Proceedings of the IASTED International Conference
Software Engineering and Applications (SEA 2010)
November 8 - 10, 2010 Marina Del Rey, USA

AN EXTENSION OF AN ENVIRONMENT FOR BUILDING/USING ONTOLOGIES "HOZO" TOWARD PRACTICAL ONTOLOGY ENGINEERING

Mamoru Ohta, Kouji Kozaki and Riichiro Mizoguchi
The Institute of Scientific and Industrial Research (ISIR), Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{ohta, kozaki, miz}@ei.sanken.osaka-u.ac.jp

ABSTRACT
Through the spread of ontological engineering, many technologies and software tool for ontology construction were developed. By using them, many ontologies have been constructed in various domains. On these backgrounds, we have been developing an ontology engineering environment "Hozo" and using the tool to construct a lot of ontologies in various domains such as medical science, bioinformatics, nano-technology, education, environment engineering and so on. Through these practical experiences, we found out many issues concerning ontology construction and have solved them by enhancing the ontological theories and technical functions of Hozo. The number of items of improvements amounts to 67 in both theoretical and practical issues. This paper focuses on the practical issues and presents the improved functions of Hozo. Then, we consider how those functions have been useful for ontology construction through actual uses in six ontology development projects and evaluation experiment of usability of Hozo. Through these extensions, usability and reliability of Hozo have been improved. It also would contribute to development of other ontology engineering environments.

KEY WORDS
ontology, ontology engineering tool, practical ontology engineering.

1. Introduction

With the recent progress of Semantic Web technologies, ontologies serving as the core of these technologies are being built in various technical domains. In the midst of this trend, regarding computer environments used for building ontologies and/or developing application systems based on them (ontology building/using environments), problems are becoming prominent from both theoretical aspects of ontologies for suitably conceptualizing real problems and technical aspects for practical building and usage.

We have been working on the development of Hozo, which is an ontology building and usage environment based on a unique fundamental ontology theory and building methodology that make it possible to handle role concepts explicitly [1, 2], as well as practical ontology building and usage with Hozo in various projects inside and outside our lab [3, 4, 5, 6, 7, 8]. Problems have also surfaced in these practical experiences, from both theoretical aspects and practical, technical aspects of ontologies. The number of functional extensions we implemented in Hozo in order to solve these problems totaled 67. We can say that, through such practical problem solving and functional extensions involved in the problem solving, we have been improving the usefulness and robustness of Hozo. The findings obtained through these efforts will lead to theoretical and technical development of ontology researches and will contribute to the evolution of ontologies into more practical technology.

In this paper, we would like to demonstrate that an ontology building tool has developed into a matured tool that is powerful and usable through the practice of ontology building and usage for long years. In the following, Section 2 gives an overview of the frameworks of Hozo extended in view of demands from the field of ontology building. Section 3 summarizes achievements and evaluation of application of the frameworks to actual ontology building, and discusses an evaluation experiment regarding the effects of improvement in usability. Section 4 discusses comparisons with related research. Section 5 summarizes the achievements of this research and concludes this report by discussing issues to be addressed.

2. Functional Extensions of Hozo and its Applications

2.1 Overview

The life cycle of ontology building and usage is broadly divided into three phases: building, usage, and refinement. Hozo provides a computing environment that assists ontology builders throughout these phases of the life cycle. In the actual practice of ontology building, however, difficult problems often arise. Among these, there have been problems that cannot be handled within the frameworks of existing theories or tools and that require theoretical extensions for proper handling, and problems that require enhancement of tools from technical aspects. Figure 1 lists a total of 67 improvements made in the

