Hozo: an Ontology Development Environment -Treatment of "Role Concept" and Dependency Management -

Kouji Kozaki, Eiichi Sunagawa, Yoshinobu Kitamura, and Riichiro Mizoguchi
The Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567 -0047 Japan
{kozaki, sunagawa, kita, miz}@ei.sanken.osaka-u.ac.jp

Abstract

We have developed an environment for building/using ontologies, named Hozo, based on both of a fundamental consideration of an ontological theory and a methodology of building an ontology. This paper presents an outline of the functionality of Hozo.

1 Introduction

Ontology is one of key technology to realize Semantic Web. It is, however, difficult to develop a well-organized ontology because the principles of ontology design are not clear enough. Therefore, a methodology for ontology design and a computer system supporting for ontology design are needed. Our research goal is development of a methodology for ontology design and supporting environment based on the methodology. And we aim to develop ontology-based applications using the system.

2 Hozo

Many researchers have discussed ontological theories and methodologies for building ontologies. However, there are not many tools which reflect the results of these research results. We have developed an environment for building/using ontologies, named Hozo, based on both of a fundamental consideration of an ontological theory and a methodology of building an ontology. The features of Hozo include 1) Supporting to role concepts, 2) Visualization of ontologies in well considered format, and 3) Distributed development based on management of dependencies between ontologies.

2.1 Architecture of Hozo

Hozo is composed of four modules: Ontology Editor, Ontology Manager, Ontology Server and Onto-Studio (Fig.1). ***Ontology Editor*** provides users with a graphical interface (Fig.2), through which they can browse and modify ontologies by simple mouse operations. Its feature is the functionality to treat "role concept" and "relation" on the basis of fundamental consideration [Kozaki 02]. ***Ontology Manager*** helps users for distributed ontology development. It manages ontologies as components of a target ontology based on their dependencies. And when a change of some ontology influences others, it supports modification of influenced ontology for keeping its consistency [Sunagawa 03]. ***Onto-Studio*** is based on a method of building ontologies, named AFM (Activity-First Method). It helps users design an ontology from technical documents. It consists of 4 phases and 12 steps [kozaki 02]. ***Ontology Server*** manages the storage and use of ontologies and models. Models are built by choosing and instantiating concepts in the ontology and by connecting the instances by defining specific relation among them. Hozo also checks the consistency of the model using the axioms defined in the ontology.

The latest version of this ontology editor is published at the URL: http://www.hozo.jp.

2.2 Treatment of Role Concept

Since Hozo is based on an ontological theory of a role-concept, it can distinguish concepts dependent on particular contexts from so-called basic concepts and contribute to building reusable ontologies. Based on the theory, we identified three categories for a concept. That is, a basic concept, a role-concept, and a role holder. A ***role-concept*** represents a role which a thing, usually a basic concept, plays in a specific context and it is defined with other concepts. On the other hand, a ***basic concept*** does not need other concepts for being defined. An instance of a basic concept that plays a role such as husband role or wife role is called a ***role holder***. For example, a bicycle has a wheel which plays the front wheel role and the steering role. A wheel that plays these roles is called "a front wheel" and/or "a steering wheel", respectively, which are role holders.

In ontological theory, the role concept is very important and discussed by many researchers [Sowa 95, Guarino 98].

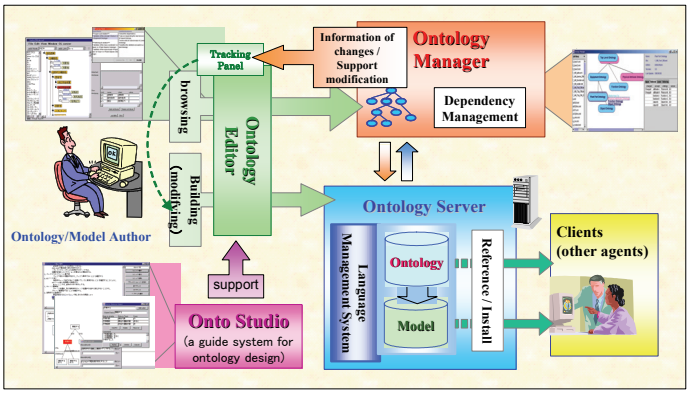

Fig.1. Architecture of Hozo

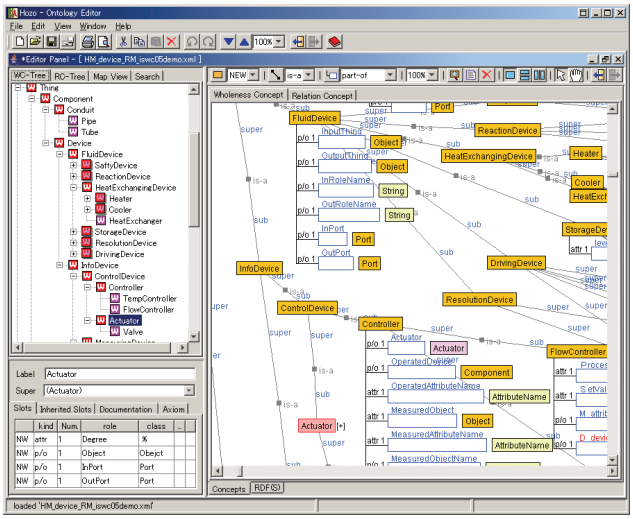

Fig.2. A snapshot of Ontology Editor

We discuss how to organize role concepts and propose a framework for organizing role-concepts in a hierarchy according to their context dependencies [Sunagawa 05].

