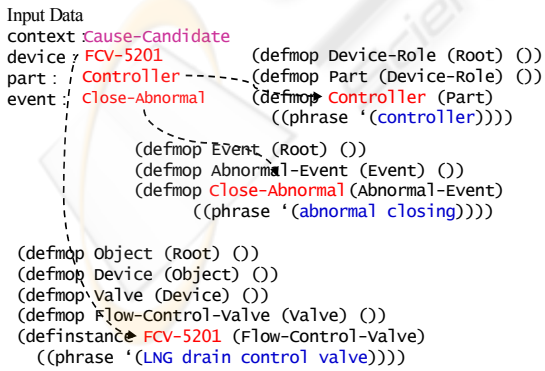

Figure 5a: Message Generation

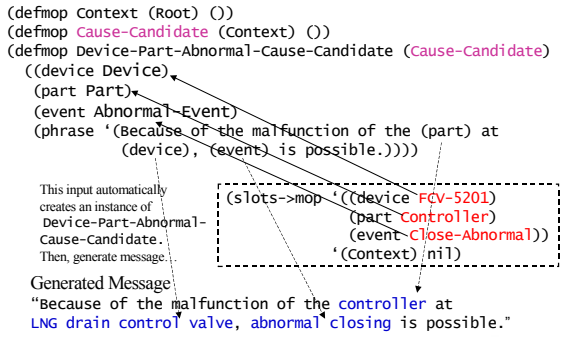

Figure 5b: Message Generation

5.2 Synonym Retrieval Service

Synonym retrieval is a Web service for information retrieval. The Web client queries for a synonym that has the same or similar meaning as a given word; for example, for *liquid-oxygen* the synonym would be *oxidant*, for *temperature-sensor* the synonym is *thermocouple* or *thermometer*, and so on. Furthermore, in specialized fields engineers use different technical terms to indicate identical or equivalent physical objects, such as *fuel-feeding-system* and *launching-tower*, or even to differentiate one particular physical hazard from another more general one, such as *fatigue-crack* and *crack*. Since operational tasks for rocket launching comprise a large-scale system embracing hundred of engineers of different divisions and companies, a Web-searching service for adequate synonyms gains significant importance during diagnostics.

5.3 Ontology Sharing Service

Use of a fixed source of reference such as EDR or WordNet makes the built ontology transferable to other database servers distributed on the network. Thus, sharing of ontology becomes possible because I-S-A link and hypernymy/hyponymy or holonymy/meronymy relationships can be considered "invariable" during a long period of time.

The Web client of an ontology sharing service queries for a copy of one part or a whole ontology stored in the ontology server. A copy of class hierarchy will be shared with other data servers. In this work, we have tried a sharing service between an ontology server and a so-called large-scale multimedia filing data server. A sharing service is used for a rapid keyword-based search of multimedia data (documentation, drawings, logs, sound, pictures and movies) during operational tasks and diagnostics at the rocket launch facility.

6 SUMMARY AND FUTURE WORK

MOP Editor demonstrated ability in constructing ontology applicable to Web services for message generation, synonym retrieval, and ontology sharing. We have confirmed throughout our computer simulation that the tool can be used for graphically modeling class hierarchy and semantic links. Furthermore, we have assured that ontological models provided by EDR and WordNet work relatively well for the presented Web services.

Future work is needed to enhance GUI facility and MOP functionality to incorporate coming themes such as text extraction from any sources, task ontology, ontology distinguishing, integration of different ontologies, and plug-in for database connectivity.

ACKNOWLEDGEMENT

This work has been supported in part by the Ministry of Education, Culture, Sports and Technology (MEXT) of Japan. The authors are grateful to Professor Dr. R. Mizoguchi and Research Associate Dr. K. Kozaki of Osaka University for providing helpful comments on ontology methodology. Many thanks to the technical group of Franz Incorporated for valuable advice on Allegro's IDE-CG. Many thanks also to all the colleagues of Galaxy Express Corporation for their enthusiasm and contributions in developing the project.

REFERENCES

Alloui, I., Megzari, K., Oquendo, F., 2003. Modeling and generating business-to-business applications using an architecture description language-based approach. In *ICEIS-2003, 5th International Conference on Enterprise Information Systems*. ICEIS Press.

EDR (2003)
http://www.jsa.co.jp/EDR/
Communications Research Laboratory. Web site.

Farquhar, A., Fikes, R., Rice, J., 1996. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *Knowledge Systems Laboratory*, Stanford University.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., Tu, S. W., 2002. The evolution of Protégé: an environment for knowledge-based systems development. In *International Journal of Human-Computer Interaction*.

Koide, S., Kitamura, Y., 2002. MOP3: Memory Organization Package on Meta Object Protocol. In *ILC-2002, International Lisp Conference,* San Francisco, USA.

Koide, S., Nishio, H., Kitamura, Y., Gofuku, A., Mizoguchi, R., 2003. Operation-support system for large-scale system using information technology. In *ICEIS-2003, 5th International Conference on Enterprise Information Systems*. ICEIS Press.

Kozaki, K., Kitamura, Y., Ikeda, M., Mizoguchi, R., 2000. Development of an environment for building ontologies which is based on a fundamental consideration of "relationship" and "role". In *PKAW2000, Proceedings of the Sixth Pacific Knowledge Acquisition Workshop*.

Kozaki, K., Kitamura, Y., Ikeda, M., Mizoguchi, R., 2002. Hozo: an environment for building/using ontologies based on a fundamental consideration of role and relationship. In *EKAW2002, 13th International Conference Knowledge Engineering and Knowledge Management*.

Maier, A., Schnurr, H., Sure, Y., 2003. Ontology-Based Information Integration in the Automotive Industry. In ISWC-2003, 2nd International Semantic Web Conference.

Noy, N. F. and McGuinness, D. L., 2001. Ontology Development 101: a guide to creating your first ontology. In *SMI technical report (SMI-2001-0880)*, Stanford University.

Noy, N. F. and Musen, M. A., 2003. Ontology versioning as an element of an ontology-management framework. In *IEEE Intelligent Systems*.

Riesbeck , C. K. , Schank, R. C., 1989. *Inside case-based reasoning*. Lawrence Erlbaum Associates, Publishers. New Jersey.

Schank, R. C., Kass, A., Riesbeck, C. K., 1994. *Inside case-based explanation*. Lawrence Erlbaum Associates, Publishers. New Jersey.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., Wenke, D., 2002. OntoEdit: collaborative ontology engineering for the semantic Web. In *ISWC-2002, 1st International Semantic Web Conference*.

OntoEdit (2003)
http://www.ontoprise.de
Ontoprise GmbH. Web site.

Ontolingua (2003)
http://www-ksl-svc.stanford.edu/
http://ontolingua.stanford.edu/
Knowledge Systems Laboratory, Stanford University. Web site.

WordNet (2003)
http://www.cogsci.princeton.edu/~wn/
ftp://ftp.cogsci.princeton.edu
Cognitive Science Laboratory at Princeton University. Web site.