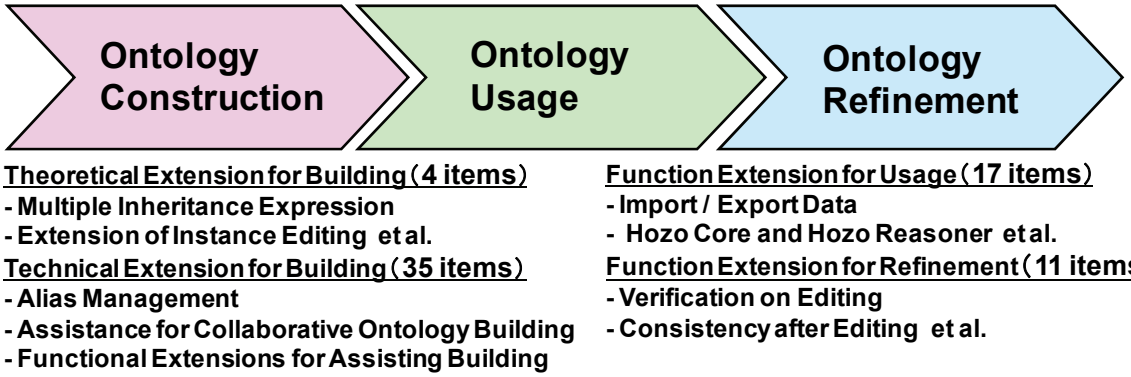

Figure 1. Functional extensions of Hozo

theories and tools of Hozo based on findings obtained through practice, classified on the basis of the phases of the life cycle described above. The following sections discuss the functional extensions for each phase of the life cycle of ontology building in view of such practical demands.

2.2 Functional Extensions for Ontology Building

2.2.1 Theoretical Extension for Ontology Building

In the ontology building phase, a class hierarchy is designed, and concepts and relationships among the concepts are defined. In practice, with problems in ontology building that cannot be handled adequately in existing frameworks, a demand has arisen for clarifying theoretical problems of the frameworks and extending the theories so that the problems can be suitably conceptualized. The main problems were theoretical handling of multiple inheritance expression and instance editing.

Theoretical handling of multiple inheritance expression: In an ontology, an is-a relationship indicates the inheritance of essential attributes that determine the identity of an instance. One problem with using a multiple inheritance expression is that the upper concept from which essential attributes are inherited is hidden. Therefore, it is believed that multiple inheritance expressions should be avoided in order to build right ontologies [9, 10, 11]. In the practice of ontology building, however, there has been a demand for alleviating the difficulty of conceptualization by using multiple inheritance, which is intuitively easy to understand. Thus, as a theoretical framework for avoiding the problem when multiple inheritance expressions are used, we introduced the IS-A relationship, which represents an is-a relationship in a weak sense, i.e., only attributes are inherited without inheriting identity. Furthermore, we set out a usage policy that the IS-A relationship and the ordinary is-a relationship should be used with proper distinction when describing ontologies, and practiced the policy so that the problem described above does not arise when multiple inheritance is used. Figure 2 shows an example of multiple inheritance

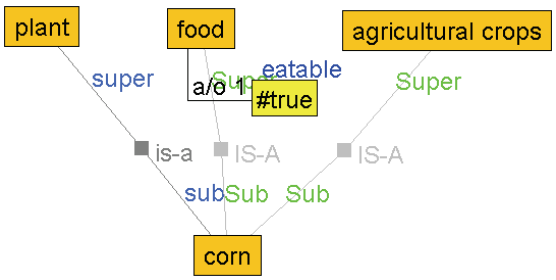

Figure 2. Multiple inheritance in Hozo

expression. In this example, "corn" is essentially defined as a sub class of "plant" by an is-a relation, and two IS-A relationships represent that corn inherits attributes of "food" and "agricultural crops" without inheritances of their identities.

Theoretical extension of instance editing: Regarding this issue, when concepts are defined in practice, there are cases where one sometimes needs to designate a specific instance for value restriction of a slot. There are two cases: (1) instance categorical value as a kind) and (2) ordinary instance of a class. As to the first case, we introduced #operator, which represents an instance called a "species concept", which designates an instance as a categorical value[1]. As to the latter case[2], we introduced a simple instance editing function which enables simple editing of instances on ontology editor rather than instance editor.(see Figure 3) In this example, a class hierarchy of "business" is shown at the left side. On the right side, the definition of "prefecture" designates the class hierarchy as restriction value of "major industry" slot using #operator. Instance of the prefecture can have species concepts such as "#steel business" and "#electric equipment manufacture" as a value of its "major industry" slot rather than instances such as ABC Steel Production Inc.

[1] The latest version of the ontology standard OWL(OWL 2.0) has introduced a similar framework in which a class is handled as an instance by a function called "punning."