2.3 Distributed Ontology Development

We assume a situation where a target ontology is divided into several component ontologies and they are constructed individually in a distributed environment (and sometimes in parallel by different developers). Development of ontologies in such a manner applies to many situations: cooperative development, understanding the total picture of conceptual hierarchy, reusing ontologies and so on.

Hozo provides not only a basic mechanism for distributed ontology development (e.g. distributed transparency, access control or versioning) but also effective support for avoidance of inconsistency between component ontologies during their construction process. When a developer changes a component ontology which is referred to by other component ontologies, the change influences on them. And, if the developers would not cope with the influence, it may destroy the consistency between the depending ontologies. To keep and restore them, Hozo helps developers to cope with such a situation based on fundamental theory of ontology. These supports are provided by Ontology Manager and Tracking Panel.

Ontology Manager handles ontologies stored in Ontology Server and manages not only ontologies but also dependencies between them. Ontology developers can easily browse the latest version of the target ontology and know which component ontologies have been changed and which are influenced by the changes. For keeping the consistencies between the ontologies, this information contributes to both of proactive restriction on the influencing ontologies and passive modification of the influenced ontologies. ***Tracking Panel*** is a tool included in Ontology Editor and available for modification of an ontology which is influenced crucially by the change of other ontology. This panel tracks how the related ontology has been changed and lists what countermeasures are supported for coping with the change. A developer selects a countermeasure from the list, and then the ontology is modified semi-automatically for keeping the consistency of the influenced ontology.

2.4 Creating Ontology-based Applications

Hozo provides several functions to develop applications based on ontologies and instance models which is built by its Ontology Editor. It helps users to develop ontology-based applications by the following two ways:

1) Translation of ontologies and models into different formats/languages (hierarchical text, XML/DTD, DAML+OIL, RDF(S), and OWL) for another application
2) Access to Hozo using API implemented in Java

3 Conclusion and Future work

We discussed an environment for ontology development, Hozo, concentrating mainly on its functionalities. Some applications are built using Hozo, such as idea creation support system for materials design, an interface system for an oil-refinery plant operation [Mizoguchi 00]. We have identified some room to improve Hozo through its extensive use. The following is the summary of our future plan:

-Management of ontologies and instance models by version control, updating and reusing.
-Improvement of the ontology development method based on ontological theory of role-concept.
-Import function from different formats (RDF(S), OWL .etc.)

Acknowledgments

We are grateful to Mr. M. Ohta for his valuable discussions and contributions to implementation of Hozo.

References

[Guarino 98] Guarino, N.: Some Ontological Principles for Designing Upper Level Lexical Resources. Proc. of the First International Conference on Lexical Resources and Evaluation, Granada, Spain, 28-30, May 1998.

[Kozaki 02] Kozaki K., et al.: Hozo: An Environment for Building/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship", *Proc. of EKAW2002*, pp.213-218, Siguenza, Spain, 2002

[Mizoguchi 00] Mizoguchi, R., et al.: Construction and Deployment of a Plant Ontology, Proc of EKAW2000, Juan-les-Pins, French Riviera, October, 2000.

[Sowa 95] John F. Sowa: Top-level ontological categories, International Journal of Human and Computer Studies, 43, pp.669-685, 1995

[Sunagawa 03] E. Sunagawa, et al.: An Environment for Distributed Ontology Development Based on Dependency Management, *Proc. of ISWC2003,* pp. 453-468, 2003.

[Sunagawa 05] Sunagawa, E., et al. :Organizing *Role-concepts* in Ontology Development Environment: Hozo, *Proc. of 2005 AAAI Fall Symposium on Roles, an interdisciplinary perspective*, 2005

Outline of the demo presentation

1) Functionalities of Ontology Editor

We demonstrate functionalities of Hozo mainly to show how it treats the role concept. Fig.1 shows a snapshot of the ontology editor of Hozo. In the *browsing pane* "Teacher Role" defined in the context of High school is represented like this. And users can switch three views to define the definition of role concepts in the *definition pane*.