[2] While Hozo has an instance editing tool "Instance Editor", it is designed for complicated instance models and seems to be exaggerated for editing small instances.

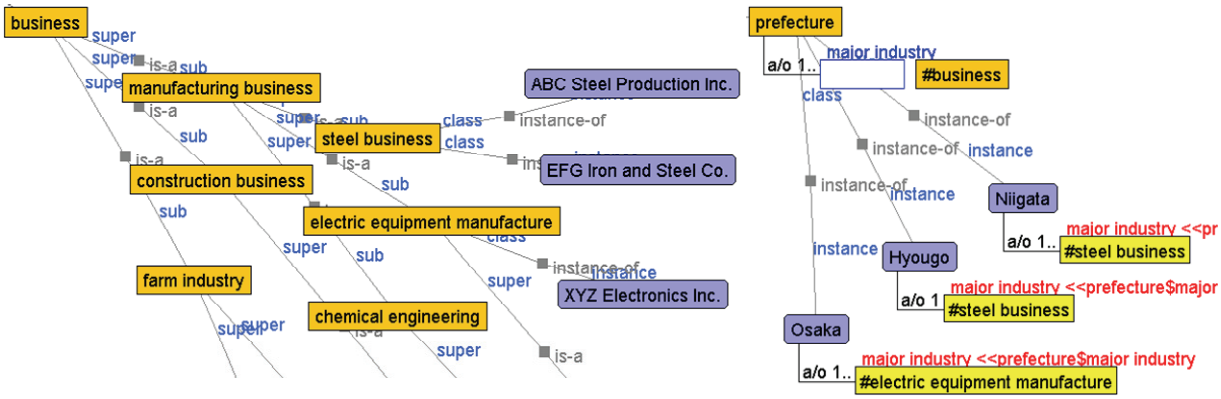

Figure 3. #operator and a simple instance editing function

2.2.2 Technical Extensions for Ontology Building

As functional extensions from technical aspects in view of demands from the field of ontology building, we introduced alias management, collaborative building assistance, building work assistance, etc.

Alias management: When a concept is defined during ontology building, a name is assigned to the concept. On the other hand, in the real world of the target domain, there are cases where multiple names exist for the same concept and the suitable name change according to its users and purposes. Thus, a demand has arisen for suitably managing the names in consideration of various users and usage scenes. Thus, we introduced alias, which allows assigning an arbitrary number of aliases to each concept, and term type, which represents the types of aliases, so that it became possible to edit and manage aliases (see Figure 4). The introduction of this framework makes it possible to write both concept definitions at a semantic level and label descriptions at a surface vocabulary level and to suitably manage the concept definitions and label descriptions.

Assistance for collaborative ontology building: With the increasing scale of ontologies, in order to reduce the building cost and workload, there is an increasing trend for developing an ontology through cooperation among multiple builders, including domain experts, and using the ontology in practice while modifying it as needed [12]. Thus, we proposed a basic design of a distributed development assistance environment that enables sharing of a workspace among multiple users and accessing the same ontology from multiple locations, and developed a prototype [13]. We have improved the prototype for practical use. It includes ontology sharing via a network and a function of visualizing modified parts of an ontology as a function for assisting in conveying modifications that have been made asynchronously to other users. Figure 5 shows a screenshot of user interfaces which show developers which ontology has been changed. The tracking pane (on left hand of Figure 5) lists the changes in ontologies. The changes are shown by three types of icons which represent deletion, modification and addition with lists of concepts which are influenced by the changes in a tree structure. In the browsing pane (on right hand of Figure 5), the ontology is visualized in network structures, and the changed concepts are represented by the same icons. Therefore, the developer can know which concepts are influenced by the change of the ontology. These functions has contributed to smooth communications among co-developers.

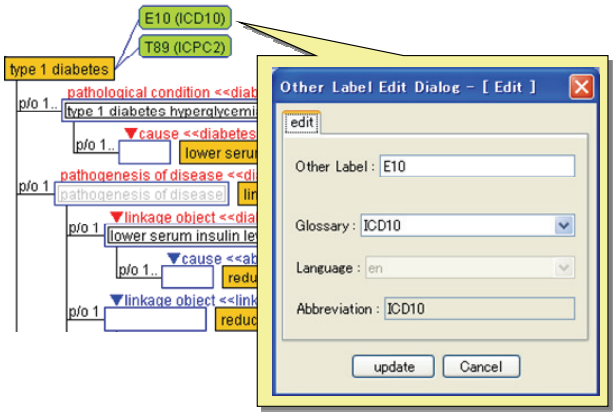

Figure 4. Alias management

Functional extensions for assisting building: We made functional extensions for assisting the building work in individual work spaces, such as jumping about concepts, displaying concepts, and editing operations (Table 1, 18 items in total). Particularly, when the scale of an ontology being built becomes large, there is an increased burden on the builder operating the tool in the limited work space of a computer screen, which could inhibit conception and work. The individual functional enhancements listed in Table 1 are all minor but are appreciated in the course of practical building work. The suite of these functions contributes to realization of a comfortable ontology building environment.

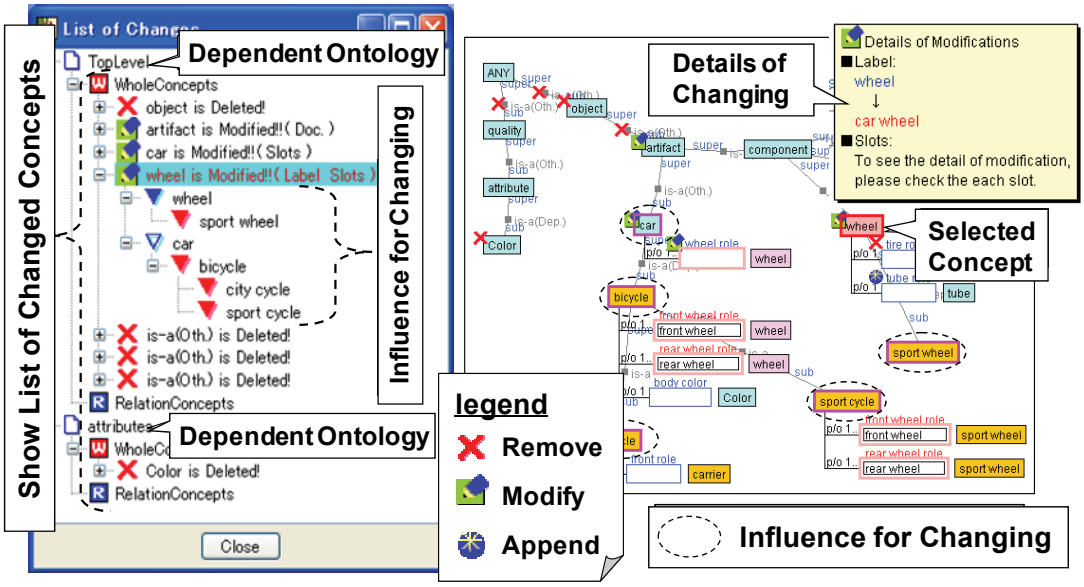

Figure 5. Assistance for collaborative ontology

Table 1. Functional extensions for assisting ontology building (18 items)

Category	Size	Items of Improvement
Moving among Concepts	5	Jumping to Class Constraint, Jumping to Relevant Concepts, Setting Bookmarks, Searching Concepts by Labels, Tracking Selection
Displaying Concepts	5	Display with Sub Window, Highlight on Selection, Display with Map, Zooming, Compact Window Mode
Editing Concepts	8	Copy and Paste, Undo and Redo, Enhanced Support is-a Hierarchy Editing, Alignment Concepts, Printing Ontology, Setting Shortcut Menu, Multilingualization

2.3 Functional extensions for ontology usage

2.3.1 Import/Export Data

In actual ontology building, from the viewpoint of knowledge sharing and reuse, there has been a demand for converting necessary information from existing knowledge sources such as databases and classification systems and incorporating the results into ontologies. Thus, we developed a data import and export function. This function supports input and output in various formats, including the ontology standard OWL and some other formats. The support of the standard is an important feature for improving interoperability of ontologies. The followings are data formats which Hozo supports to import and export:

- OWL
- RDF(S) (*export only)
- HTML (*export only)
- Simple text format (*export only)
- CSV format
- Special purpose text format (* import only)

The simple text format is designed for representing ontologies in human readable format. The special purpose text format is a machined readable format using simple syntax designed for importing data generated by other systems.