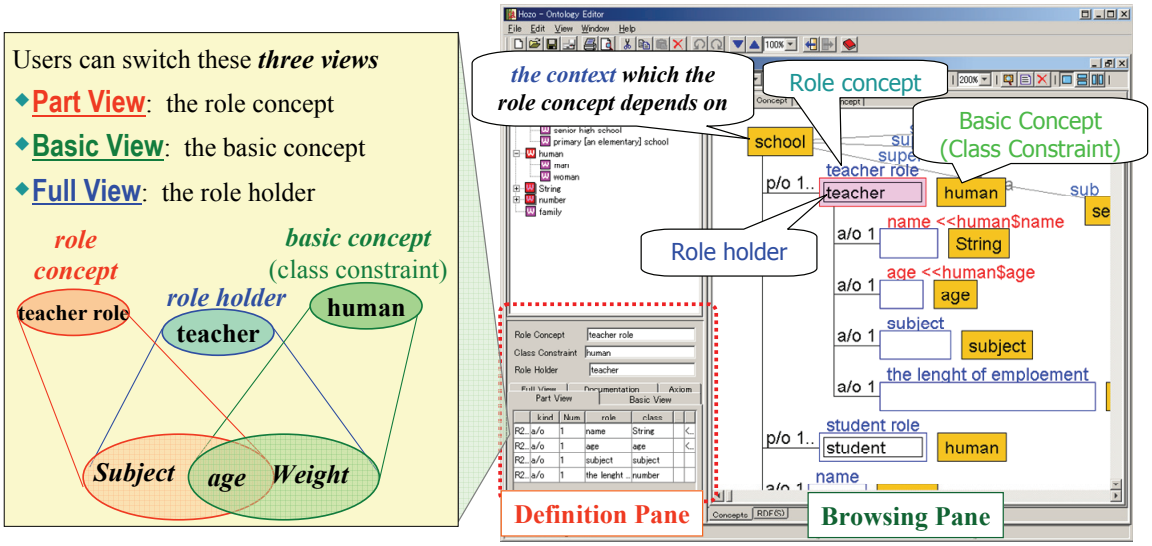

Fig.1 Treatment of role concepts in Hozo

2) Distributed ontology development support based on dependency management

The function for the modification of the influenced ontology

Fig.2 shows the interface for managing dependencies between ontologies, which is called Ontology Manager. It consists of 4 panes:

- *Ontology List* shows a list of ontologies which is registered in Ontology Server.
- *Ontology Viewer* shows dependencies graphically by using nodes and links.
- *Ontology Information Pane* shows the name, file name, author, version, last update of the selected ontology.
- *Dependency Pane* shows the lists of ontologies which have a dependency on the selected ontology.

The function for the modification of the influenced ontology

This function is used when the user opens the influenced ontology to edit it and whenever he/she requests the change information of other ontologies. When the user is going to edit ontologies, this system checks the change of the ontologies it depends on. If there is any change, the *Tracking Pane* shows how the influencing ontology has been changed (Fig.3-a)). And user selects one of the changes, and then the system shows what countermeasures are available (Fig.3.-b). When the user selects one of them, the system modifies the ontology semi-automatically according to the selected countermeasure.

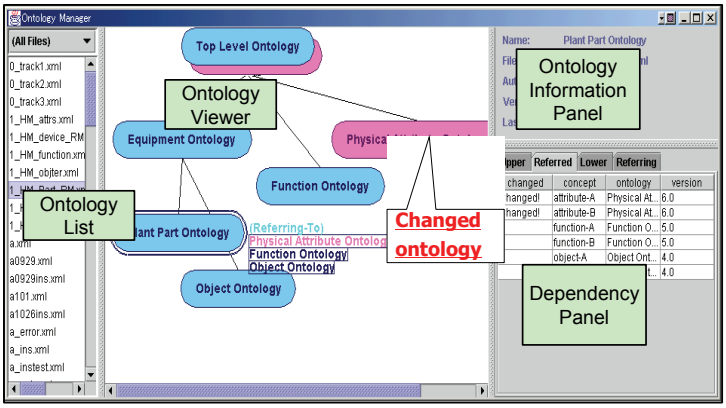

Fig.2 A snapshot of Ontology Manager

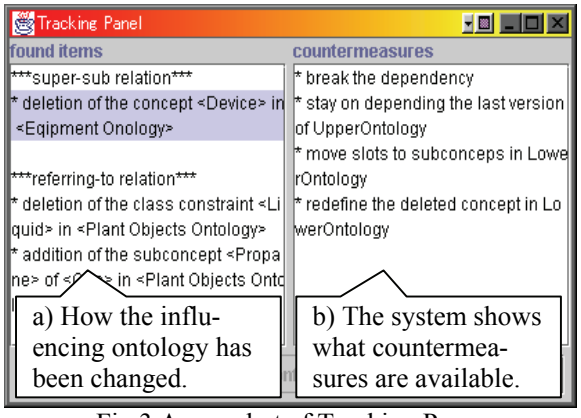

Fig.3 A snapshot of Tracking Pane