2.3.2 Hozo Core and Hozo Reasoner

Meanwhile, in developing an application that uses an ontology, it is necessary to interpret the content expressed by the ontology and to process the results on a computer. Thus, we developed two software modules, namely, Hozo Core (the ontology usage API) and Hozo Reasoner, which made it possible to use ontology processing provided by Hozo from external systems, and made the software modules publicly available as free software[3].

Hozo Core: Hozo Core is an application interface for computer programs to process ontologies and instances which are built by Hozo. It includes primitive functions to treat ontologies such as load, save, access and modify data model of ontologies and instances.

Hozo Reasoner: Hozo Reasoner is software for reasoning an ontology by using the data model of ontology in Hozo Core. It can extract implicit relationships among concepts in an ontology. It includes functions for applying requests to search, retrieve and verifying an ontology.

[3] http://www.hozo.jp/

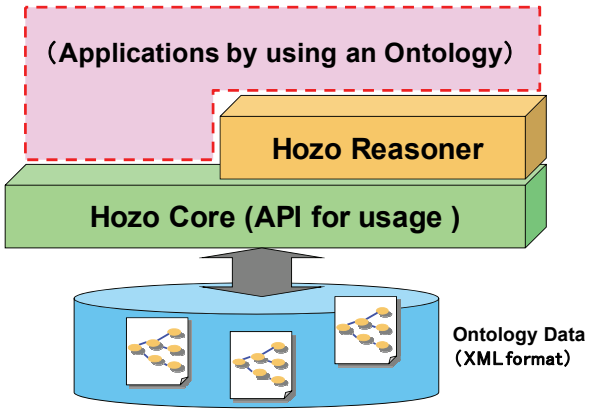

Figure 6. Hozo Core and Hozo Reasoner

They were developed in Java Standard Edition 6 and published as program modules in Java. By using these modules, it is possible to readily develop applications by using an ontology built with Hozo (see Figure 6).

2.4 Functional Extensions for Ontology Refinement

For ontology verification, there are two ways: (1) on-line way and (2) off-line way. In the former case, ontology is verified at any time when any change is made during editing and errors are reported if any. In the latter case, ontology verification is done after editing process and ontology is checked globally. In Hozo, these two ways are implemented. In assistance for ontology editing work, various assisting functions are provided so that users can properly build ontologies in compliance with theories through editing. The latter way includes an ontology integrity verification function, with which the integrity of an ontology as a whole is verified by using the ontology usage Hozo Core and Hozo Reasoner described in Section 2.3.

3. Evaluation and Discussion

3.1 Overview of Ontology Building and Usage in Practice

The ontology building/usage environment Hozo is being used practically in six projects related to ontology building and usage in technical domains such as education, medicine, and nanotechnology. Of the six projects, the authors are leading ontology building in two projects, and external cooperating agencies are leading ontology building in the other four. The extensions of Hozo described in this article have received favorable evaluations as a result of their use in the projects. Table 2 summarizes the use of the extended functions in the projects.

In building a clinical ontology [3], those extensions were appreciated in defining and editing individual concepts. We received an opinion from an ontology builder that desired functions were fully provided and that without the assisting functions it would be conceivably difficult to build such a complex and large-scale ontology exceeding 10,000 concepts. Particularly, the use of aliases for the names of diseases, for which international classification exists, was evaluated as useful. In building a sustainability ontology [7], since sustainability is an interdisciplinary domain ranging over multiple fields, it was necessary to recognize concepts from multiple viewpoints. This ontology was built by utilizing the IS-A relationship for multiple inheritance expression. Furthermore, in building experimental protocol ontology [8], an ontology of learning, instruction and instructional design (OMNIBUS) [6], and a nanotechnology ontology [5], Hozo Core (API) and Hozo Reasoner were used effectively, contributing to the development of applications using ontologies. In building a genetics ontology[4] [4], #operator and the simple instance editing function were used effectively for attribute descriptions and descriptions regarding values. Furthermore, considering that the project was a collaborative building project with an external research institute, the collaborative building assisting function was used actively, and the work would conceivably have been impossible without the change management function. Furthermore, when the OMNIBUS ontology was built, the collaborative building assisting function was used for collaborative building work with overseas researchers.

As for other functions, assisting functions such as building work assistance, editing work assistance, and integrity verification are used in most projects. In particular, the assisting functions are being used actively in projects for building large-scale ontologies, and the functions are contributing to improvement in work efficiency and to quality assurance. Furthermore, the data import and export function is contributing to improvement in work efficiency through import of knowledge from existing resources and to improvement in interoperability of ontologies through output in OWL.

3.2 Usability Evaluation

In practical ontology building, the usability of the building tool is an important factor. Building work assistance serves to improve the efficiency of user's work, and improved accuracy can be expected through appropriate guidance for editing. Thus, of the extended functions of Hozo, regarding two of the improvement related to improved usability, namely, assistance for ontology building work (Section 2.2.2) and assistance for ontology refinement work (Section 2.4), an experiment was performed for evaluating usability with users having no experience of ontology building. According to the result, the number of ontology components that were built

[4] Riken (http://www.riken.go.jp/), together with related agencies, is working on standardization of mouse phenotypes for advancing common usage of a biology ontology, including a genetics ontology, and interoperability among data formats.

Table 2. Overview of ontology building and usage in practice

Ontology (*Size of Concepts / Slots(Avg.)*) \ Extensions	Construction (Theoretical)			Construction (Technical)			Usage				Refinement	
	multiple inheritance	#-Operator	Editing Instance	Alias Management	Collaborative Development	Construct Support	Hozo Core (API)	Hozo Reasoner	Importing Data	Exporting Data	Editing Support	Checking Verification
Clinical Ontology *(32,120 / 7.98)*	○		○	◎	○	◎	◎	◎	◎	○	◎	◎
Sustainability Ontology *(4,515 / 1.08)*	◎		◎		○	◎	○		◎	◎	◎	○
Experimental Protocol Ontology *(1,429 / 13.76)*			◎			○	◎	◎			○	○
Ontology of Learning, Instruction and Instructional Design *(1,433 / 2.76)*		○	○		◎	○	◎	○		◎	○	○
Nano-Tech Ontology *(1,425 / 0.00)*				○		○	◎		◎		○	
Genetic Ontology *(669 / 0.85)*	○	◎	◎		◎	○			○	◎	○	○

Legend) ◎: be used positively ○: be used (empty): be not used very much

all increased compared with the number before introducing the improvements. This can be considered as a result of the improvements in this research, i.e., the efficiency of building work improved, so that the subjects were not hindered with the operation of the tool and were able to concentrate on building work. In addition, the number of integrity errors of the ontologies decreased by 52%, indicating that the improvements also contributed to improvement in the quality of ontology. This can be attributed to proper functioning of the building assisting function on editing operations involving inheritance.

4. Related Work

A typical example of a tool that assists in the life cycle of ontology building and usage is Protégé[5] of Stanford University. Protégé is being used in various ontology building projects, mainly in biology, and various extensions are being added continuously in accordance with user's needs, such as collaborative building assistance. A large number of plug-ins that provides a variety of additional functions has been developed. Furthermore, the NeOn Project[6] proposed an ontology system architecture in which the life cycle of ontology building and usage was taken into consideration [14].

Meanwhile, as efforts regarding practical ontology building, the W3C working group[7] is making efforts for feeding back practical problems to development of building methodologies and related technologies, such as activities for development and standardization of a new version of OWL[8] in which demands from the field of ontology usage are reflected, collection and sharing of ontology design patterns, and creation of software suites related to ontology through various research projects. Also with Hozo, efforts are being made similarly to Protégé and W3C in that, in parallel with development of a unique ontology building/using environment, ontology building in various technical domains is being practiced, and the theories and tool are being improved based on findings obtained through practice. Particularly, regarding theoretical extensions, the theoretical examination of the meanings of multiple inheritance expression and instance editing from the perspective of ontology engineering is a feature not seen in other research. Furthermore, similarly to Protégé and NeOn, Hozo provides a special API and makes standards-supporting modules available in public, and the API and modules are being used for developing various applications.

[5] http://protege.stanford.edu/
[6] http://www.neon-project.org/
[7] http://www.w3.org/2001/sw/BestPractices/
[8] http://www.w3.org/TR/owl2-profiles/

5. Conclusion

This article reported on functional extensions of the ontology building/using environment Hozo based on findings obtained through experiences in practical ontology building. Regarding all 67 functional extensions developed in this research, evaluation from a practical perspective through usage in actual ontology building and an evaluation experiment regarding usability were conducted. As a result, it was demonstrated that, with the extensions both in theory and tool function, Hozo is a tool that is practically usable in actual fields.

An issue to be addressed related to assisting ontology building such that even domain experts not familiar with ontologies can build ontologies properly from an ontological perspective. We believe that active building assistance for domain experts facilitates collaborative building work with ontology experts, which enables deep consideration of ontologies regarding technical knowledge.

In addition to three phases discussed in this paper, the life cycle management is an important function for ontology development tool. The quality assurance of an ontology is an important factor that determines the function. Especially, we consider that a framework for domain experts to evaluate the content of ontologies in ontology construction and refinement phases is one of serious issues for quality assurance. Thus, we are currently developing such a framework in Hozo.

Acknowledgements

This research was supported by the Ministry of Health, Labour and Welfare of Japanese Government as Development business and research of "Medical-knowledge-based database for medical informatics system."

References

[1] Kozaki, K., Kitamura, Y., Ikeda, M., Mizoguchi, R.: Hozo: An Environment for Build-ing/Using Ontologies Based on a Fundamental Consideration of "Role" and "Relationship", Proc. of the 13th International Conference Knowledge Engineering and Knowledge Man-agement (EKAW2002), pp.213-218, (2002).

[2] Mizoguchi, R., Sunagawa, E., Kozaki, K., Kitamura, Y.: A Model of Roles within an Ontology Development Tool: Hozo, Journal of Applied Ontology, Vol.2, No.2, pp.159-179, (2007).

[3] Mizoguchi, R., Kou, H., Zhou, J., Kozaki, K., Imai, K., Ohe., K.: An Advanced Clinical Ontology, In Proc. of International Conference on Biomedical Ontology (ICBO), pp.119-122, (2009)

[4] Masuya, H., Mizoguchi, R.: Toward Integration of Mouse Phenotype Information, Proc. of the Second Interdisciplinary Ontology Meeting, pp.35-44, (2009)

[5] Komiyama, H., Yamaguchi, Y., Noda, S.: Structuring knowledge on nanomaterials processing, Chemical Engineering Science, Volume 59, Issues 22-23, pp.5085-5090, (2004)

[6] Hayashi, Y., Bourdeau, J., Mizoguchi, M.: Using Ontological Engineering to Organize Learning/Instructional Theories and Build a Theory-Aware Authoring System, International Journal of Artificial Intelligence in Education, Vol. 19, No. 2, pp. 211-252, (2009)

[7] Kumazawa, T., Saito, O., Kozaki, K., Matsui, T., Mizoguchi, R.: Toward knowledge structuring of sustainability science based on ontology engineering, Sustainability Science, Vol.4, No.1, (2009)

[8] PREIMS: http://preims.tanpaku.org/preims/

[9] Mizoguchi, R..: Tutorial on ontological engineering - Part 3: Advanced course of ontological engineering, New Generation Computing, OhmSha&Springer, Vol.22, No.2, pp.198-220, (2004)

[10] BFO: http://www.ifomis.org/bfo

[11] DOLCE: http://www.loa-cnr.it/DOLCE.html

[12] Noy, N. F., Chugh, A., Alani, H.: The CKC Challenge: Exploring tools for collaborative knowledge construction. IEEE Intelligent Systems 23(1), pp.64-68, (2008)

[13] Kozaki, K., Sunagawa, E., Kitamura Y., Mizoguchi, R.: Distributed and Collaborative Construction of Ontologies Using Hozo, Proc. of Workshop on Social and Collaborative Construction of Structured Knowledge, (2007)

[14] Tran, T., Haase, P., Lewen, H., Muñoz-García, Ó., Gómez-Pérez, A., Studer, R.: Life-cycle-Support in Architectures for Ontology-Based Information Systems, Proc. of the 6th International Semantic Web Conference, ISWC/ASWC 2007: 508-522, (2007)